"Hopefully We Are Mostly Secure": Views on Secure Code in Professional Practice

Tamara Lopez[*], Helen Sharp[*], Thein Tun[*], Arosha K. Bandara[*], Mark Levine[†] and Bashar Nuseibeh[*][‡]
[*]School of Computing & Communications, The Open University, Milton Keynes, UK
[†]Department of Psychology, University of Exeter, Exeter, UK
[‡]Lero - The Irish Software Research Centre, University of Limerick, Limerick, Ireland
Email: [*]firstname.lastname@open.ac.uk, [†]firstname.lastname@exeter.ac.uk, [‡]firstname.lastname@lero.ie

***Abstract*—Security of software systems is of general concern, yet breaches caused by common vulnerabilities still occur. Software developers are routinely called upon to "do more" to address this situation. However there has been little focus on the developers' point of view, and understanding how security features in their day-to-day activities. This paper reports preliminary findings of semi-structured interviews taken during an ethnographic study of professional software developers in one organization who are not security experts. The overall study aims to understand how security features in day-to-day practice, while analysis of the interview data asks whether developers are responsible for security. The study reveals that awareness around security matters is raised through several paths including processes, standards, practices and company training and that a focus on security is driven by contextual factors. Security is taken care of with policies and through safeguards, and is handled differently depending on whether a team is developing new features, and hence "looking forward", or working with existing code and hence "looking back". Developers take and share responsibility for security in the code, but suggest that their responsibility has limits, and relies on collective practice.**

***Index Terms*—secure software development, collaborative environments, empirical studies**

I. Introduction

Are developers responsible for security? Several of the breaches reported in recent years question the role of developers in keeping software systems secure. A look at media sources makes it clear that there are many views about what developers *should* be doing about security. Developers should be security champions, the "first line of defense" [1]; they are increasingly responsible for security, but *shouldn't* be [2]; both development and security departments should take responsibility for application security [3], and at the most extreme, developers should be liable for code with vulnerabilities [4].

This is an amplified portrayal of application security coverage in the media, but it should be noted that the claims made are often grounded in established principles. It is possible to build security in from the start by integrating operations tools and staff with developers and security experts . For example, DevSecOps integrates security controls and processes into the DevOps software development cycle [5]. Like DevOps [6], DevSecOps has a reliance on operational tools, established agile engineering practices and the right culture of collaboration [7].

It is also possible to consider security at all phases of the development process, as in secure-by-design approaches [8], [9]. One objective of these approaches is to ensure that software architectures have appropriate security features, by integrating appropriate elements of best practice that address relevant threats (e.g., OWASP Top 10 List [10]).

The sources touch on another valid point. Many software developers *do not* consistently and comprehensively make use of security practices and technologies, raising questions about what else might be going on. It may be true that "good" security requires effort by developers at every step in the process. However, it may be that security in software development is also driven by intrinsic factors that can be supported through social interactions in the community and culture of software development.

To investigate this possibility, the Motivating Jenny project[1] is conducting a series of ethnographic studies that build upon frameworks of personal motivation and team culture [11]–[13], to understand more about how professional developers can be motivated to adopt security coding practices. This paper gives a preliminary account about the responsibility developers feel toward writing secure software, based on a set of interviews conducted as part of one such ethnographic study.

II. Background

Security has been described as a secondary concern to developers, one that must be prioritized and managed alongside other tasks developers need to complete [14]. However, the stakes in security are high. Security related errors made by developers can be costly, they can endanger all those who rely on the software they build [15]. One problem is that developers rarely need to use security techniques such as cryptography in their software, and when they do, the APIs are difficult to use [16]. Guidance in online sources used by developers is not comprehensive or robust [17], and often not correct [18]. Activities designed to raise awareness are perceived by developers to be helpful, but may not have lasting impact on teams [19].

Within software engineering, security is commonly considered in terms of a particular mindset [20]. Engineers are often encouraged to adopt the attacker mindset, in terms of the ill intent or aims that other people might have toward the system they are building, and how an attacker might try to gain access

[1]https://www.motivatingjenny.org

to the system or what an attacker might do when they have access to it [21]. Security can also be considered using the defender mindset, in terms of knowing weaknesses or flaws in software that might make it easier for attacks to take place. It considers how prevalent weaknesses are, and how easy they will be to find and mitigate [10]. Both approaches include valid and useful techniques, however they place the developer in defensive or offensive roles, positions that do not support this project's aim to understand what developers think about security, about their interest in writing secure code or how important they believe secure code is.

