Security Responses in Software Development

Journal Item

Security Responses in Software Development

TAMARA LOPEZ, School of Computing and Communications, The Open University, UK
HELEN SHARP, School of Computing and Communications, The Open University, UK
THEIN TUN, School of Computing and Communications, The Open University, UK
AROSHA BANDARA, School of Computing and Communications, The Open University, UK
MARK LEVINE, Department of Psychology, University of Lancaster, UK
BASHAR NUSEIBEH, School of Computing and Communications, The Open University, UK, Lero-The Irish Software Research Centre, Republic of Ireland

The pressure on software developers to produce secure software has never been greater. But what does security look like in environments that don't produce security-critical software? In answer to this question, this multi-sited ethnographic study characterises security episodes and identifies five typical behaviors in software development. Using theory drawn from information security and motivation research in software engineering, this paper characterizes key ways in which individual developers form security responses to meet the demands of particular circumstances, providing a framework managers and teams can use to recognize, understand and alter security activity in their environments.

CCS Concepts: • **Security and privacy** → **Human and societal aspects of security and privacy**; • **Software and its engineering**; • **Human-centered computing** → *Empirical studies in collaborative and social computing*;

Additional Key Words and Phrases: Security, Developers, Software engineering

1 INTRODUCTION

Software developers — whether programmers, testers, designers, or product managers — typically make hundreds of decisions every day. Very few of those decisions have obvious or direct security implications. And yet, the pressure on developers to produce secure software has never been greater as the potential personal, reputational, and monetary costs of security breaches are high [12]. To address the need for security within code, there is a growing number of cybersecurity tools, guidance, training materials, and case studies. Unfortunately, the number of breaches seems to be undiminished. Analysis of the top 5000 websites in 2018 highlighted that 8%-21% of them were susceptible to attack through different types of known software vulnerabilities [64]. Entries in the National Cyber Awareness System bulletins [15] regularly include issues that feature on the most recent Open Web Application Security Project (OWASP) top 10 list [69], such as those that involve structured query language (SQL) injection.

The continued prevalence of vulnerabilities in code raises questions about the role developers play in keeping software secure. Existing work examining security practices has focused on the issue of usability for

Authors' addresses: Tamara Lopez, School of Computing and Communications, The Open University, Milton Keynes, UK, tamara.lopez@open.ac.uk; Helen Sharp, School of Computing and Communications, The Open University, Milton Keynes, UK, helen.sharp@open.ac.uk; Thein Tun, School of Computing and Communications, The Open University, Milton Keynes, UK, thein.tun@open.ac.uk; Arosha Bandara, School of Computing and Communications, The Open University, Milton Keynes, UK, arosha.bandara@open.ac.uk; Mark Levine, Department of Psychology, University of Lancaster, Lancaster, UK, mark.levine@lancaster.ac.uk; Bashar Nuseibeh, School of Computing and Communications, The Open University, Milton Keynes, UK,, Lero-The Irish Software Research Centre, Republic of Ireland, bashar.nuseibeh@open.ac.uk.

end users [3] and for developers [27, 39], while other research has focused on identifying the psychological and cognitive factors that lead to security vulnerabilities in code [48]. There is broad agreement in the research community that developers need support in writing secure code [1], and researchers are exploring ways to provide help, for example by raising awareness with software developers about security issues [46, 71], or improving engagement by bringing security and engineering teams together [44].

In a similar vein, this research is developer-centred, examining in more detail the practices of software engineers who are not security specialists, but are tasked with building systems that must be secure. The focus is to understand in more depth how individual developers prioritise security, communicate about security issues within their teams, and the ways in which the security goals of companies and clients influence the tasks undertaken by developers in daily work [54]. This study aims to provide insight into the social and human aspects of security in software development, asking:

- RQ1. Where can security be found in "ordinary" software development environments?
- RQ2. How do non-specialist developers engage with security in practice?

In answer to these questions, this paper reports the outcomes of a multi-sited ethnographic study undertaken at two companies within the United Kingdom (UK). The report analyzes findings from fieldwork conducted at both sites, and includes a situated, developer-centred account of security practice. The paper also includes a set of empirically-informed inferences that exemplify how individual aims intertwine with team and organizational concerns to produce security activity. Positioned at the intersection of individual, team and organizational practice, the findings comprise an empirical baseline that managers and teams can use to recognize, understand and alter security activity in their own environments.

The following pages are organized as follows. Section 2 provides background for the study's examination of the social and cultural aspects of security practice. Section 3 describes the methods used to collect and analyze data. Section 4 characterizes security practice at both field sites. Section 5 categorizes findings from both sites into a set of inferences about how security responses form in professional software development environments. Section 6 provdes limitations to the work, while Section 7 discusses the implications of findings. The article concludes in Section 8.

2 BACKGROUND

Security is recognised to be socio-technical [7], marked by the actions people take as they encounter measures put in place to provide protection. Security measures are defined by policy and are increasingly enacted by software engineers given the task to implement security within software. Crucial to the success of many security initiatives is engineers' use of accepted programming and design principles such as input sanitation, or the principle of least privelege and defense, and technologies, such as TLS/SSL, digital certificates and public key cryptography. However, successful security is not only about technical skill and knowledge. It is also social, dependent on attitudes and work practices that are "suitable" [7] for meeting security goals.

This section provides a survey of literature that addresses three social dimensions of security. Taken together, the sections align the research conducted in this study with a notion of secure coding practices within professional development that extends beyond technologies and techniques to accomodate the social and cultural aspects of security.

2.1 Security Attitudes and Hindrances

Within software engineering, “suitable” attitudes [7] are often described in terms of mindset [56]. Software engineers are encouraged to think like attackers, to consider the ill intent or aims that other people might have toward the system they are building. In imagining how an attacker might try to gain access to a system — and what an attacker might do when they gain access [60] — developers also need to take a defensive position. They must find and acknowledge weaknesses or flaws in their software that might make it easier for attacks to take place.

As a part of adopting a security mindset, the broad expectation is that developers will consider how to find and mitigate weakness at each stage of the software development lifecycle [37]. In so doing, engineers can ensure that software architectures and implementations address current [15] and relevant threats such as those that appear on the OWASP Top 10 List [69].

One difficulty in fostering the right mindset is that the need for developers to actively engage with security is intermittent. In many environments, developers rarely need to use security techniques such as cryptography in their software. When they do, using security implementations in available application programming interfaces (APIs) can be difficult [39]. Another concern is the methods developers use to fill in gaps of understanding. One way developers augment their own experience is through the use of online sources [33]. App developers report that they gain secure coding skills using internet sites and podcasts[72]. Developers have also been shown to share perceptions and informally learn about security as they discuss programming problems on answer sites like Stack Overflow (SO) [32]. However, guidance in online sources has been shown to be incomprehensive and unsound [2], raising concerns about work practices around security. The concerns have warrant: the choices developers make can lead to vulnerabilities in code, as — to take one instance — when insecure code snippets are cut and pasted from online sources [23].

Even when they are knowledgeable, developers can overlook security in the midst of tasks [42]. A recent series of experiments run with students, freelance workers, and professionals found that participants in each environment did not write code to securely store passwords unless they were given an explicit prompt [40, 41]. Whether it is a question of poor understanding or oversight, the findings about prompting raise questions about the responsibility programmers have to keep software secure. It may be that security is a secondary concern [1] and, like other non-functional requirements [65], must be prioritized alongside the tasks developers complete to meet organizational demands for production.

2.2 Security in Organizations

Workers play a key role in delivering information security for organizations [30]. As users of technology with access to sensitive information resources and systems, workers comply with organizational policies through automated controls [26] that enforce general rules about what can and cannot be done. Awareness and understanding of how to comply with organizational security policies are supported through training and education provided by organizations [13]. The aim is to develop a strong security culture, in which workers not only comply, but also have a personal commitment to being secure [25]. It is through a high level of individual commitment that secure behavior on the job is said to become so ingrained it is unconsciously performed [45].

In the context of software development, developers enact security mechanisms, but also must cultivate an understanding of how particular techniques and actions taken within code provide assurance that software complies with security policy [6]. The research suggests that this understanding is difficult to foster in practice. Activities designed to raise awareness, such as providing access to pen testers, are perceived by developers to be helpful, but may not always translate to changes in behavior [46]. In examining software development at different points in the lifecycle, Assal and Chiasson found that in many cases, best practices for improving security in code were simply ignored by developers, on the basis that they were considered to be burdensome [9]. Related research looking at engineers' perceptions of privacy similarly found that participants thought privacy was important, but technically difficult to implement, in part because mechanisms and policies were perceived to be vaguely defined within companies [11].

The nature of the software being produced in organizations may also matter. Developers writing security-critical software work in environments with organizational support to prioritise security within the software development lifecycle. However, the environment within "ordinary" software development environments may not include formalized processes that ensure that risk is properly managed and that security is fully considered [7]. Upholding security may instead fall to the efforts of individuals who take a personal interest [72]. The surroundings matter: as has been shown in privacy research, it is possible that even if engineers are willing to take responsibility for security, social factors within the broader environment may discourage them from taking action [28].

These studies point out that individual commitment to being secure does not ensure compliance with security policy. There is a further point to make. When a company's security implementation does not meet workers' needs, they have been found to actively devise adaptations or techniques to allow them to remain productive [45]. Such "shadow" practices often indicate that workers are isolated from security decision making within companies, and as a result, prioritse being productive over behaving securely [30].

2.3 The Social Side of Security

Work practices are influenced by organizational surroundings, but also through networks of peers [47]. It has been argued that workable security is not championed by a single person but is instead created by a mix of non-specialists who contribute to a secure workplace from within their own competencies [10]. Such groups include individuals for whom behaving securely hinders productivity, others who primarily follow policy, and groupings of individuals that actively discuss the social imperative for security and also challenge perceived gaps in organizational policy and process.

Social interactions in the workplace are also recognised to be a part of security within software engineering. In a study examining social influences on security tool adoption, nearly half of the participants were found to learn about tools through the recommendations of co-workers [75]. Social interactions outside the engineering fold are also important. When members of security teams work alongside software engineers, engagement with security and secure development practice improve [44]. There are also indications that within software engineering, the "mix of individuals" extends to social connections made online. Xiao et al.'s study of tool adoption also found in some cases that developers trusted engineers with a high reputation online more than colleagues in the office [75], a finding explored in more detail by van der Linden et al., who determined that developers select on-line sources in part based on surface features within posts that signal — sometimes erroneously — that an answer is authoritative [68].

