How Does Usable Security
(Not) End Up in Software Products?
Results From a Qualitative Interview Study

Marco Gutfleisch [*], Jan H. Klemmer [†], Niklas Busch [†],
Yasemin Acar [‡], M. Angela Sasse [*], and Sascha Fahl [†§]

[*]Ruhr University Bochum, Germany, {marco.gutfleisch, martina.sasse}@ruhr-uni-bochum.de
[†]Leibniz University Hannover, Germany, {klemmer, busch}@sec.uni-hannover.de
[‡]Max Planck Institute for Security and Privacy, Germany, yasemin.acar@mpi-sp.org
[§]CISPA Helmholtz Center for Information Security, Germany, sascha.fahl@cispa.de

Abstract—For software to be secure in practice, users need to be willing and able to appropriately use security features. These features are usually implemented by software professionals during the software development process (SDP), who may be unable to consider the usability of these mechanisms.

While research has made progress in supporting developers in creating secure software products, very little attention has been paid to whether and how these security features are made usable. In a semi-structured interview study with 25 software professionals (software developers, designers, architects), we explored how they and other decision-makers encounter and deal with security and usability during the software development process in their companies.

Based on 37 hours of interview recordings, we qualitatively analyzed and investigated 23 distinct development contexts in detail. In addition to individual awareness and factors that directly influence the implementation phase, we identify a high impact of contextual factors, such as stakeholder pressure, presence of expertise, and collaboration culture, and the specific implementation of the SDP on usable security in software products. We conclude our work by highlighting important gaps, such as studying and improving contextual factors that contribute to usable security and discussing potential improvements of the status quo.

I. INTRODUCTION

Software security is important. However, security alone is not enough. Instead, security features also need to be usable: Users have to use the security features and use them correctly.

Previous research demonstrated the ineffectiveness of poorly designed warning messages [5], [14], [16], [22] the prevalent issues with passwords [18], [25], [47], or security and privacy problems due to wrong mental models and misconceptions. For example, most Android users do not pay attention to permissions and lack comprehension, which can lead to wrong security decisions [17]. Secure systems, however, should account for all these human factors to be actually secure; namely, they should implement *usable security*.

The creators of software, such as software developers and designers, greatly influence whether security is implemented in

a usable way. However, while many unusable security mechanisms are known to have created problems in the past [69], and while research has been done on how developers (fail to) implement security [1], [2], [24], the root causes for lack of *usable* security (or successfully implemented usable security) in software products is yet unknown.

In addition to software professionals' individual factors, such as awareness and skill sets, we hypothesize that similarly to security [24], the entire software development process (SDP) and contextual factors impacts software product usable security. This software creation process involves many more stages than writing code, ranging from requirements engineering to deployment, and often involves multiple stakeholders, e. g., customers, management, and designers. In this study, we investigate how usable security is handled within the software development process in practice. Therefore, we conduct 25 semi-structured interviews with software professionals, including developers, architects, designers, and other decision-makers. We use qualitative coding to extract and explore problems, practices, and motivations for secure and usable software [11]. With this study, we aim to explore the following research questions:

RQ1: *Which factors in the software development process and in companies influence usable security?*

RQ2: *What are contributors and blockers for usable security in software development?*

RQ3: *When and where in the development process are important decisions made that influence usable security?*

Based on our results, we aim to deepen the understanding of how usable security can be achieved while also identifying factors that prevent usable security. We analyze how company culture and contextual factors contribute to usable security and suggest how academia and industry can improve the adoption of usable security by improving awareness, interdisciplinary collaboration, and developing measures that support a holistic usable security process.

The remainder of this paper is structured as follows: We give an overview of related work in Section II. In Section III,

we present our methodology, including the interview guide and our analysis procedure. Section IV presents our findings. We discuss our results' implications in Section V, where we also propose interventions for better usable security in SDPs and highlight implications for academic research. Section VI concludes the paper.

II. RELATED WORK

We discuss related work in three key areas: (1) the history of usable security, (2) studies and experiments with software professionals, and (3) organizational processes and security culture in software development processes.

Usable Security. Usable security has been heavily researched for more than 20 years. In 1996, Mary Ellen Zurko established the term *user-centered security* and made the case that usability for different groups of non-security specialists (software developers, system administrators, and end-users) was a necessary condition for systems to be secure and function in practice [71]. In 1999, Whitten and Tygar demonstrated in their seminal paper that the target users of PGP 5.0 were not able to operate it securely and identified the usability issues that were the cause [69]. Stransky et al. [60] conducted a field study in 2021 and confirmed that end-to-end encryption for email is still a huge issue. Adams and Sasse found that the password policies common in organizations were hard to follow and led to constant skirmishes of workarounds and sanctions that did little to make non-specialists take security seriously [4]. The usability of authentication and encryption for end-users has improved over the past 20 years. However, in many organizations, people still encounter impossible security tasks and are blamed when they cannot cope [52], [55].