Rather than using negative and confrontational constructs, it is possible to think of security in positive terms [22], to consider security as a quality to be striven for rather than to be feared, a "property of the social world" [23], that reflects shared values about what is important [24].

This approach has corollaries in computing. In the current moment, keeping sight of the underlying ethics and fairness that drive technology initiatives is a primary concern [25]. Efforts are underway to find ways to empirically measure values in software [26]. Value sensitive design methods have arisen to help designers account for and support human values in technology products [27].

The emphasis on values is key. It acknowledges that making technology is a social process [13], and should result in technologies that reflect what people think is important [28]. In the same way, considering security with a value oriented lens may offer developers a way to advocate for what they believe is important in their work, a point that has connections with research in software engineering motivation [29].

III. Case Description

The case focuses on a global software company that specializes in workforce management software. Specifically, this study engaged with one organization within this parent company, based in the South East of England. This organization consists of about 100 people with around 40 of those being formed into software development teams that use Scrum-based Agile practices (a further 16 engineers who work with those based in this office are located overseas). 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 a technical author who are spread across the teams. All teams follow a release cycle of 8 weeks in which 3 sprints of 2 weeks each are focused on the product backlog, and the final sprint (i.e. the last 2 weeks) is mostly devoted to making fixes and regression testing.

The case organization was an independent company until mid 2016, when it joined the parent company based in the US. This parent company has a specialist security function and employs a Chief Information Security and Privacy Officer. In the case organization, there is one principal software engineer who takes a key role in network and architecture security, and one technical product owner, who takes the lead locally on a range of security-related matters, including GDPR. The organization uses a service called Qualys which runs automated tests against the software to identify issues that appear in the OWASP top ten list; this is done weekly. If issues are found, then depending on their severity, technical stories are generated and either actioned immediately or added into the relevant backlog. The organization also uses an external penetration testing service. External penetration testers are given information about the system and access to the code and then asked to identify vulnerabilities.

The second author had studied the organization in the recent past, and so the organization was known to the research team. However, previous studies were not related to security practices.

IV. Method

The ethnographic method is used to study peoples' actions and accounts of actions. The method allows researchers to develop understanding about what practitioners working in socio-technical environments do and why they do it [30]. Ethnography permits researchers to consider experience from the perspective of the insider [30]. In this study, the aim is to understand the point-of-view that professional software developers have about security.

Ethnographic studies play a number of roles in empirical software engineering [30]. They can inform the design of software engineering tools and improve process development. This analysis intends to strengthen investigations into the social and human aspects of software engineering, and to assist in the exploration and refinement of the research question guiding the larger study, which asks:

How does security feature in developers' day-to-day practice within a non-specialist context?

Analysis in ethnographic studies often includes exploration of other open-ended questions that signal problems or provoke a sense of unease [31]. The account given here represents an analysis of this kind, by considering a subset of the data collected in light of the question that opened the paper:

Are developers responsible for security?

A. Data collection

The study was planned before data collection began within two initial meetings and several email exchanges. Data were collected at three different points in time.

The first period involved observation and informal interviews across the first sprint of a new release cycle. Three teams were observed, each with between 3 and 8 developers and testers, plus their scrum masters and product owners. The observations were driven by the activity of the developers in the team, but were focused on daily rhythms, and on uncovering security-related activities in daily work. During this period, contextual interviews were also undertaken with the Vice President of product development, the technical product

TABLE I
SECURITY-RELATED INTERVIEW QUESTIONS

Q1. Does security come up often in your work? - What needs to be secured? Or Who?
Q2. What is doing the securing?
Q3. Why is [the subject] being secured?
Q4. Who (or what) is [the subject] being secured from?
Q5. When did you start spending more time on security issues? - What caused this change?
Q6. How do you keep up with security technologies?
Q7. Do you feel you could/want to do more about security? - What kind of support would you need?
Q8. Do security measures get in your way? What do you do about it?

owner and the security officer from the parent company. The final set of data was collected during a workshop and feedback session.

Data collected in the first and third parts of the study were used to provide context for the account reported here, but are not reported in detail. These data included observation and meeting notes, physical layout sketches, screen captures of the software artefacts used, still photographs, audio recordings of interviews and meetings, and development artefacts such as wiki pages, user stories and exported data from chat software. The second period of the study forms the basis of reporting in this paper. In this part, semi-structured interviews were conducted with 13 developers, testers and technical managers from the teams that had been observed. Interviews were conducted by two of the researchers who also performed observations.

The development of the interview script for this part was informed by the preceding observation work, and other parallel project activities. Prior observations had taken place during everyday practice. Though a few instances of practice were observed that included security elements, this interview was the first time developers at this company were directly asked about security.