These studies point toward the importance of the communities within which developers operate: communities within a team, an organization, and across the profession. Use of programming languages, programming paradigms like object oriented programming, and software quality initiatives are influenced by community and culture within software practice [59]. Developers' engagement in peer-to-peer interactions and the cultures that form around them have been found to bring about lasting cultural change within the software developer community, for example through new technology adoptions [51], and continued adherence to processes and methods [50]. There is every expectation therefore that a deeper examination of the professional cultures developers inhabit will yield a greater understanding of how technical, organizational, and social factors impact security in the workplace.

3 METHOD

Prompted by the discrepancy between the wide availability of security tools and the ongoing problem of breaches and vulnerabilities, the ethnographic method was selected within the Motivating Jenny project [1] to examine the technical, organizational and social dimensions of security highlighted in the prior section.

Ethnography is used to study peoples' actions and accounts of actions from the perspective of the insider [52], and allows researchers to develop understanding about what practitioners working in socio-technical environments do and why they do it [5]. Ethnographic studies play a number of roles in empirical software engineering. They can inform the design of software engineering tools or, as in this work, provide insights that can be used to improve process [58].

In common with other ethnographic studies, the research was not conducted to a fixed design, but was flexible [52]. Research questions were pursued and refined [29] through collection and reflection upon data gathered within a range of activities and sources [70]. To gain sufficient access to naturalistic practice [19], the decision was taken to conduct field studies in multiple sites. The organizations that provided access were known to the research team as the second author had studied both in the recent past. However, the previous studies were not related to security practices. The research was organized as a multi-site ethnography [34], described in more detail in the following section.

3.1 Multi-sited Ethnography

Multi-sited ethnographies acknowledge that some cultural phenomena are not spatially bounded [21] and cannot be accounted for through a localized focus on a single site. The approach requires researchers to draw connections between data collected from different environments and challenges them to change perspective about the object of inquiry [35] by acknowledging multiple perspectives held by participants in different places [35]. In the case of this research, through collaborations formed with industry partners in two professional environments, the researchers were able to maintain a critical stance toward security as a phenomenon [29] and to challenge conceptions [58] of the role developers play in keeping software secure [18] as they are commonly reported in trade and research literatures.

In practice, the multi-sited approach meant that the researchers engaged with participants at the two field sites in different ways. The gatekeeper at Site A was fully briefed about the study's focus on security practice among developers. However, to reduce reactivity [29], participants at the site were only gradually

[1] https://ordo.open.ac.uk/projects/Motivating_Jenny_to_Write_Secure_Software_Community_and_Culture_of_Coding/76284

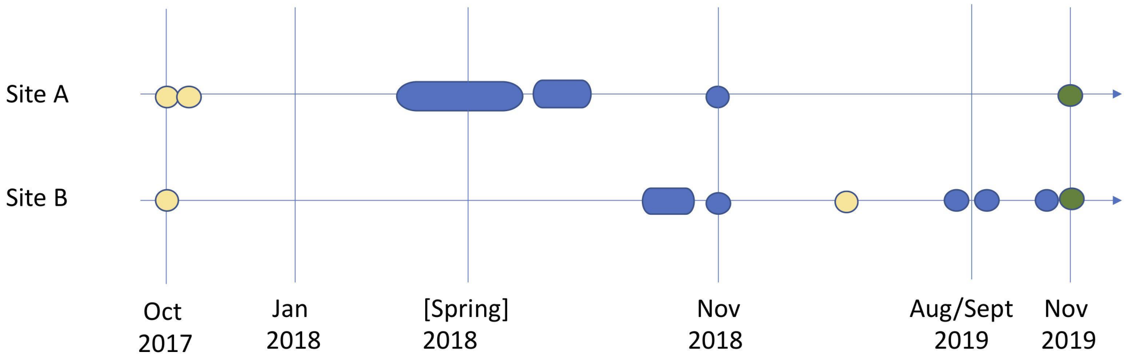


Fig. 1. Timeline of site visits. Yellow dots indicate planning meetings. Blue dots indicate site visits on a single day that included observation, interviews and workshops. Blue blocks indicate a visit spanning multiple days. Green dots indicate member checking performed with a questionnaire.

made aware of the research focus in the course of multiple interactions. Observations made in Spring 2018 were followed up within semi-structured interviews and a feedback and workshop session held in November, 2018, a timeline illustrated in Figure 1. As a result, interactions at Site A produced insight into how security naturally occurs, that is, how it "comes up" in daily practice.

By contrast, participants at Site B were informed from the initial planning meeting in 2017 that the project had as its focus the investigation of security practices among developers. In this meeting, members of technical staff were openly asked about their views on security, and were invited to ask the researchers questions about how the topic was being investigated. Subsequent access to developers was delayed and constrained at this site. For this reason, interactions in late summer, 2019 were organised to collect data about how security fits into project work that includes interactions with clients, providing additional insight into the ways that organizational context intersects with developer practice.

The following subsections introduce each organization and give an overview of their engineering practices.

3.1.1 Workforce Management (Site A). The first organization develops and maintains a single, expanding suite of workforce management software. Originally an independent organization, it was acquired in mid-2016 by a global software company. The offices house about 100 employees, 40 of whom comprise software development teams. A further 16 engineers who work with teams based in this office are located overseas. Some overseas developers work alone; others are co-located with other developers, who may or may not be working on related projects.

At the time of the study, the site was in the process of integrating a set of applications with those developed by the parent company. The software development teams use Scrum-based Agile practices. Each team is assigned a scrum master and one or two product owners; each scrum master and product owner may be associated with one or more teams. In addition, there is one user interface specialist, one UX specialist and one technical author, who work across the teams. All teams follow a release cycle of 8 weeks organized into three two-week sprints dedicated to the product backlog, followed by 2 weeks for making fixes and regression testing. Developers often interact with clients by proxy, through front-line contacts on the sales and service teams.

3.1.2 Management Consultancy (Site B). The second organization is a software house that has a history of engineering solutions to meet tricky technological problems, and has experience working across a broad range of industries and employing different languages and platforms. Due to the wide variety of projects they managed, engineers usually collaborated with additional teams at partner agencies or client-owned teams who were often not co-located for an entire project. This group also has a strong engineering culture; the preferred approach to all projects has been for early and consistent input from engineers, with a lead engineer being assigned to manage the project from the technical point of view.

This software house was acquired by an international management consultancy company in November 2016. At the time of the study, they were integrating themselves into the larger firm. The engineering approach was changing: engineers who formerly worked in the software house were no longer as involved in identifying requirements, and had less interaction with the client. Instead, a consultant (or group of consultants) interfaced with the client to identify, analyse, and document requirements. The majority of engineers from the organization were working at client sites for most of the week, entering the client's environment to develop software. This involved interacting with employees of the client organization.

Table 1. Data Corpus

Field Site	Activity	Participants	Sources
Site A	Observation	3 teams of 3–8 developers, scrum masters & product owners over one sprint	Fieldnotes; audio recordings; photographs; output from tools
	Contextual Interviews	VP product development, technical product owner, security officer	Audio recordings, partial transcriptions
	Semi-structured interviews	13 developers	Audio recordings, transcriptions
	Feedback session	8 developers	Audio recordings
	On-line Questionnaire	14 respondents	21 open and closed questions
Site B	Observation	1.5 days, ∼15 developers, managers	Field notes; photographs
	"Ice breaker" project interviews	4 developers	Field notes, audio recordings, partial transcriptions
	Contextual Interviews	2 technical managers	Field notes, audio recordings
	Semi-structured interview	1 developer	Field notes, audio recording
	Modelling Exercise	8 engineers; 2 technical managers	Video; audio recording, photographs
	On-line Questionnaire	5 respondents	21 open and closed questions

3.2 Data Collection

Field data were collected over a period of 2.5 years through observation, contextual and semi-structured interviews and workshops. Figure 1 shows a timeline of activity at Sites A and B, while Table 1 gives an overview of data collection activities.

At Site A, data collection began within two planning meetings and several email exchanges in 2017. Data were subsequently collected at four different points in time during 2018. At Site B, the researchers met with line management and members of the engineering team in October, 2017 to explain and plan the study. Data were collected at different points in time, beginning in November, 2018 and finishing a year later in 2019. Data collection was undertaken under the approval of the Open University human research ethics committee (HREC). Participants were informed about the study, asked for their consent to participate, and were given information about how to withdraw from participation. The researchers signed non-disclosure agreements (NDA) with both field sites. The NDA agreements and ethnographic nature of the data collection preclude data sharing.

3.2.1 Participants. Twenty-three developers directly informed this analysis through interviews, observations and participation in workshop sessions. Thirteen were located at Site A and 10 at Site B. Developers at both sites were employed to work directly on software produced for organizations as a part of engineering teams. Participants included application and interface programmers, testers, product- and technical managers. The informants at Site A were primarily co-located, but had experience working remotely, and in interacting with remote team members. At Site B, engineers frequently varied their location to be on client sites or in their organization's central offices. Table 2 gives an overview of participants at each site.

3.3 Analysis

Analyses were of two main kinds: descriptive analyses were performed to identify patterns in the data, reported in Section 4, followed by theoretical analyses, in which patterns in the data were explained through comparison and in relation to related literatures [8], reported in Section 5. The analytic aims were two-fold:

(1) Descriptively, to produce an authentic account [5] of the fieldwork that is also authoritative, convincing the reader of the legitimacy of what was seen and reported by participants [5].
(2) Theoretically, to elicit and explain a set of inferences detailing essential connections between characteristics that were observed in the field [61].

As depicted in Figure 2, research was organised in three phases. Analysis began at the point of data collection as first two authors formulated ideas about how to refine the research problem, identified additional sources of data, and gave preliminary interpretations. Data from each field site were catalogued and annotated, and cursory evidence was found during this process to suggest possible connections to motivation and individual career paths. These insights were pursued through examination of related literatures within information security, management science, and software engineering. Accounts were written that described patterns in the field data, surfaced through an inductive examination of interview and observational data. Following Small [61], these findings were compared and interpreted to identify a set of inferences about how developers respond to security.