Over the past decade, the research community started to apply and transfer usable security principles to developers: In 2016, Green and Smith suggested supporting developers and presented ten principles for usable and secure cryptographic APIs [19], and Acar et al. proposed a research agenda for developer-centered usable security [3]. Pieczul, Foley, and Zurko extended this approach and called for the development of new measures for *developer-centered security* in addition to the transfer of principles from end-users to developers [48].

Studies and Experiments with Software Professionals. When software professionals choose insecure or unusable options, this impacts the end-users interacting with the software – so improving usability for this group improves security for everyone. Research on "developer-centered security" has produced important insights by conducting empirical studies with software industry professionals (e. g., developers, system operators) that document and analyze their knowledge (and knowledge gaps), attitudes, and behaviors around security. Naiakshina et al. [37]–[40] and Danilova et al. [12] found that developers struggle with the secure implementation of password storage. Fahl et al. [15] found that many developers fail to provide secure TLS implementations in Android apps, which is still a problem as recent studies by Oltrogge et al. demonstrate [44], [45]. In 2017, Acar et al. examined the usability of Python cryptography libraries and identified

several problems with those APIs leading to insecure code [1]. The problem was also identified by others [36], [46]. In 2017, Krüger et al. presented the IDE plugin CogniCrypt that assists developers in using cryptographic APIs securely [32]. This tool was later extended by CogniCrypt$_{GEN}$ to generate secure crypto code fully automatically [33]. Nguyen et al. created the IDE plugin FixDroid and demonstrated that it significantly helps Android developers in writing more secure code [41]. Acar et al. examined the impact of information sources on development and found the usage of StackOverflow to produce more functional correct but less secure solutions compared to the official Android documentation [2]. Ruef et al. presented a novel contest for secure development and a quantitative analysis of emerging vulnerabilities [51]; Votipka et al. performed a qualitative analysis to understand what security mistakes are made by developers and why [64]. All these studies focused on developers as users of programming languages, APIs, documentation, and tools necessary to create secure software products – but they do not support the usability aspects of security.

Another strand has studied how system operators and other expert users handle security. Krombholz et al. conducted an experimental study with system administrators and reported that they struggled with the complexity involved in implementing secure TLS [31], and an interview study revealed that the same group administrators have significant misconceptions about HTTPS [30]. Yakdan et al. created and evaluated a usability optimized decompiler for malware analysts [70]. While those studies provided in-depth insights into the nature of their difficulties, they only studied one specific group of software professionals on one specific security task in the implementation and deployment phase of the SDP. Research has also identified obstacles to fast update behavior with Android developers [13] and system administrators [62], also finding the important impact of organizational factors, such as management and policies [34].

Our study investigated these aspects in both more breadth and depth by asking software professionals specifically about individual and organizational factors that contribute to *usable* security.

Organizational Processes and Security Culture. In addition to individual factors that contribute to security, research has identified organizational processes and culture that impact software security [23], [26], [28], [49], [57], [58], [63]. Haney et al. studied organizations developing cryptographic products [24] and found a strong security mindset as well as high expertise in cryptography. Thomas et al. conducted interviews with 32 application security experts and concluded that organizational processes and related factors must be improved to improve software security [61]. In an interview study, Assal and Chiasson identified company culture as a reason that security best practices are not followed in software development life cycles (SDLC) [8]. In another paper, Assal and Chiasson evaluated their assumptions quantitatively and concluded that security processes being followed depends more on organizational processes than individual develop-

ers [7]. In an interview study, Votipka et al. found that good communication between hackers and software testers is important for finding vulnerabilities, leading to better security outcomes [65]. Stevens et al. presented a case study evaluating the introduction and effects of threat modeling in an enterprise environment [59]. They found that this is beneficial for the overall security, only leveraging existing resources. In 2019, Lopez et al. conducted an ethnographic study with software developers and concluded that awareness of 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 [35]. Weir et al. found a high need for security, only 14–22% of Android developers have access to security experts, and identified the need for increasing the use of assurance techniques [68]. They proposed and evaluated a set of lightweight assurance techniques that help developers create more secure software and raise awareness [67]. The effectiveness was evaluated further in a long-term study [66]. In 2020, Shostack and Zurko underlined the need for research on secure software development techniques [56]. However, there has been little research with a specific focus on *usable* security: In 2016, Caputo et al. conducted three exploratory qualitative case studies in large US companies that investigated organizations' approaches to make their security products more usable [9]. The authors identified barriers to this goal, namely that usability was perceived as less important than security and common sense, i.e., not requiring specialist knowledge or skills. In a preliminary study mainly with end-users, Iacono, Nguyen, and Schmitt analyzed requirements for usable security using a mixed-methods approach and identified the need for lightweight measures, e.g., models, patterns, guidelines, tools, and checklists [27]. We recruited a set of developers from across the industry to see which contextual factors impact the SDP and usable security outcomes.