The interviews included questions of two types. The first set of questions, which are not reported here, examined aspects of the interviewee's career, drawing on theory related to motivation [32] [12]. The last portion of each interview was spent discussing the interviewee's attitudes and beliefs about security and secure software development in the organization.

Taking the view that security is a quality to be striven for, a set of eight baseline questions were defined to guide the interview, which were modified to suit circumstances with individual informants (Table I). In some cases, fewer than 8 questions were asked, in other cases as many as 17 questions were asked to probe for detail or to clarify particular points. The ordering of questions was not consistent; questions were asked in the sequence that made sense for the interaction.

B. Data Analysis

Analysis in ethnographic research begins at the point of collection as researchers formulate ideas about the significance of what was seen, and new questions arise about what else might need to be examined to answer the research question. The meanings formed in analysis must be forged through writing [33]: of field notes, transcriptions, descriptions and, ultimately, the accounts.

In this study, field work was discussed by the researchers, field notes were taken and key findings shared. From this overall contextual view, a subset of interview data were selected for analysis to develop a preliminary series of accounts relating to security themes. The analysis did not conform to one particular theoretical underpinning, but was informed by the various theoretical frameworks introduced in section II above.

Analysis proceeded in an iterative, inductive fashion, taking into account observations of the organization and developers' daily work practices. The purpose of the analysis has been to better understand the relationship between responsibility for security and software developers.

The analytic process was performed on interviews taken by one of the two interviewers. This researcher conducted 7 interviews over the course of three days with one software development manager, 5 developers and 1 tester from the teams that had been observed during the first week of the study.

The audio recording of each interview was transcribed and then examined to find themes [34] using the following steps:

1) Each interview response was segmented into meaningful units. Three categories of information were considered:
   a) A response or portion of a response that corresponded directly to a question from the basic schedule of eight — these represented the broad themes under examination.
   b) Factual or contextual information about operations or policies at the company.
   c) Information revealing a personal attitude or belief held by an interviewee.
2) The meaningful units for each interview were printed onto sheets of paper, cut into individual pieces and grouped together into sets to reflect the categories of information noted above. As Figure 1 shows, this was not an exhaustive process; the primary aim was to collate responses from different informants into related groups to facilitate comparison.
3) In the last stage of analysis, responses in each grouping were examined together to refine a set of categories and themes for reporting.

V. FINDINGS

In the rest of this paper, a preliminary account is given that provides insight into perceptions developers have about security in this environment. Some themes closely reflect questions that were asked, while others emerged out of the

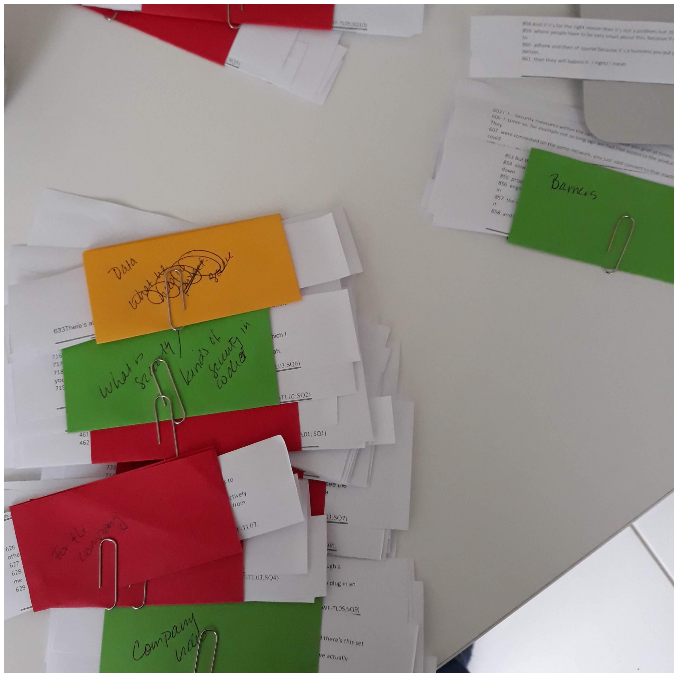
Fig. 1. Interview responses grouped into categories