Table 2. Participants at Sites A and B

ID	Role*	Site	Age	Years of Exp.*	Time at Org*
P1	SE	A	24	4	1
P2	Lead SE	A	45	20	20
P3	Sr. SE	A	46	13	1
P4	Lead SE	A	29	10	3
P5	Software Def Mgr.	A	32	12	4
P6	Tester	A	40	17	17
P7	Sr. SE	A	67	45	17
P8	Sr. SE	A	28	6	15mo.
P9	Principal SE	A	35	10	3
P10	SE	A	29	8	2
P11	Tester	A	34	4.5	2.5
P12	Senior SE	A	40	18	5.5
P13	SE	A	31	4	15mo.
P14	Sr./lead SE	B	46	26	17
P15	SE	B	30s	14	1
P16	Sr./lead SE	B	35–45	16.5	2
P17	Sr./lead SE	B	25	2	2
P18	SE	B	25	3	1.5 mo.
P19	SE, Release Eng.	B	41	19	1.5 mo.
P20	Software Dev	B	30	6	1.5
P20	Lead Front-end Dev	B	27	7	9mo.
P21	Sr./lead SE	B	45	24	5
P22	Sr./lead SE	B	56	33	5
P23	Sr./lead SE	B	30s	n/a	n/a

* Roles, years of experience, and time at organization were self-reported by participants.

Though described in terms of stages, the process was iterative and reflexive. Analysis entailed listening to what participants at field sites said, but also considered how participants' reports might have been shaped by their interactions with the researchers, and with other aspects in their environment [29], including other developers, the workplace, and within the broader software development profession. In addition, the fieldwork reported in this paper ran in parallel to other activities that examined security in online environments, practitioner engagements, and workshops. Interactions within each of these streams were examined for evidence of security activity that could help explain or call into question [5] the "ways of thinking and working" [7, p.856] observed and reported in the field sites.

The following subsections give insight to activities that provided structure to analysis, alongside representative examples drawn from the data.

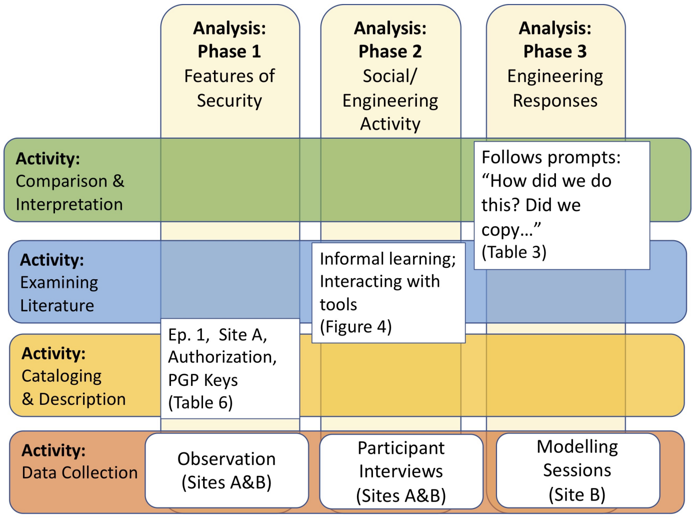

Analysis: Phase 1 Features of Security
Analysis: Phase 2 Social/ Engineering Activity
Analysis: Phase 3 Engineering Responses
Activity: Comparison & Interpretation
Follows prompts: "How did we do this? Did we copy…" (Table 3)
Activity: Examining Literature
Informal learning; Interacting with tools (Figure 4)
Activity: Cataloging & Description
Ep. 1, Site A, Authorization, PGP Keys (Table 6)
Activity: Data Collection
Observation (Sites A&B)
Participant Interviews (Sites A&B)
Modelling Sessions (Site B)

Fig. 2. Overview of the phased analysis, as applied to interview and observational data from Sites A and B. Data were catalogued and features of security were described within accounts; literatures within software engineering, management science, and information security were consulted to identify topics, and; data from different sources were compared with one another to produce the interpretation, described in Sections 3.3.2 and 3.3.3. Diagram adapted from [14]
.

3.3.1 *Phase1: Features of security.*

To support RQ1, this phase established a catalog of security work episodes, described in Section 4.1.2 and included in Appendix A.2. The catalog included the security aim for each episode, the perceived source of the threat (e.g. insider/outsider), the security mechanism, and the "source" of security in the episode (e.g. a developer, a company, a tool, a policy). The catalog was created through examination of observational, interview and workshop data, as described in the numbered points that follow.

1. **Observation, Context (Sites A&B)** In Sites A and B, non-participatory observation was used to gain a sense of the rhythms of practice [58]. Observations were driven by the activity of the developers within teams, but were focused on observing instances of security-related practice as they occurred. This led to the collection of a set of traces of security that gave contextual support to RQ1, highlighting aspects of security awareness within each site, and providng information to support interviews and other interactions. For example, at Site A, bunting with colored flags was strung around the office to mark important milestones. One flag was marked with the text "bcrypt", indicating that the engineering teams had undertaken work in a prior release to secure user passwords. The flag was used as a deepening probe in an interview to situate security activity of the department within the personal chronology of a participant's account.
2. **Participant Interviews (Sites A&B)** Work episodes were also identified at Site A from data collected through semi-structured interviews taken with members of the teams that had been observed. Interviews were conducted by two of the researchers who also performed observations. The interviews included questions of two types. The first set examined aspects of the participant's career development and are not reported here. This study analyzed data from the second set of questions— adapted from use in political [62] and everyday security [17] contexts— that asked about attitudes and beliefs about

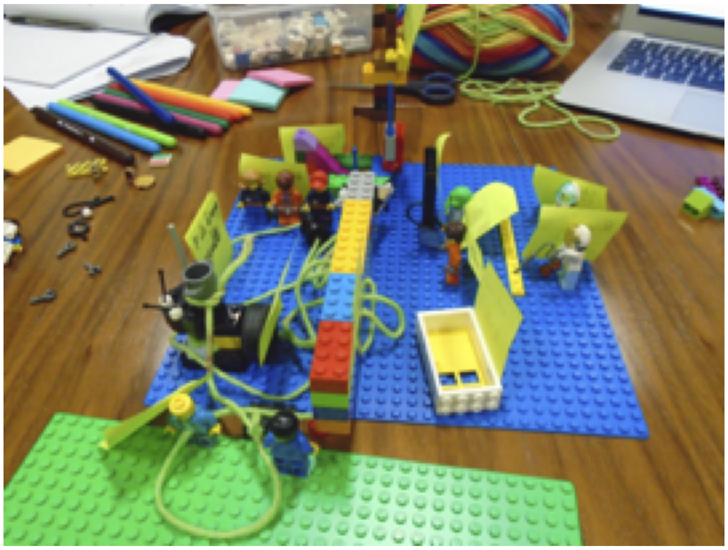


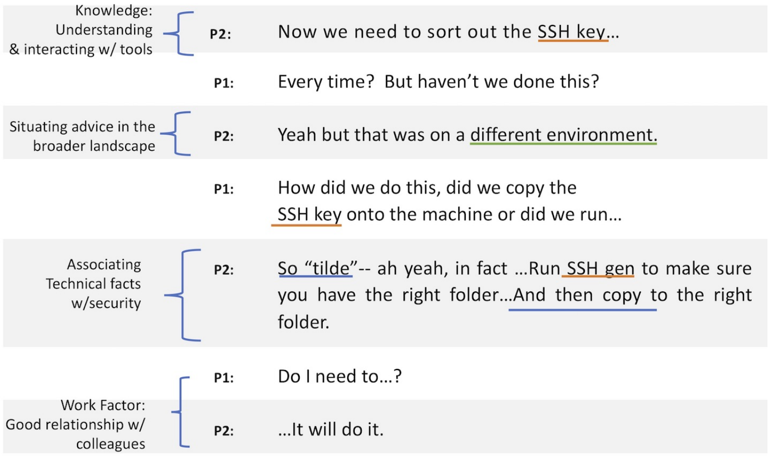
Fig. 3. Modelling exercise. Participants from Site B worked in pairs to create a physical model of a project that included a security aspect and represented policies, technical or other measures, and social aspects. Each session was filmed to capture the workspace, participants' hands and discussion.

security and software development in the organization. These questions can be read in Appendix A.1. At Site B, incidents were identified within data collected during short "icebreaker" interviews and within one longer, semi-structured interview.

3. **Modelling Sessions (Site B).** Work episodes for Site B were also identified through data collected during modelling sessions held in 2019. Tailored after the creative securities [16], the sessions were designed to provide insight into how security fits into project work that includes interactions with clients. Five hour-long sessions were held with pairs of engineers and managers. In each session, the participants were asked to create a physical depiction of their working environment and to identify aspects of security within it that represented policies, technical or other measures, and social aspects (see Figure 3).

3.3.2 Phase 2: Engineering and social activity. In this phase, and contributing to RQ2, interview and observational data were examined for evidence of engineering and social activity. Lending structure to this process, data from each work episode (see Table 6) were used to categorise and provide high-level descriptions for the episodes, and to characterise the security knowledge employed within them.

Next these factors were used to annotate conversations held between developers at both sites (see Figure 4). Affirming prior findings in studies conducted online [32], developers in the field sites were observed to form small, informal relationships with one another to give and receive help in solving problems that have a security aspect. In particular, exchanges indicated that within common engineering tasks, developers provided each other with focused assistance about individual questions, associated technology facts about tasks with security problems, and situated advice within the broader security landscape.



P2: Now we need to sort out the SSH key…

P1: Every time? But haven't we done this?

P2: Yeah but that was on a different environment.

P1: How did we do this, did we copy the SSH key onto the machine or did we run…

P2: So "tilde"-- ah yeah, in fact …Run SSH gen to make sure you have the right folder…And then copy to the right folder.

P1: Do I need to…?

P2: …It will do it.

Fig. 4. This figure shows an example of an informal conversation observed at Site A. Annotations highlight security knowledge [63], features of security in interactions [33] and work factors [24]. For other details about this episode, see episode 1 in Appendix A.2)

This confirmed impressions that engineering and security activity are similarly intertwined within social interactions in on-line and face-to-face settings. The conversations also suggested that developers have an inner work life in which they respond to events [4], leading toward the activities undertaken in Phase 3.

3.3.3 Phase 3: Engineering responses to security. In this phase, and providing the theoretical contribution, a framework was developed to explain the interplay between security features identified in phase 1 and the engineering and social factors isolated in phase 2.

As a part of this, extracts from interview and observational audio data were selected from the data corpus in which participants described or undertook security activity in the workplace. Building upon impressions formed in phase 2 activities of the inner work life [4], individual statements about the described events were examined for positive and negative perceptions that have been shown to affect satisfaction in the software engineering workplace [24]. The context surrounding the event and relations to work factor perceptions were noted. Statements were also identified in which the participant commented or reflected about the perceived significance of the experience to their practice.