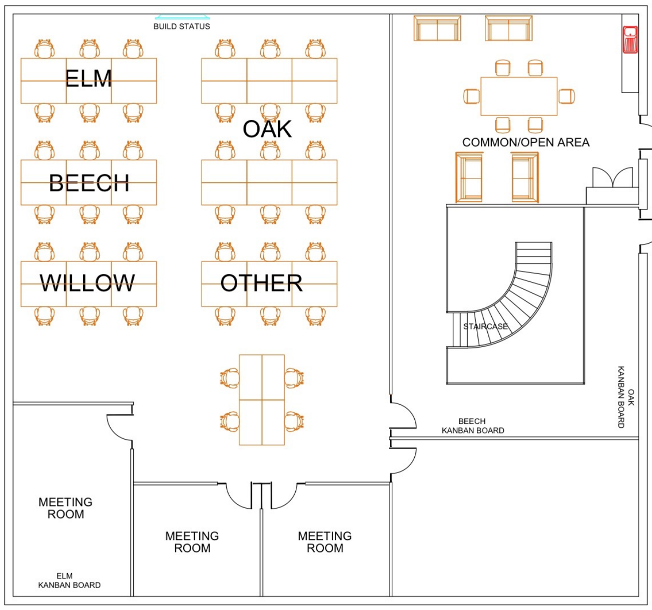

Fig. 2. Software development area. Note that this is indicative only, and not to scale

data when they were examined. In bringing together responses from different informants, the findings indicate a synthesis and situated definition. They reflect views about this particular environment, in the period during which data were collected.

An overview of the people whose interviews were analysed for this section is given in Table II, represented with pseudonyms. In the following sections, findings are presented with anonymous quotes given by these individuals; in cases of longer quotes, a pseudonym is referenced.

A. Context and Environment

Agile software development practices have been used by this organization for over five years. The teams studied are 'typical' of agile software development [35]. Each team followed a standard agile rhythm, beginning the day with a standup, and punctuating the day with code reviews, backlog grooming sessions and ad hoc team member conversations. Conversations are open, there is talk between individuals and between teams, with developers joining in to help others when needed. Daily stand-ups took place around the physical boards, indicated on the floorplan in Figure 2. The teams used a physical Kanban-style board for keeping track of work during standups, and an on-line system, Jira, to maintain all data.

The development teams are located primarily in one area of one floor of the building (see Fig 2). Some developers are located remotely, and are interacted with via videoconferencing or instant messaging, either as part of a planned meeting or ad hoc when needed. There are two meeting rooms and several breakout areas on the same floor, and further meeting rooms on the floor below. These are used for stand-ups, sprint planning meetings, retrospectives and other team and customer meetings. Five teams are located in the software development area (see Fig 2). The three teams that were the focus of this study, and from whom the interviewees primarily came, are profiled below.

1) Oak: Oak is one of three teams that add features to the core products. They also undertake work to maintain and improve codebases written in PHP, Javascript and C#. The team includes five engineers, three of whom are senior, and two testers. This team, recently formed, is proving strong, but the mix of experience and approaches can raise tensions between less experienced members who have a lot of "fun, new ideas", and those who have been in the profession for a long time, and prefer to use proven ways.

2) Elm: Elm is responsible for the company's mobile applications, written in Ruby and AngularJS. Recently, the team has also taken up some work in PHP to independently handle their integration with the core product set. Currently, the team has four members: three engineers and one tester. The lead engineer and tester have longstanding experience in the company; the lead engineer has come in and out of this team once or twice. One of the other engineers is more junior, recently brought over from a different team to take up work on the interface. A third engineer works remotely in a shared office space in a different country.

3) Beech: Beech is in charge of business intelligence. In this capacity, the team uses third-party reporting tools such as Talend and Juniper to write reports out of data produced by the core products. Recently, this bespoke reporting process has been migrated into a product that customers can incorporate into their own reporting systems. The team has three senior engineers, and access to a tester working remotely in India. The seniority of the engineers can make discussions tricky

TABLE II
INTERVIEWEE DEMOGRAPHICS

Identifier/role	Age	Years of experience	Years at ORG	Team
Oscar (SE)	24	1	1	Oak
John (Lead SE)	45	20	20	Elm
Ned (Senior SE)	46	13	1	Beech
Jacob (Lead SE)	29	10+	3	Oak
Ben (Software Development Mgr)	32	12	4	n/a
Frank (Tester)	40	17	17	Elm
Edmond (Senior SE)	67	45	17+	Oak

because everyone has strong opinions. But it also has a positive effect - the solutions are perceived by team members to be of high quality.

B. Developing Awareness

Workers develop awareness about security on the job and through engagement with the wider world [36]. In this environment, there was clear recognition among engineers of the importance of security for customers, and for the organization. Interviewees indicated an awareness of the need to avoid "reputational damage", and to meet the requirements of customers who "expect their data to be secure".

These impressions may have been formed, in part, by training about information security provided by the company. All employees at the organization are required to watch a security training video each year that includes a quiz. The video explains what kinds of information shouldn't be disclosed outside of the company, and how employees should manage data for clients on their laptop machines, or when using external storage devices. It also explains how employees can responsibly leave their computers unattended, and how to responsibly post and send information through email.

Interviewees showed varying levels of awareness of security matters within the software. Awareness is heightened in the environment among the engineering teams in five ways.

1) Audits or other compliance mechanisms like GDPR which can expose potential weaknesses in code, or issues that they had "never thought about".
2) Security scanning reports from a static analysis tool that generates automated pen-testing reports. Outputs from these reports are fed into code reviews.
3) Internal review processes, like code reviews. These can highlight that something was "missed" in, for example an authentication process, that will in turn trigger changes in the code, and may prompt changes to legacy code.
4) Internally documented standards about how secure code should be written. These were observed to have been recently added to the wiki by one engineer who remarked that these were prompting more discussion about the topic.
5) Reading others' code. Seeing security implementations in the code raises awareness that security measures have been taken, but not complete understanding of how they work.

C. Contextual Drivers

Awareness is only the "first step" [37] to secure systems. Workers also must engage with security [38], and they must take it up in practice.

Security was described by most of the interviewees as not being a top priority for the teams, as something that doesn't come up very often or only occasionally. Although security seems to be becoming a more common point of discussion, it remains a "minor aspect" of what the engineers do and is not at the "forefront of any decisions" that are made. Security was described in one or two cases as being something that people deal with a lot or frequently. The explanation for the difference in these two perspectives isn't clear.

During the period of this study, security at the organization was not perceived by the engineers to be in the hands of a single person. One developer explained that it was not "like an email" or a directive had been sent telling the developers that they need to look at security. Instead, security development was perceived to be event-driven. In the recent past, for example, the engineers made a number of security related changes in the code in response to a drive initiated by the sales department to meet the requirements of an audit. In cases like this, the need to comply with an external requirement triggers a process in the teams that produces a list of things that need to be "sorted out" in the code. This happens on an intermittent basis.

The engineers also observed that reports from the static analysis tool can prompt secure coding activity from time to time. One interviewee hadn't noticed anything coming from the static analysis tool vulnerability report in "quite a while". As he noted, this did not mean that issues were not being raised by the system, only that the issues may have been passed to other teams. This perception was confirmed by the technical Product Owner, who noted that the tool had not recently identified actionable security issues.

D. Handling Security

System security engineering refers to the software and hardware development life-cycle and includes a range of activities, including architecture design, code implementation, analysis of economic incentives and human psychology, as well as policy and assurance [39]. Security is handled differently by different teams in the organization. For teams building new features, security is considered "going forward", by looking at what is going into code, and checking it over in code reviews. That is the moment when the team as a whole says "these are the things we should and should not be doing". For teams that look after existing code, security is and will be about looking back, seeing what is currently there and trying to improve it.

Another engineer noted that security is a consideration when design is happening, when sprints are planned or when stories are tasked. That is the point at which assumptions are checked, and discussion is had about extra precautions that might need to be taken. However, this view wasn't shared by other engineers, and may represent desired, rather than actual, practice [31].

Recently, different safeguards have been put in place. Some of these measures were introduced by the company, while others were initiated within application frameworks:

1) Securing URLs and passwords. This has involved stronger encryption of passwords, and use of encryption on laptops, in other hardware and on the network.
2) Introducing access logging, now that we are in "GDPR age", to make sure that if changes are made to records, it is possible to track when and by whom the changes were made.
3) Securing access within the applications by making sure that users see what they are "supposed" to see. This also entails making sure that when a customer submits a form, it is not possible to just go into the form and change a variable and actually now make it affect someone else.

E. (Deliberate) Reliance on Others

One way the engineers ensure that code is secure is by relying on security features within the technologies. As they put it, "everything is already within the application", the code is already within "layers of security". Security controls are built into the databases, the network, and API endpoints, allowing the developers to "piggyback" on rules for access that have been defined and are controlled by clients. The developers thus give most of their time and attention to extending and developing the products, while relying on the "security stuff" already put into place to handle security.

This is not an entirely passive process. It is also deliberate. Engineers indicated that relying on existing code is something they must actively do. For example, the application framework has a built-in permission system. However, it is up to the engineers to "make sure" that the system is used when new pages are added to the product.

Indications are also given that colleagues provide support for security coding to one another in a number of ways, including through existing implementations in the code and in discussion during code reviews. When barriers to practice arise, for example in configuring SSH keys, engineers were observed to provide ad hoc information to one another to help work move along. Edmond explained the support in this way:

> *"[N]ormally there is someone you can [go] to and say well look I need to set this up. Can you tell me how?"*

VI. Threats to validity

Threats to validity relating to this study have been compiled using Runeson et al's guidelines [40].