Finally, individual instances were compared with one another by the first two authors to compile a set of five distinct responses that reflect characteristics of "security response" within software development. The set of responses is displayed in Table 3, that are more fully explained in Section 5.

4 FINDINGS

As explained in the prior section, the analytic process in this study was pragmatic, drawing different sources of data together for description, comparison, and interpretation [29]. In this section, a descriptive account of the fieldwork is given [5]. First, in answer to RQ1, an overview of organizational security for Sites A and B is provided,, followed by a description of security work episodes. In Section 4.2, and supporting RQ2, subsections describe how developers personally engaged with security at both sites. Taken together, the

Table 3. Response synthesis

Statement	+/- Work Factors	Context	Reflection	Response
“It was in front of me, it was available, and I started reading and started saying suggestions on how to get around [problems].”	+ creativity	Org – introduced a vulnerability scanner	"After [that task], I used OWASP a lot. Now I understand that the second there’s a slight hole it’s a big deal."	Explore
“The objective from the outset was a high level of security. We began with the architecture. Previous projects got stymied by rules and bureaucracy and we thought this time we won’t do it that way.”	+ good relationship with users	Client – flexible, open to new solutions, increase in budget.	“The heartbleed issue was a vindication for us. Though the system included TLS, we were able to say no action needs to be taken, our line of defense is completely adequate.”	Direct
“They can’t upgrade it because it would break half of the projects on there at least. And they have no way to check or to audit.”	- poor environment	Parent Org - legacy environment	“I know because I was trying to get something to work for XXXXX. If I have the ability to commit code on one project, I can probably break all the others.”	Worry
“. . . Run SSH gen to make sure you have the right folder. . . And then copy to the right folder.”	+ good relationship w/ colleagues	Team – communication between teams encouraged	"I quite like the ’someone’s got a problem’ and helping them resolve that problem so they can carry on with whatever they are doing."	Guide, Follow

sections describe ways in which developers at Sites A and B interact with and contribute to security activity within their environments, and provide an empirical foundation for the framework of security responses presented in section 5.

Interviews taken with security professionals and managers at both sites provide contextual depth for each of the sections. However, the descriptions reflect the perspective and perceptions of the developers from each site who informed the research (see Table 2). Text within quotes has been edited to remove repetition, para-linguistic utterances, and company specific information. Omissions are indicated with ellipses in square brackets “[. . .]”. Where necessary, words are supplied. Supplied words appear in square brackets, for example “[supplied]”.

4.1 RQ1. Where can security be found in “ordinary” software development environments?

Sites A and B were each acquired by larger companies. Site A was an independent company until mid 2016, when it joined the parent company based in the US. Site B was acquired in November, 2016, the year before the study began in 2017. In both cases, the parent companies introduced a stronger organizational emphasis on security and brought with them security expertise and knowledge. The effect of the acquisition was perceived differently by participants at the two sites, as described below.

Since the acquisition, Site A has an increasing "will" to improve security within the engineering department. This change was associated by participants with a growth in the business over the years from a small operation with a small technical team that "organically grew" software to meet the needs of a few customers. As it has grown, software practices have had to be shored up and security issues in legacy code have needed to be addressed. However, the parent company also has a specialist security function and a Chief Information Security and Privacy Officer (CISPO) who is responsible for protecting both company and customer data, and making sure appropriate security controls and auditing are in place. Security roles within software development in the parent company were expanding during the study, and increased auditing reports and security training were being rolled out globally.

Site B's journey has been different. In the past, security processes were perceived to be "simpler" and security was built in from the beginning by the engineering team. Engineers used technical spikes and prototypes to help identify risks and tease out uncertainties with clients. By contrast, the acquisition brought with it a dedicated security team. More people are involved now and additional layers of approval are required for choices made in software design and implementation. Detailed written security specifications are produced for each project and engineers have the sense they have less freedom to find solutions. Perhaps as a consequence, developers at Site B make a distinction between complying with security policy and engineering a secure system, explaining "[Now] you're thinking about compliance, whereas [in the past] you're thinking about practically how you develop a secure system".

4.1.1 At the Team Level. This acquisitions at each site provide insight into the different ways that security handling within teams changed as a result of organizational shifts. Changes brought in by the larger company were perceived to have an effect within teams at both sites in physical and network security, and in security policies, described below.

Physical Security The engineers at Site A were aware that new security measures had been put in place to protect the company. One participant explained that measures are taken in code to keep external attackers out, but can also be used by companies to slow down, or counter internal threats from employees. This, they explained had been done at Site A through changes to the physical environment. For example, access to the building and to individual rooms was now controlled through keycards, a change that another participant noted "gets in the way" but was not reported to create problems.

Similarly, at the end of 2018, the engineers at Site B moved locations, leaving their own set of offices and joining a corporate office in a different part of London. The physical environment changed immensely in the process, going from an open plan arrangement accessible by a buzzer and a key card, to offices with several layers of access control applied to each floor in the building. External visitors were permitted to access assigned meeting rooms on a single floor, and were required to register with staff upon entrance to the building.

Network Security At Site A, access to production machines was taken away from developers. At the time of the study, engineers were required to ask the lead engineer on their team to let them into the production environment to complete some tasks. Though one team lead found these interruptions to be "annoying", they understood why the change in responsibility was necessary. Security measures were also added to control access to machines on the networks. As one engineer put it, "it is all the same kit, but with an extra layer of protection around it". Both changes were described by multiple participants as positive, in that the new controls countered the risk of malicious activity.

The technical environment underwent changes at Site B, too. For example, requirements from the parent company's information security team regarding vulnerabilities in the packages that they use introduced new audit tools; and new security access controls were placed on deployment and development environments. These controls and processes were reported to cause frustration and workarounds that worried developers. As one participant explained, "If you're on the developer list, you have access to every single workbench, including a whole bunch of projects [we are] not meant to know about".

Security Policy Security policies were differently formalised within the development lifecycle at the two sites. The work of the security team in the parent company at Site B ran parallel to the work of the software engineers, so although a specification might say "This has got to be highly secure", non-functional requirements might not be discussed for many months. In response to this, the software teams had tried adding audits into the integration process to protect the systems from malicious insiders, explaining, "all code commits were associated with a developer so we had control and audit of what was going on through code submission, review and release". But further down the line it slowed down progress, and so was abandoned.

By contrast, the teams at Site A had only recently begun to write security-related coding standards, initiated by the principal software engineer. The engineering teams also took steps to integrate security in the development lifecycle, running automated tests each week against the software to identify issues that appear in the OWASP top ten list. Depending on the severity of issues reported in this tool, the technical product owner responsible for security and privacy generated technical stories that were either immediately actioned or added into the relevant backlog.

4.1.2 Within Tasks. Changes in organisational security brought by acquisitions at both sites provide useful context to everyday software development tasks observed and reported at both sites. Organised within a set of 10 episodes (see also Appendix A.2), three tasks in this set were observed as they occurred, including a task in which a pair of developers set up a virtual work environment (EP1), a second in which a developer interacted with the software repository (EP8), and one in which a pair implemented code to meet regulatory requirements for the European Union General Data Protection Regulation (GDPR) (EP7). The other episodes were reported, drawn from interview and workshop data.

Work episodes had different durations and scales. One of the observed episodes comprised only a minute or two (EP8), one lasted for around 15 minutes (EP1), and the third comprised around three hours (EP9). Of the remaining episodes, seven developers recounted current or recent encounters with security in the contexts and timeframes of projects. One developer recounted an experience with security in a project that he had led for the company and had ended more than a year before the interview (EP5).

Tasks involved a range of activities including architecture and design (EP5), the use of tools during code implementation, and code integration or administrative work within and between environments on the technical network (EP10). Task completion required developers to employ or identify a range of different kinds of security information [63], including knowledge about how to prevent particular attacks, approaches and fixes as they related to the code and functionality, network security, application context and end-user interactions, and the ability to understand and interact with tools that included security mechanisms.

Security activity was not always apparent, even within tasks that included explicit security or privacy implications. For example, in EP7, two developers were observed for three hours while working together to add logging features to meet requirements for the GDPR. The developers involved did not at any point mention aspects of the work that could be associated with security or privacy. Similarly, interactions with security processes and teams were reported in another case to run parallel to the SDLC, resulting in development outputs that failed security audits late in the process (EP6). This indicates that the requirement for developers to engage with security varies by task. One participant from Site B explained this difference in terms of "activeness":

> "When I'm thinking about security, these are the things I'm usually thinking about. So, authentication, authorisation, encryption. A general hardening of say operating systems, doing something with the firewall settings, or something that I'm actually thinking about security. Whereas in a lot of the main part of security, I may be thinking '*How do I make sure that I've properly handled the various use cases?*' I'm not thinking about the security aspect of that. It may increase the security, but I'm not thinking of it from that perspective. I'm thinking, '*Let's make sure that the client doesn't crash because the users will be unhappy*' [. . .] There's also the third category [. . .] everything from the source code repository we are using to password management [. . .] and issues around the services and environment and which bits of them I have choices over."

4.2 RQ2. How do non-specialist developers engage with security in practice?

The prior section descripted aspects of security within the organizations at Site A and B and as security was observed and reported to be a part of day-to-day software development tasks. Within episodes, each participant had access to sensitive information and information systems within their organization or in client sites. However, though many participants demonstrated a degree of awareness of common vulnerabilities and indicated security training or education, only a few participants reported that security was a prominent aspect of their day-to-day work. This section describes how participants engaged with security at both sites, as a part of their professional development, interest in and curiousity about technology, and need for social connection.

4.2.1 Desire to Produce Good Software. Among participants, the desire to produce good software was associated with security in two ways: as both a duty to protect and in terms of the personal ability to prioritise and drive security activity. Both were characterised by participants using positive and negative experiences.

A duty to protect. At Site A, developers associated writing good software with a need to protect assets. This was described in terms of protecting data for people, and of protecting the company from external threats. These aims sometimes intertwined. P8 described a near-miss in which an area of the code was found during testing that would allow client data to be compromised through the browser. In this case, the problem wasn't part of the sprint, but the team pushed for extra time to address the security concerns. In a separate interview, P11 described the same incident in this way: "It was a real shocker when we found it. We happened to be doing some work in a related area and I thought 'What happens if I do this?' and then found that [vulnerability] and thought ‘Oh, this is dreadful’, and we completely circumvented our process at that point because we felt the situation justified it. It was a very quick fix and really straightforward.”