Internal validity The main threat to internal validity is that the data collected be misunderstood or misinterpreted, or that the interview questions be misunderstood by the interviewees.

The interviews were conducted by experienced researchers, and the interview protocol allowed for concepts and issues to be explored thereby confirming joint understanding. The findings of this study have also been presented to and discussed with the study participants in order to check interpretation.

Construct validity A key purpose of the study was to understand the participants' views of security. The constructs were therefore coming from the participants and not imposed by the researchers. Existing literature on security and its relationship to developers was examined to mitigate this threat.

Reliability In an ethnographic study, the researcher is the instrument and in that sense, the results are dependent on the researcher involved. However, this study was not undertaken by a sole researcher, and two of those collecting data are experienced researchers in ethnographic software development studies. With the same focus and the same exposure to the organization, it is reasonable to expect that similar results would emerge.

External validity This study focuses on one organization. This limits the extent to which the conclusions presented here can be generalized. However, this organization operates in a similar fashion to other agile software development teams, in terms of practices, tools, and daily rhythms.

Generalization can be achieved through analogy (analogic generalization) and explanation (abductive inference). Analogical generalization takes a situation that has similar characteristics to the one studied and discusses whether the circumstances are similar enough to allow the conclusions to be applied to that situation. In this case it would require the circumstances of a different team to be characterized and the applicability of these findings to be discussed. With abductive inference, an explanation of activity can help support analogic generalization. The explanations of the studied team, provided in previous sections, gives information to explore whether the findings can be applied to another team [41].

VII. Discussion

The engineers shared the view that this organization is taking data protection for their clients seriously. On this point, one engineer was very positive, explaining that he didn't feel a need to advocate for data protection because the "information governance" at the company is stronger than it was in a prior field in which he had worked. There, he described the attitude taken as "*laissez-faire*" at best.

Yet, the belief was widely held that more could be done, that there is an "occasional lack of attention" or "lacking areas". Sometimes the lack was linked to individual knowledge or experience in the sense that "what you don't know, you don't know", but also to a lack of structure in how security related issues are currently handled. Though processes designed to be secure are being initiated by a security specialist in the parent company, there was a degree of skepticism conveyed about whether or not it would be possible to ensure in anyway that the safeguards were working.

The sense was also given that the engineers do not have a clear way to assess what “enough security” is. Security can be “good enough” at a point in time, but that may just be luck. Likewise, there was the sense conveyed that more could always be done. This particular observation was not couched in terms of how things are done at this particular organization, but more generally in terms of the nature of security in the context of software development.

In the main, however, the engineers believe that they are making efforts around security and doing pretty well. John mentioned that the team “would like to think that we are producing secure software” and hopefully “we mostly are.” He conceded, however, that things are sometimes brought to the attention of the engineers during code reviews or by audits:

> *“[W]e do miss things. And the security audits come along and expose some things that we have missed. Or sometimes they might expose a whole class of things that we’ve just never thought about and that causes a bit more of a severe issue.”*

A. Are developers responsible for security?

The engineers and testers in this environment do feel a responsibility toward making sure that the code they write and test is secure. Echoing views given in the introduction, interviewees commented that they should always bear security in mind. As Jacob described:

> *“It should be there at every stage. So we should be thinking about it when we’re grooming, we should be thinking about it when we are reviewing code. And then after that, we should be thinking about it when we are testing code”.*

However, while the need for security was asserted, the responsibility for upholding it was also perceived to be shared with other members of the teams. As an addendum to his comment, Jacob noted that testers might already be looking for vulnerabilities, but he didn’t know. The developers rely on security features written by other team members. These examples in the code are used to develop security awareness, but also to foster assurance within and among the engineers about making the code secure. The rationale given was that the software will be secure if “you do [it] in a way that everybody else does it”.

Likewise, indications were given that the engineering teams may not always feel ownership of security. Decisions about security measures to take have been made for the application frameworks, databases and networks. Several interviewees suggested that the principal software engineer had a role in these decisions, and knows more about security than the other engineers. As one interviewee explained, ”I’ve delegated that responsibility to [the principal software engineer]”. This was a position which he acknowledged was a ”get out” but also defended as being prudent, because this person is, if not a security specialist, more “specialised” and has looked at the problem of security in more detail.

This group of engineers believes the software they write is secure because they use security features within the technologies, and develop understanding among themselves about how to employ them and to ensure that they are properly used. They also follow a set of procedures and internally documented standards that they have been asked to use, by the principal software engineer or by the organization.

Suggestions were also made that responsibility for the security of the product lies, in the end, with the organization. The business is responsible for ensuring that the data it handles for clients isn’t “prone to being nicked” and for vetting the security landscape of vendors the company uses for things like data storage.

Even here, however, the perception was held that the limits of responsibility toward security go beyond the organization. As Ned explained, security is put in place to keep honest people honest. But security won’t keep determined, or dishonest, people out:

> *“If someone is determined, they will get into our systems either by a brute force attack, a technological attack, or by socially engineering a situation where they can actually get in through a back door. They find what they want, and they take it.”*

VIII. Conclusion

This study looked at the beliefs and perceptions of software engineers at one organization. The engineers had a broad range of levels of experience and roles. The organization is taking security seriously by introducing safeguards, policies and measures to provide security assurance. Likewise, the engineering teams take collective responsibility for security in the code they write and and test. They accept measures brought into the organization, and use events like audits to build knowledge and improve the code. They believe that the code they are writing is mostly secure, but they feel that improvements can still be made.

Developers do feel a responsibility toward security. The views of developers from this organization indicate a pragmatic approach toward secure coding practice that has recognized limits and boundaries. Policies, technical environments, infrastructure and social aspects of practice provide security assurance to some degree. Combined with the experience and skills of practitioners, security measures produce software systems that are arguably “mostly secure”, a view tempered by the understanding that responsibility for security is ongoing, shared among roles within the organization and beyond.

Acknowledgment

The authors would like to thank the organisation and the developers for supporting this study. Thanks also to Lizzie Coles-Kemp for introducing us to the international relations literatures and to Sorren Hanvey for providing background for the introduction. The work presented in this paper was supported by the UK National Cyber Security Centre. Nuseibeh thanks SFI, EPSRC and ERC for financial support.

References

[1] P. Danhieux, “Are Developers Your First Line of Security Risk or Defense?” Nov. 2018. [Online]. Available: https://devops.com/are-developers-your-first-line-of-security-risk-or-defense/

[2] T. Gillis, “Developers Shouldn’t Be Responsible For Security.” [Online]. Available: https://www.forbes.com/sites/tomgillis/2016/04/21/separate-is-good-developers-shouldnt-be-responsible-for-security/

[3] M. Neely, “Who is Responsible for Application Security? Development or Security?” Jan. 2013. [Online]. Available: http://www.infosecisland.com/blogview/22847-Who-is-Responsible-for-Application-Security-Development-or-Security.html

[4] N. Heath, “Should developers be sued for security holes?” [Online]. Available: https://www.techrepublic.com/blog/european-technology/should-developers-be-sued-for-security-holes/

[5] D. Shackleford, “A DevSecOps Playbook - SANS Institute.” [Online]. Available: https://www.sans.org/webcasts/devsecops-playbook-101472

[6] B. Fitzgerald and K.-J. Stol, “Continuous software engineering: A roadmap and agenda,” *Journal of Systems and Software*, vol. 123, pp. 176–189, 2017.

[7] V. Mohan and L. B. Othmane, “Secdevops: Is it a marketing buzzword?-mapping research on security in devops,” in *Availability, Reliability and Security (ARES), 2016 11th International Conference on*. IEEE, 2016, pp. 542–547.

[8] Microsoft, “Microsoft Security Development Lifecycle.” [Online]. Available: https://www.microsoft.com/en-us/securityengineering/sdl

[9] H. Rygge and A. Jøsang, “Threat Poker: Solving Security and Privacy Threats in Agile Software Development,” in *Secure IT Systems*, ser. Lecture Notes in Computer Science, N. Gruschka, Ed. Springer International Publishing, 2018, pp. 468–483.

[10] OWASP, “Top Ten Project,” 2017. [Online]. Available: https://www.owasp.org/

[11] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp, “Motivation in Software Engineering: A systematic literature review,” *Information and software technology*, vol. 50, no. 9, pp. 860–878, 2008.

[12] C. França, F. Q. B. da Silva, and H. Sharp, “Motivation and satisfaction of software engineers,” *IEEE Transactions on Software Engineering*, p. (Early Access), 2018.

[13] H. Sharp, H. Robinson, and M. Woodman, “Software engineering: community and culture,” *IEEE Software*, vol. 17, no. 1, pp. 40–47, Jan. 2000.

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

[15] M. Green and M. Smith, “Developers are not the enemy!: The need for usable security apis,” *IEEE Security & Privacy*, vol. 14, no. 5, pp. 40–46, 2016.