However, developers reported that they were not always able to drive security activity within the company. P9 explained that making the case for bigger security initiatives was difficult with senior managers in the organization, noting that one tactic they used was to entice managers with examples about how security was done at “big” companies.

An ability to prioritise. The ability to direct and drive security activity was connected to good software at Site B. P14 gave an account of a positive experience in which they were given the freedom and flexibility to engineer a bespoke system for a company in the financial sector. P14 tied the freedom he was given to build security in from the very beginning to the success of the solution, which was described as optimal, and completely secure. This was contrasted by the participant with more recent projects in which clients are not as open to newly engineered security solutions. P14 has the impression that newer clients are more interested in ensuring that software is secure through the use of existing third-party products.

At Site B, P15 described working on a year-long, greenfield development project for a client. The team working on the project is a blend of employees from the client, and consultants and engineers from Site B. Though security was talked about by the “higher ups”, and was said to be important, there was no follow through. Work on features seemed to be more important. A colleague working on a different project described a “sense of unease” in knowing how hard to push back on projects where he could see that more could be done, explaining it as a “mismatch” between his priorities as an engineer and the reality of how projects are run.

4.2.2 Having a Keen Interest. Engaging with security out of personal interest was reported in terms of developing particular skills or techniques as a part of project work, and also as an individual topic of interest or curiosity.

Security skills were gained as a part of duties. In the course of completing his successful project, P14 needed to develop a high degree of skill and knowledge about cryptography, an interest that is still cultivated on the side. Similarly, P8 explained that they didn’t feel that they had been “picked to be the security guy” at a prior company, but the team was tasked with running a scanner to identify vulnerabilities and “it was in front of me, it was available.” P8 took an active role, giving suggestions about how to resolve vulnerabilities. This was also true of other participants. P9 had active role in promoting security at Site A, driving technology changes in the architecture to upgrade the language used in a core product and improving the security of data at rest.

Security was explored for its own sake. Though several participants mentioned an interest sparked by coursework at university, or simple hacking attempts that were treated as a game or puzzle, self-study was the primary path into security. As P9 explained, “Most of this was on my own, reading stuff and reading other people’s pitfalls. I realised that I knew an awful lot about this. I don’t know when that happened though, I couldn’t pinpoint it [in time].” Not every participant found security to be accessible or easy to learn. In one case a participant explained that they might like to learn more about security but found it difficult to get past the “basics”, in part because it was not about building some thing “tangible”, but rather about building something “proper”.

Interestingly, having a sense of duty or interest **were not guarantors of active engagement with security**. This appeared to be connected to circumstances within the work environment. During this study, P9 was actively pushing forward security initiatives as a part of their role at site A. However, though P14 is still interested in cryptography, the project is over. On newer projects, they have less influence with clients over engineering decisions that could impact security and so they are considering taking a less active role in leading projects with a clear security focus and getting into different technologies, like Rust. Similarly, P8 indicated they had developed a high degree of explicit knowledge about security in past employment. However, when asked whether they needed to do more about security at this company, they explained that a recent promotion left them with less time. As a consequence:

> "I am not looking for these holes or problems. I'm not invested in trying to see if I can find a better solution. I'm happy to implement them when they're found or I know about them."

4.2.3 Feeling Secure in One Another. Developers reported engaging with security out of interest and to meet professional demands, but also as a way to feel secure as professionals. The sense of security was fostered while working through situations in three ways: through informal learning, by establishing or exercising values like trust, reputation, and belonging, and through assessments of their own standing as professionals within their teams and companies.

Security was a collective responsiblity. One participant at Site A was observed to pass tips about security hygiene to employees outside the engineering team. Code reviews were reported as one place where security implications were traded and discussed. As might be expected, P9 is respected in the department team for their knowledge about security, and team members mentioned more than once that P9 was the best person to speak to about security within the engineering department. However, the developers within the teams were also dependent on one another, reporting "seeing things in the code" written by colleagues that showed them how to do security and feeling secure in the choices taken by others in the company.

Developers informally learned about security through social interactions. Though P14 displayed detailed knowledge about cryptography in our meetings, they explained to us that their knowledge was fostered by a team member with experience using cryptographic techniques to secure data, a relationship that was reported as enjoyable and significant. Similar relationships were observed at Site A. In the exchange excerpted in Figure 4, P1 had trouble in setting up SSH keys on a new virtual environment. When asked about the session P1 noted that the time the session took was about lack of experience, explaining that the issue might not have come up or taken so long for a more experienced developer. However, P2 recognised what was going on with the SSH key, but didn't remember exactly how to fix it and did not self-identify as an expert. P2 saw their role as stepping in to work through the problem alongside P1, offering advice and support.

Security was upheld through trust and inclusion. Even as the acquiring company at Site A is placing more security controls on environments to protect themselves from internal threats, developers on one team worked extensively with data created by customers who use the software produced at the company. The data sets can be so large that it isn't practical or desirable to port and test the datastores in local environments at Site A. Instead, the developers were given direct access to production environments in the customer's server space. When asked during an interview if the clients believed that the developers at Site A were honest, P3 commented that they were not sure if this had ever even been considered. P3 noted that the

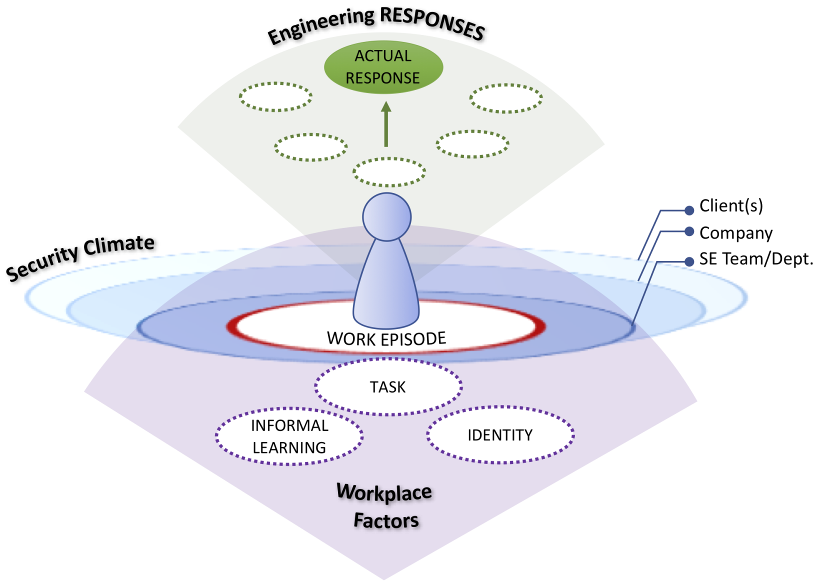


Fig. 5. Security responses in software development: in a given work episode, the response given by an individual is taken from a set of possible responses; the response is shaped by workplace factors [24] and is influenced by the surrounding security climate [28].

developers on the team are "trusted to do the right thing" with the access they have been granted. Security measures and attitudes of clients at site B were less trustful. Engineers reported needing to undergo multiple background checks, a process that made it complicated to enter client sites and to "get going" with projects. Other participants reported that they felt they missed out on information sharing at client sites because they were outside the corporate fold.

5 SECURITY RESPONSES IN SOFTWARE DEVELOPMENT

The findings in Sections 4.1–4.2 describe ways that security were found at the team level and in individual tasks at Sites A and B, and detailed developer engagement with security as a part of meeting individual professional aims and social needs.

Building upon this account, this section outlines a set of inferences that relate key attributes of security that were observed and reported in the collected data to findings in related areas of research. The inferences culminate in a set of responses that typify professional developer behaviour and can be used by teams as a lens on local attitudes and work practices around security. Figure 5 indicates how these aspects relate to each other; these concepts are further described in the following sections.

5.1 Security Climate

Security activity takes place within an organizational climate that reflects shared perceptions around security policy and the assurance measures implemented or adhered to by companies [28]. Within software development, the climate is comprised of the organizational environments with which the developer interacts as a part of their employment. This climate includes teams, departments, the broader organization and customers or clients; security policies in any of these environments; and the technical infrastructure and associated security measures embedded within it: libraries, tools (including database management software) and hardware.

Findings in Section 4 show that security is linked to the kind of work required to complete the task, but also to the priority given to security by organizations and clients. As with privacy, the approach toward security held by a company can constrain or influence the perception developers have to prioritise and direct activity [11]. Three inferences can be made about how the security behaviour of developers in similar environments may intersect with organizational expectations for compliance and priority:

(1) **As users, developers must comply with security measures that are in tools.** In these cases, compliance is necessary: the security policies are mechanized, and the controls— often over access— act as forcing functions [49] that stop forward progress. Developers interact with security policies through measures put in place by the company to control their actions. Behaving securely is tangential to the task at hand, but the need to comply is high. Participants in this study were observed to get past security barriers as quickly as possible so that they could move on with the task at hand.

(2) **Within technical systems, developers are expected to comply with security measures** as they integrate the software they write with the technical infrastructures of their companies and clients. In integration tasks, the primary security mechanism employed is access control. In this study, the need to gain access to a system by participants was often tangential to the development task and shadow tactics were reported, suggesting an imbalance between the drive to be productive and the priority to behave securely [45].

(3) **As engineers, developers are entrusted to enact security policies for their companies and clients** through design and architectural decisions and in subsequent implementation and deployment activities. In these cases, security is a primary concern within the development task, reported by participants as "active". Findings included examples in which participants played a dominant role in setting project directions around security when organizations and clients were supportive.

5.2 Workplace Factors

A work episode refers to a developer's workplace and to events that come up in it [24]. In this context, episodes have a security aspect and entail common software engineering tasks within development lifecycles and technical environments. The security aspect is any part of a work episode that involves efforts to make or keep a system dependable in the face of malicious activity [6]. Such activity was associated by participants at Sites A and B with professional standards for good practice. Reported and observed instances included evidence of informal learning through localised problem solving [20]. In solving technical problems, participants also reported that they gained awareness of acceptable security behavior within the security climate. Examples of workplace factors are given in Table 4.