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

[17] Y. Acar, C. Stransky, D. Wermke, C. Weir, M. L. Mazurek, and S. Fahl, “Developers Need Support, Too: A Survey of Security Advice for Software Developers,” in *Cybersecurity Development (SecDev), 2017 IEEE*. IEEE, 2017, pp. 22–26.

[18] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, “You Get Where You’re Looking For: The Impact Of Information Sources on Code Security,” in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 289–305.

[19] A. Poller, L. Kocksch, S. Türpe, F. A. Epp, and K. Kinder-Kurlanda, “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*, ser. CSCW ’17. Portland, Oregon, USA: ACM, 2017, pp. 2489–2503. [Online]. Available: http://doi.acm.org/10.1145/2998181.2998191

[20] “The Security Mindset - Schneier on Security.” [Online]. Available: https://www.schneier.com/blog/archives/2008/03/the_security_mi_1.html

[21] B. Schneier, “Attack trees,” *Dr. Dobb’s journal*, vol. 24, no. 12, pp. 21–29, 1999.

[22] P. Roe, “The ‘value’ of positive security,” *Review of International Studies*, vol. 34, no. 4, pp. 777–794, Oct. 2008.

[23] B. McSweeney, *Security, identity and interests: a sociology of international relations*. Cambridge University Press, 1999, vol. 69.

[24] G. M. Smith, “Into Cerberus Lair: Bringing the Idea of Security to Light,” *The British Journal of Politics & International Relations*, vol. 7, no. 4, pp. 485–507, 2005.

[25] “Ethics: Don’t be derailed by the trolley problem | Business | Subject areas | Publishing and editorial | BCS - The Chartered Institute for IT.” [Online]. Available: https://www.bcs.org/content/conWebDoc/59735

[26] E. Winter, S. Forshaw, and M. A. Ferrario, “Measuring Human Values in Software Engineering,” in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM ’18. New York, NY, USA: ACM, 2018, pp. 48:1–48:4.

[27] B. Friedman, P. H. Kahn, A. Borning, and A. Huldtgren, “Value Sensitive Design and Information Systems,” in *Early engagement and new technologies: Opening up the laboratory*, ser. Philosophy of Engineering and Technology. Springer, Dordrecht, 2013, pp. 55–95.

[28] S. Turkle, *Alone together: Why we expect more from technology and less from each other*. Hachette UK, 2017.

[29] H. Sharp, N. Baddoo, S. Beecham, T. Hall, and H. Robinson, “Models of motivation in software engineering,” *Information and software technology*, vol. 51, no. 1, pp. 219–233, 2009.

[30] H. Sharp, Y. Dittrich, and C. R. B. d. Souza, “The Role of Ethnographic Studies in Empirical Software Engineering,” *IEEE Transactions on Software Engineering*, vol. 42, no. 8, pp. 786–804, Aug. 2016.

[31] M. Hammersley and P. Atkinson, *Ethnography: Principles in practice*, 3rd ed. Routledge, 2007.

[32] E. H. Schein, “Career anchors revisited: Implications for career development in the 21st century,” *The Academy of Management Executive*, vol. 10, no. 4, pp. 80–88, 1996.

[33] B. Anderson, “Work , Ethnography and System Design,” in *The Encyclopedia of Microcomputers*, A. Kent and J. G. Williams, Eds. Marcel Dekker, 1997, vol. 20, pp. 159–183.

[34] V. Braun and V. Clarke, “Using thematic analysis in psychology,” *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[35] H. Sharp and H. Robinson, “Three ?c?s of agile practice: collaboration, coordination and communication,” in *Agile Software Development: Current Research and Future Directions*, T. Dingsøyr, T. Dybå, and N. B. Moe, Eds. Berlin: Springer, 2010, pp. 61–85.

[36] S. Furnell and A. Rajendran, “Understanding the influences on information security behaviour,” *Computer Fraud & Security*, vol. 2012, no. 3, pp. 12–15, Mar. 2012.

[37] M. Beyer, S. Ahmed, K. Doerlemann, S. Arnell, S. Parkin, M. Sasse, and N. Passingham, “Awareness is only the first step,” Hewlett Packard, Business white paper, Tech. Rep., 2015.

[38] C. Weir, A. Rashid, and J. Noble, “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)*, 2016.

[39] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2nd ed. Wiley Publishing, 2008.

[40] P. Runeson and M. Host, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical Software Engineering*, vol. 14, pp. 131–164, 2009.

[41] R. Wieringa, “Guidelines for conducting and reporting case study research in software engineering,” *Journal of Systems and Software*, vol. 95, pp. 19–31, 2014.