5.3 Responses

Responses are given by developers within work episodes that have a security aspect. Directed toward the task at hand, a response is an analytic term describing practice that reflects an interplay between personal factors and the broader organizational climate, depicted in Figure 5. Applied to examine work episodes that have a

Table 4. Workplace Factors

Dimension	Description	Examples
Task	The kinds of tasks in which security activity is needed. Roles within tasks may include use or administration [45], and instances in which developers are expected to enforce or enact security policies [6] for their companies and clients.	• Using tools, APIs or application frameworks • Altering configurations or accessing multiple technical environments • Making design or architectural decisions, writing and testing code
Informal Learning	The professional competence a developer exhibits and extends through problem solving within a specific situation [20] that can support task completion and improve individual knowledge [72].	• Online information sources • Training and education • Support from other developers during pairing, code reviews, informal interactions "between desks"
Identity	The professional standard to write "good" software and other personal aims and interests [57] that can direct security actvity within work episodes	• Creativity - encompasses the need for growth, variety and technical challenge. • Professional standards - the desire to produce good software; includes seeing tasks through to completion. • Social connection – including trust, interdependence, respect and responsibility.

security aspect, teams and companies can use the response types as a lens to examine local attitudes and work practices around security. In recognising that a range of responses exist and by identifying contributing factors in their own environments, it is possible to adjust security activity.

Following from the findings given in Section 4, several inferences can be made about security responses as a represented behavior within software development:

(1) Responses reflect personal professional values and needs [17] and are visible at the team level within informal collaboration and learning [20].
(2) Developers possess a set of possible responses that are assembled, refined and applied through multiple experiences orrywithin their career [55], i.e. within different projects, roles and employment.
(3) Security responses are selected by developers to meet needs in particular situations. This statement includes two implications:
   (a) the priorities of project stakeholders within the broader security climate subsume those of the developer, and
   (b) the priorities for task completion influence developer behavior

As the example in Box 5.3 shows, factors that combine to produce a response are nuanced and context dependent. The response typology is neutral and faceted: similar factors can be associated with any of the responses in a given situation. As a consequence, implications for stakeholders can be negative or positive. For example, meeting production targets is a condition that was reported by participants to produce the worry response. In Box 5.3, worry is a signal that highlights security as a priority to the developer's team. The recognition of the response in this case produces corresponding activities within the team intended to improve security in the system.

To guide use of the response typology, the full set of empirically-informed responses, contributory factors and implications for organisations, teams and individuals is provided in Table 5.

Box 5.3: Worry

An engineer is aware that a design choice has poor implications for security. They raise this issue in a standup meeting, but the team is under pressure to meet production targets and the group decides to add the problem to a technical user story in the backlog. **In this case, task completion is of higher priority to the organization than security, subsuming the aim of the engineer to produce code to a high standard.** The engineer implements the vulnerable design, but **worries** about the quality of the product. The team picks up on their teammate's discomfort, and **pushes back** with the product owner in the next sprint planning meeting to make sure that the technical story is scheduled in.

6 LIMITATIONS

Ethnographies seek to capture, interpret and explain social life within groups, among individuals, and within communities [52]. By examining security activity at two field sites, this study addressed constraints on access [19] and the problem of ecological validity within developer-centred security studies [36]. The report presents a situated representation of security activity that reflects the environments and periods during which data were collected. As such, the findings cannot be applied to explain security behaviour in any small or medium sized enterprise (SME). It is not possible to state that because multiple instances were observed in which "following" was the security response, that professional developers only or primarily exhibit following behaviour. However, it is possible to infer that professional developers are likely to follow in certain circumstances, as described in Section 4, explained in Section 5, and exemplified in Table 5. This research argues that responses like "following" are a distinct entity within software engineering, formed out of an interplay of individual characteristics, work factors, and environmental circumstances. This claim is logically composed [61] and could be tested in future work to examine practice in other environments. As explained below, the research underpinning the claim is also trustworthy [52].

Descriptive accuracy and NDA compliance were ensured through peer debriefing and member checking [52] at several points. **1)** The response schema was presented to the full research group at a meeting in November 2019. **2)** At both sites, researchers restated and summarised information provided by each participant or asked contextual questions to clarify and correct understanding. These activities mitigated the risk that the data collected were misunderstood or misinterpreted [53], and clarified that the interview questions and

Table 5. Table of Observed Responses, Contributory Factors and Potential Implications

Response Given	**Observed Factors**	**Potential Implications (for Indiv.,Team,Org.)**
Follow: A person that follows examples & prompts from colleagues, or the policies & norms set forth by a client/org.	**Priority:** security may be a primary or secondary concern. **Task** use, administration or implementation **Identity:** good relationship with colleagues (trust), learning & growth **Climate:** the org/client is active with security	- may involve shadow tactics (I) - may diminish learning or initiative (I,T) + can promote awareness, knowledge (I,T) + can strengthen culture (T)
Guide: A person that provides support for localised problem solving for a range of technical topics, including security.	**Priority:** security may be a primary or secondary concern **Task:** use, administration or implementation **Identity:** good relationship with colleagues (helping behaviour); learning & growth **Climate:** the environment supports collaboration	- security approach may be/remain adhoc (O,T) + can promote awareness, professional competence (I) + can strengthen culture (T)
Explore: A person that is engaged & may take an active role in organizational security initiatives.	**Priority:** security is a primary concern for the developer **Task:** use, administration, design or implementation **Identity:** learning/growth, creativity **Climate:** the org/client exhibits a commitment to security	- dependence on "champions" (T,O) - fluctuating levels of engagement/expertise (T,O) + can promote knowledge, career development (I) + can strengthen culture (T,O)
Direct: A person that is engaged, active in security initiatives &very capable.	**Priority:** primary for developer, may be primary or secondary for org/climate **Task:** designing or architecting software **Identity:** strong need for growth or technical challenge **Climate:** security commitment, good relationship w/employees (trust)	- dependent on concern alignment (O, I) - dependent on strong engineering culture (T,O) + can produce developers with a security specialism or near-specialism (I) + can improve security outcomes in code (O,T,I)
Worry: A person that is engaged with security but does not perceive or have an ability to act.	**Priority:** mismatch – primary for the developer, secondary for org/client **Task:** all tasks **Identity:** professional sense of responsibility or duty of care to keep systems secure **Climate:** may not prioritise security or reward developer initiative	- may accompany shadow tactics (I,T) - may indicate poor working conditions or poor relationship with clients (O) + may promote compensatory secure behaviour or pushback (T,I) + may provide a project health indicator (O,T)

session prompts were accessible to the interviewees. **3)** Gatekeepers at each field site were asked to review and authorize the organizational descriptions given in Sections 3 and 2.2, and were provided with a copy of the complete manuscript submitted for review.

Credibility and authenticity of the interpretations were established in three ways. 1) Preliminary findings were shared with participants at site A in a session held in November 2018 during which a brief of the full research project was presented and participants were asked to reflect and openly comment as a group about security as a part of daily practice at Site A. **2)** In meetings and informal conversations, the researchers were able to compare preliminary interpretations formed at Site A with information gathered through interactions held at Site B. **3)** The multi-sited approach ensured access to multiple security events and situations within professional development activity, countering the risk that time spent in the field would lack depth [29].

The soundness of the response framework was assessed using the following techniques. 1) The feasability of the situated nature of responses was established through examination of related literatures and was further assessed in analysis of interview data collected at Site A. In these data, participants' perceptions about security events could be compared at different points in time. **2)** Analytically compiled responses were assessed in an online questionnaire. The questions were designed to evaluate construct descriptions [22] and to affirm the claim that individuals have a set of potential responses that can be applied in different situations. The questionnaire was deployed to participants at Sites A and B and was employed as a source for member checking with developers at additional companies in India and Brazil. Every respondent indicated that multiple responses described them – no participant reported that none of the responses described them and no participant selected a single response.**3)** Information collected using different methods within the two field sites were compared with each other [52] and with information about security practice collected online in Stack Overflow, as described in Section 3.3.2.

7 DISCUSSION

Security in software development is complex and multivalent. In a range of roles and within different tasks, this research has shown that developers are not only concerned with technical aspects of keeping software secure. Like workers in other fields [10], participants in this study promoted and enacted security policy, but also discussed, interpreted and questioned the decisions made within their companies and by clients. The notion that workers respond to events in the workplace has been recognized within management science [4], as an aspect of developer practice [24] and within privacy research [11] in software engineering. The ethnographic account given in this report demonstrates that responses are a represented behavior in software development activity that includes security.

Section 5 argues that security responses arise out of an interplay between personal factors, task requirements and the broader organizational environment, a connection also proposed by Mokhberi et al. [38]. Through a systematic literature review, Mokhberi et al. outlined challenges to writing secure code and tracked dependencies between human, technological and organizational factors. Organised within a conceptual framework, the authors distinguished the organizational dimension, finding that factors such as the manager mindset, attention to requirements and policies, and culture were found to have the greatest influence on factors within other dimensions, including the efforts and capabilities of developers.

A recent qualitative analysis that used narrative and situation analyses to characterise the organisational security landscape in terms of "social worlds" [66] observed that the stakes for security are different for participants of different arenas. While the reputational stakes were higher for organisations, arenas within the engineering fold, including those of developers, project management, and security teams were more directly involved in setting security priorities.

Evidence of organizational influence in the findings presented in this research had negative and positive aspects. Positively, participants held the perception that developer responsibility for security was shared by organizations and their clients. The controls applied to technical systems provided assurance to engineers and were recognized as necessary and beneficial. The organizational circumstances documented in Section 4.1 suggest that the measures taken by organisations freed developers from the need to consider security. Though other studies examining security within the development lifecycle have interpreted similar behaviour as one of taking secuity for granted [9], the broader contextual circumstances observed at the two field sites suggest that organizational measures instead allowed participants to focus on their primary responsibility of implementing and maintaining features in the code.

On the negative side, the priority placed on security within the broader organization and by clients affected the perceived ability participants had to drive security activity. Findings reported in Section 4 and outlined in Section 5 put the notion of security as a secondary concern [1] into perspective, showing a connection between the priority of security within the broader climate, and the perceptions developers had that they could direct security activity within projects. As has been shown in privacy studies [11], in projects in which security was a secondary concern to stakeholders outside the engineering fold, the ability for developers to initiate security activity was dampened. In contrast, when security was a primary concern within the broader climate, participants reported that they felt empowered to influence security outcomes in engineering activities that included design and architecture or that they were free to explore security topics and integrate discoveries within the workflow. This finding held regardless of reported levels of skill or conviction, suggesting that the security positions of organisations may have more weight in promoting security activity than does championing by individuals.

Nonetheless, conviction among participants was associated by participants with professional integrity — a "duty of care" – and expressed in terms of the "golden rule" to manage others' information as they would want their own to be managed [47]. Belying the notion that security is considered to be "burdensome" [9], the sense of duty to users held by participants at both sites in this study was coupled with loyalty toward the company, indicating a willingness and commitment to "good" security.

The findings illuminate additional details of security knowledge and experience in developers. Recent experimental work has found that developers do not unconsciously behave securely during programming tasks, even if they have knowledge about security and an awareness of its importance [43] and that developers working in different contexts will only address security when explicity prompted [40]. Though the participants at Sites A and B possessed a degree of awareness and declarative knowledge about how to recognise and address common vulnerabilities, none described themselves as security experts, underscoring the sense that many developers are "ordinary insiders" [47]. Though knowledgeable and aware about security, they lacked specialism. However, observational data at both sites included evidence that knowledge and engagement were supported and fostered informally among peers, suggesting that security expertise in software development is intertwined with and embedded within informal networks [73], a finding reported in [33], and also described in the context of situated learning [67] .

Observed social interactions included knowledge exchange about technical aspects of security, but not always. In some cases, technical security was an implicit feature of work and entirely absent from discussion. Instead, reflections by participants indicated that interactions fostered feelings of responsibility, trust and connection, suggesting that they produced a sense of security [17]. Furthermore, at times such interactions accompanied behaviors that were unequivocally insecure. Though these two findings appear at odds, they drive home the point that security knowledge and awareness of software developers cannot be assessed solely through measures of technical correctness. Maintaining strong professional relationships can matter more than technical expertise or behaving securely.

This research points toward future work in several areas. As a muli-sited ethnography, the field sites visited in this study provided multiple points of access to developers' perceptions of security in the workplace. The groups at each site were of a similar size and had established traditions and practices. Both sites had also recently been acquired, a serendipity that provided a lens on changes in organizational perspective and practice. However, constraints on access made it difficult to follow up with either site in the longer term. Future work could examine how attitudes and work practices as an aspect of organizational security maturity.

In addition, the data collection methods precluded detailed study of the direct influence factors within responses have on software artifacts. For example, though participants described having a keen interest in finding out about security topics for their own sake — a finding that resonates with reported correlations between inquisitiveness and tool adoption [74], and continuous learning [72] — more research is needed to systematically trace observational findings into the code.

Finally, the particular aim in this research was to enlarge understanding of security in "ordinary" software development as a way to support engineers who are not security specialists. However, documenting what is known about software development practices in security-critical software would also provide perspective on professional software security attitudes and practices. It is possible that the climate within security-critical environments is somehow better, and that developers always or generally feel empowered to direct security activity. The findings presented in Section 4 and the framework presented in Section 5 lay the basis for future work to test the claim made that responses are a distinct entity within software engineering that includes security aspects, and to explore how attitudes and work practices in different kinds of development environments align — or deviate — from one another.

8 CONCLUSION

In the workplace, developers engage with security in a number of ways. Security is a part of activities in which developers are primarily users or administrators of systems that must remain secure. Developers also take an active role in interpreting and enacting security policies for their companies and clients. This research has identified that a developer responds to the security needs in these situations within common dimensions of development practice. The response is influenced by the task that must be completed, local problems that need to be solved, and a developer's individual orientation toward the situation. Findings indicate that a developer may have a preferred approach to solving security problems based on existing knowledge or past experiences, but will tailor their response to meet the demands of companies and clients.

Developers comply with security policies because they perceive value in them. Compliance facilitates task completion, is a way to cultivate interests, and "makes sense" — developers see the point in following policies set out by a company. They associate being secure with core values in the software engineering profession to write "good software" and feel a sense of responsibility to carefully manage user data. However, this research has shown that developers participate in a broader security climate made up of teams, departments, companies, and clients. Decisions that have an impact on security within code are not always made by developers or development teams but instead reflect the attitudes and priorities of companies and their clients. This has an effect on how developers engage with security in practice, resulting in a range of behaviours and activities that maintain socio-technical security alongside production. In recognising the ways in which socio-technical security is maintained through behavioral responses, managers and teams can alter security outcomes in their environments.

ACKNOWLEDGMENTS

The authors thank the professional developers and organizations that gave their time to this study and to the reviewers, including James Noble, for their generous commentary. This work greatly benefitted from the expertise and insight given by Charles Weir and Hannah Cooper, and was supported by the UK NCSC, UKRI/EPSRC (EP/R013144/1, EP/T017465/1), and SFI (13/RC/2094_P2).

REFERENCES

[1] Yasemin Acar, Sascha Fahl, and Michelle L. Mazurek. 2016. You are not your developer, either: A research agenda for usable security and privacy research beyond end users. In *Cybersecurity Development (SecDev), IEEE*. IEEE, 3–8.

[2] Yasemin Acar, Christian Stransky, Dominik Wermke, Charles Weir, Michelle L. Mazurek, and Sascha Fahl. 2017. Developers Need Support, Too: A Survey of Security Advice for Software Developers. In *Cybersecurity Development (SecDev), 2017 IEEE*. IEEE, 22–26.

[3] Anne Adams and Martina Angela Sasse. 1999. Users are not the enemy. *Commun. ACM* 42, 12 (1999), 40–46.

[4] Teresa M. Amabile and Steven J. Kramer. 2007. Inner work life: understanding the subtext of business performance. *Harvard Business Review* 85, 5 (2007), 72–83, 144.

[5] Bob Anderson. 1997. Work , Ethnography and System Design. In *The Encyclopedia of Microcomputers*, A Kent and J G Williams (Eds.). Vol. 20. Marcel Dekker, 159–183.

[6] Ross Anderson. 2008. *Security engineering.* John Wiley & Sons.

[7] Ross Anderson. 2020. *Security engineering: a guide to building dependable distributed systems.* John Wiley & Sons.

[8] Michael Angrosino. 2007. Analyzing Ethnographic Data. In *Doing Ethnographic and Observational Research*. SAGE Publications Ltd, 67–76.

[9] Hala Assal and Sonia Chiasson. 2018. Security in the software development lifecycle. In *Fourteenth Symposium on Usable Privacy and Security (${$SOUPS$}$ 2018)*. 281–296.

[10] Ingolf Becker, Simon Parkin, and M Angela Sasse. 2017. Finding security champions in blends of organisational culture. *Proceedings of USEC* 11 (2017).

[11] Kathrin Bednar, Sarah Spiekermann, and Marc Langheinrich. 2019. Engineering Privacy by Design: Are engineers ready to live up to the challenge? *The Information Society* 35, 3 (2019), 122–142. Publisher: Taylor & Francis.

[12] Hal Berghel. 2017-12. Equifax and the Latest Round of Identity Theft Roulette. 50, 12 (2017-12), 72–76.

[13] Marcus Beyer, Sarah Ahmed, Katja Doerlemann, Simon Arnell, Simon Parkin, M. A. Sasse, and Neil Passingham. 2015. Awareness is only the first step. *A framework for progressive engagement of staff in cyber security, Hewlett Packard, Business white paper* (2015).

[14] Andrea J Bingham. 2021. How distributed leadership facilitates technology integration: A case study of "pilot teachers". *Teachers College Record* 123, 7 (2021), 1–34.

[15] CISA. 2021. *National Cyber Awareness System: Weekly Bulletins.* Cybersecurity & Infrastructure Security Agency. https://us-cert.cisa.gov/ncas/bulletins

[16] Lizzie Coles-Kemp. 2018. *Practising Creative Securities.* Royal Hall University of London.

[17] Lizzie Coles-Kemp and René Rydhof Hansen. 2017. Walking the Line: The Everyday Security Ties that Bind. In *Human Aspects of Information Security, Privacy and Trust (Lecture Notes in Computer Science)*, Theo Tryfonas (Ed.). Springer International Publishing, 464–480.

[18] Pieter Danhieux. 2018. *Are Developers Your First Line of Security Risk or Defense?* https://devops.com/are-developers-your-first-line-of-security-risk-or-defense/

[19] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. 2008. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*. Springer, 285–311.

[20] Michael Eraut. 2004. Informal learning in the workplace. *Studies in Continuing Education* 26, 2 (July 2004), 247–273.

[21] Mark-Anthony Falzon. 2016. *Multi-sited ethnography: Theory, praxis and locality in contemporary research*. Routledge.

[22] David M. Fetterman. 2009. *Ethnography: Step-by-step*. Sage Publications.

[23] 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)*. IEEE, 121–136.

[24] César França, F. Da Silva, and Helen Sharp. 2018. Motivation and Satisfaction of Software Engineers. *IEEE Transactions on Software Engineering* (2018).

[25] Steven Furnell and Kerry-Lynn Thomson. 2009. From culture to disobedience: Recognising the varying user acceptance of IT security. 2009, 2 (2009), 5–10. https://doi.org/10.1016/S1361-3723(09)70019-3

[26] Dieter Gollman. 2019. Authentication, Authorisation & Accountability Issue 1.0. In *Cybok: The CyberSecurity Body of Knowledge*. The University of Bristol. https://www.cybok.org/knowledgebase/

[27] Matthew Green and Matthew Smith. 2016. Developers are not the enemy!: The need for usable security apis. *IEEE Security & Privacy* 14, 5 (2016), 40–46.

[28] Irit Hadar, Tomer Hasson, Oshrat Ayalon, Eran Toch, Michael Birnhack, Sofia Sherman, and Arod Balissa. 2018. Privacy by designers: software developers' privacy mindset. *Empirical Software Engineering* 23, 1 (2018), 259–289.

[29] Martyn Hammersley and Paul Atkinson. 2019. *Ethnography: Principles in practice*. Routledge.

[30] Iacovos Kirlappos, Simon Parkin, and M. Angela Sasse. 2014. Learning from Shadow Security: Why understanding non-compliance provides the basis for effective security. In *Workshop on Usable Security*.

[31] Tamara Lopez, Helen Sharp, Thein Tun, Arosha K. Bandara, Mark Levine, and Bashar Nuseibeh. 2019. Hopefully we are mostly secure: views on secure code in professional practice. In *Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE Press, 61–68.

[32] Tamara Lopez, Thein Tun, Arosha Bandara, Levine Mark, Bashar Nuseibeh, and Helen Sharp. 2019. An anatomy of security conversations in Stack Overflow. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 31–40.

[33] Tamara Lopez, Thein T. Tun, Arosha K. Bandara, Mark Levine, Bashar Nuseibeh, and Helen Sharp. 2019. Taking the Middle Path: Learning About Security Through Online Social Interaction. *IEEE Software* 37, 1 (2019), 25–30. Publisher: IEEE.

[34] George E. Marcus. 1995. Ethnography in/of the world system: The emergence of multi-sited ethnography. *Annual review of anthropology* 24, 1 (1995), 95–117. Publisher: Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, CA 94303-0139, USA.

[35] George E Marcus. 2012. Multi-sited ethnography: Five or six things I know about it now. In *Multi-sited ethnography*. Routledge, 24–40.

[36] Michelle L Mazurek and Daniel Votipka. 2020. 4.2 Ecological validity and study design for empirical secure development studies. *Empirical Evaluation of Secure Development Processes* (2020), 12.

[37] Microsoft. [n.d.]. *Microsoft Security Development Lifecycle*. Technical Report. https://www.microsoft.com/en-us/securityengineering/sdl

[38] Azadeh Mokhberi and Konstantin Beznosov. 2021. SoK: Human, Organizational, and Technological Dimensions of Developers' Challenges in Engineering Secure Software. In *European Symposium on Usable Security 2021*. 59–75.

[39] S. Nadi, S. Krüger, M. Mezini, and E. Bodden. 2016. Jumping Through Hoops: Why do Java Developers Struggle with Cryptography APIs?. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. 935–946.

[40] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, Emanuel von Zezschwitz, Matthew Smith, Karoline Busse, Karoline Busse, Dominik Wermke, Sabrina Amft, and Sascha Fahl. 2018. If you want, I can store the encrypted password. A Password-Storage Field Study with Freelance Developers. In *ACM CHI Conference on Human Factors in Computing Systems*. USENIX Association, 311–328.

[41] 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*. 311–328.

[42] Daniela Oliveira, Marissa Rosenthal, Nicole Morin, Kuo-Chuan Yeh, Justin Cappos, and Yanyan Zhuang. 2014. It's the Psychology Stupid: How Heuristics Explain Software Vulnerabilities and How Priming Can Illuminate Developer's Blind Spots. In *Proceedings of the 30th Annual Computer Security Applications Conference* (New York, NY, USA) *(ACSAC '14)*. ACM, 296–305.

[43] Daniela Seabra Oliveira, Tian Lin, Muhammad Sajidur Rahman, Rad Akefirad, Donovan Ellis, Eliany Perez, Rahul Bobhate, Lois A. DeLong, Justin Cappos, and Yuriy Brun. 2018. API Blindspots: Why Experienced Developers Write Vulnerable Code. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. 315–328.

[44] 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)*. 205–220.

[45] Shari Lawrence Pfleeger, M. Angela Sasse, and Adrian Furnham. 2014. From weakest link to security hero: Transforming staff security behavior. *Journal of Homeland Security and Emergency Management* 11, 4 (2014), 489–510.

[46] Andreas Poller, Laura Kocksch, Sven Türpe, Felix Anand Epp, and Katharina Kinder-Kurlanda. 2017. Can Security Become a Routine?: A Study of Organizational Change in an Agile Software Development Group. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (Portland, Oregon, USA) *(CSCW '17)*. ACM, 2489–2503.

[47] Clay Posey, Tom L. Roberts, Paul Benjamin Lowry, and Ross T. Hightower. 2014-07-01. Bridging the divide: A qualitative comparison of information security thought patterns between information security professionals and ordinary organizational insiders. *Information & Management* 51, 5 (2014-07-01), 551–567.

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

[49] James Reason. 1990. *Human error*. Cambridge university press.

[50] Hugh Robinson and Helen Sharp. 2005. Organisational culture and XP: three case studies. In *Agile Development Conference (ADC'05)*. IEEE, 49–58.

[51] Hugh Robinson and Helen Sharp. 2009. The emergence of object-oriented technology: the role of community. *Behaviour & Information Technology* 28, 3 (2009), 211–222. Publisher: Taylor & Francis.

[52] Colin Robson and Kieran McCartan. 2016. *Real world research: a resource for users of social research methods in applied settings*. Wiley.

[53] Per Runeson and Martin Host. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14 (2009), 131.

[54] M Angela Sasse and Awais Rashid. 2019. Human Factors Knowledge Area Issue 1.0. In *Cybok: The CyberSecurity Body of Knowledge*. The University of Bristol. https://www.cybok.org/knowledgebase/

[55] Edgar H. Schein. 1996. Career anchors revisited: Implications for career development in the 21st century. *The Academy of Management Executive* 10, 4 (1996), 80–88.

[56] Bruce Schneier. 2008. *The Security Mindset - Schneier on Security*. https://www.schneier.com/blog/archives/2008/03/the_security_mi_1.html

[57] Helen Sharp, Nathan Baddoo, Sarah Beecham, Tracy Hall, and Hugh Robinson. 2009. Models of motivation in software engineering. *Information and software technology* 51, 1 (2009), 219–233.

[58] H. Sharp, Y. Dittrich, and C. R. B. de Souza. 2016-08. The Role of Ethnographic Studies in Empirical Software Engineering. *IEEE Transactions on Software Engineering* 42, 8 (2016-08), 786–804.

[59] H. Sharp, H. Robinson, and M. Woodman. 2000. Software engineering: community and culture. *IEEE Software* 17, 1 (2000), 40–47.

[60] Adam Shostack. 2014. *Threat modeling: Designing for security*. John Wiley & Sons.

[61] Mario Luis Small. 2009. How many cases do I need?' On science and the logic of case selection in field-based research. *Ethnography* 10, 1 (2009), 5–38.

[62] Graham M. Smith. 2005. Into Cerberus' Lair: Bringing the Idea of Security to Light. *The British Journal of Politics & International Relations* 7, 4 (2005), 485–507.

[63] Justin Smith, Lisa Nguyen Quang Do, and Emerson Murphy-Hill. 2020. Why Can't Johnny Fix Vulnerabilities: A Usability Evaluation of Static Analysis Tools for Security. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. 221–238.

[64] Marius Steffens, Christian Rossow, Martin Johns, and Ben Stock. 2019. Don't Trust The Locals: Investigating the Prevalence of Persistent Client-Side Cross-Site Scripting in the Wild. (2019). Network and Distributed Systems Security (NDSS) Symposium.

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

[66] Inger Anne Tøndel, Daniela Soares Cruzes, and Martin Gilje Jaatun. 2020. Using Situational and Narrative Analysis for Investigating the Messiness of Software Security. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–6.

[67] 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 *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*. 617–632.

[68] Dirk van der Linden, Emma Williams, Joseph Hallett, and Awais Rashid. 2020. The impact of surface features on choice of (in) secure answers by Stackoverflow readers. *IEEE Transactions on Software Engineering* (2020), 1–1. Publisher: IEEE.

[69] Andrew van der Stock, Brian Glas, Neil Smithline, and Torsten Gigler. 2017. *OWASP top 10-2017 the ten most critical web application security risks.* Technical Report. https://owasp.org/www-project-top-ten/2017/

[70] John Van Maanen. 2011. *Tales of the field: On writing ethnography.* University of Chicago Press.

[71] Charles Weir, Ingolf Becker, and James Blair, Lynne. 2021. A Passion for Security: Intervening to Help Software Developers. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice.* IEEE, 21–30.

[72] Charles Weir, Awais Rashid, and James Noble. 2016. How to improve the security skills of mobile app developers? Comparing and contrasting expert views. In *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016).*

[73] Etienne Wenger, Beverly Traynor, and Maarten de Laat. 2011. Promoting and assessing value creation in communities and networks: a conceptual framework. Number 18. Ruud de Moor Centrum, Open University of the Netherlands, 1–56.

[74] Jim Witschey, Shundan Xiao, and Emerson Murphy-Hill. 2014. Technical and personal factors influencing developers' adoption of security tools. In *Proceedings of the 2014 ACM Workshop on Security Information Workers.* 23–26.

[75] 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 & social computing.* ACM, 1095–1106.

A APPENDICES

A.1 Interview Guide

Questions were adapted from [17] and [62].

Now I want to ask a few questions about security within software engineering. There are no right or wrong answers, I am just interested in gathering your impressions.

15. Does security come up often in your work? - What needs to be secured? Or Who?
16. What is doing the securing?
17. Why is [the subject] being secured?
18. Who (or what) is [the subject] being secured from?

Additional questions for those who spend time in security work

19. When did you start spending more time on security issues? - What caused this change?
20. How do you keep up with security technologies?
21. Do you feel you could/want to do more about security? - What kind of support would you need?
22. Do security measures get in your way? What do you do about it?

A.2 Security Work Episodes & Responses

Table 6 highlights a set of work episodes identified in our field studies through observation (O), interviews (I) and at a Modelling Session (MS). Security areas corresponding to the CYBOK taxonomy (https://www.cybok.org/) are noted in the field "Security Area". The other fields indicate the nature of the task; and who or what drives the security behaviour or enacts the policy. The description gives a brief overview of the episode.

Table 6. Security Work Episodes

Ep Site (Source)	Security Area	Task	Enacted by	Description
EP1 Site A (O)	Authorisation	Use	3rd party tool	Two participants configure SSH keys for an instance of Git running in a newly created virtual environment.
EP2 Site A (I)	Authorisation	Admin	Lead Engineer	Access to production servers is restricted to lead engineers. The participant must assess requests to access the production servers and make the connection.
EP3 Site A (I)	Authorisation	Admin	Engineers at the client site	Engineers at the client have given the participant credentials to access to their production servers for testing changes to code.
EP4 Site B (MS)	Authorisation; Hardware	Use	Consultants from different companies	Consultant engineers trade passwords with each other to gain access to password protected devices on the network using email and chat.
EP5 Site B (I)	Software Security; Web & Mobile; Network; Distributed systems; Cryptography; SSDL	Engineer	Lead Engineer	The participant described the details of a past project in which he was able to engineer security into a product from the very beginning.
EP6 Site B (MS)	Risk Anal. & Governance; SSDL	Engineer	Security Team	The security team audited a solution created by a participant and found it did not meet the company's internal security compliance requirements.
EP7 Site A (O)	Law & Regulation; Software Security	Engineer	Pair of engineers	Implementing logging for GDPR. No explicit talk about security implications of the task.
EP8 Site A (O)	Authorisation	Use	3rd party tool	A participant ran into problems when committing generated code to GIT. A second engineer explained that the problem was related to SSH_AUTH
EP9 Site B (I)	SSDL	Engineer	Client	A year-long greenfield development project for a client. Security is said to be important, but client presses for more time to be given to features. It is key that the product launches on time.
EP10 Site B (I)	Network Security	Admin	Engineering Team	A workaround was put in place opening firewall ports to allow code to be propagated from a local development environment to a networked repository.