III. METHODOLOGY

In this section, we explain the methodology of our interview study, including data collection and analysis (see Figure 1). We chose semi-structured interviews because of the exploratory nature of our research questions. It also affords the flexibility of letting participants relate their experiences with usable security – including practices we did not expect – while going into more depth with follow-up questions. The 25 interviews lasted on average 1:30:06 hours, were audio-recorded, and then transcribed by the authors. All interviews were conducted remotely between August 2020 and January 2021.

A. Interview Guide

Here, we describe the process of iteratively developing our pre-questionnaire as well as the interview guide itself. [1]

Instrument Development. We were interested in our participants' professional experiences with usability, security, and hopefully, usable security. We developed our interview guide
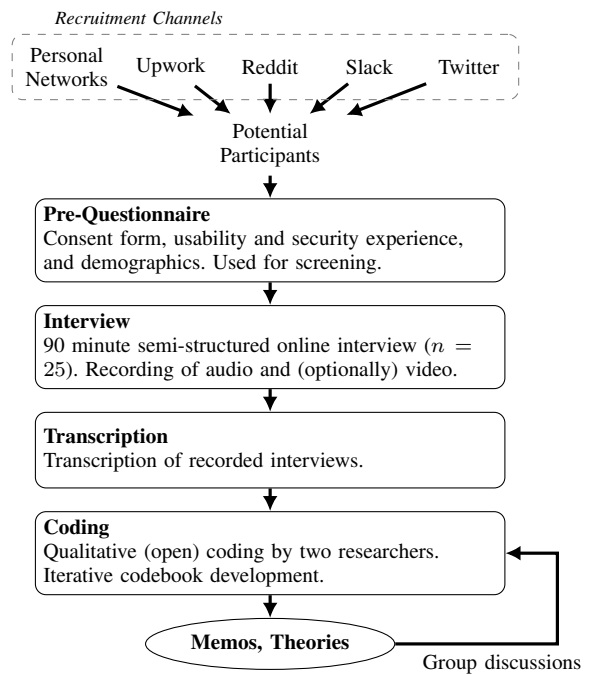

Fig. 1. Methodology overview.

based on prior research on usable security [9] to investigate the roles that usability, security, and usable security play in the development processes of the companies our participants work at. All authors reviewed and iterated the interview guide. Two authors (who had worked as software professionals) tested the guide on each other, then piloted it with a colleague who also had professional software development experience as well as with the first participant. We adapted the guide after each round of feedback. After the interview with the first external pilot participant, very few minor edits were needed, so we included that participant in our sample.

Pre-Questionnaire. Based on the duration of the pilot interviews, we moved several interview questions into a pre-questionnaire which participants were asked to fill in after consenting to the study and before scheduling the interview. [2] The pre-questionnaire asks for participant demographics and a self-assessment of knowledge about and experience with usability and security; we referred to the participants' pre-questionnaire answers during the interviews.

Interview Guide Structure. Our interview guide consists of three parts, each covering a category; first *usability*, second *security*, third the interplay, *usable security*. As we were interested in specific software development processes for usability, security, and usable security, we started the interviews by asking participants to think of a specific software project that contained a security feature with a user interaction component (e.g., login forms, warning messages) and tried to stay in

---

[1]The interview guide can be found in Appendix D, as well as the replication package (cf. Appendix B).

[2]The pre-questionnaire can be found in Appendix C, as well as the replication package; the consent form is in the replication package (cf. Appendix B).

the same context for all three parts of the interview. For the usability and security parts of the interviews, we asked the participants to walk through and explain the SDP. This included information about the product and the surrounding context (e. g., company, team, budget) and covered the whole SDP from modeling and requirements engineering to deployment. In the usable security part of the interviews, we asked about methods used to achieve usable security in the software products.

B. Interview Procedure

Those interested in participating in our study were sent to a landing page informing them about the study's purpose and screened for eligibility for the interview. Then, they were shown our consent form where we informed them about how we would handle their data, their right to terminate the interview at any time without repercussions, and where they could agree to audio-recording and opt in to the interview being video-recorded. They were then asked to fill in a pre-questionnaire, after which they could schedule their interview.

Interview Setup. We used a locally-hosted open-source, browser-based meeting tool for all interviews. All interviews were audio- and, if participants consented, video-recorded via the meeting tool's functionality, and we captured audio locally as a backup.

Interview Process. All interviews were conducted by the first author, when possible, with the support of one of the other authors, which allowed for supplemental questions, and, after the participant left the meeting, discussion and debriefing sessions. The interviews were conducted in English ($n = 14$) and German ($n = 11$).

C. Recruitment

We wanted to recruit software professionals who could offer insights into how usability, security, and usable security are addressed, including software developers and software designers. Therefore, we used *purposive sampling*: we recruited participants who matched our criteria in terms of role and level of experience in software development. We recruited through all authors' professional networks, via Reddit, Twitter, relevant Slack workspaces, and Upwork. We advertized the interview as a study for those "willing to share their professional experience with usability and security" with us. Participants qualified if they had professional experience with user interfaces or user interactions and professional experience with software security. They also needed to be willing to talk about a specific software development process (vs. only giving vague answers).[3] We offered $105[4] in compensation to be competitive with other job posts on Upwork and attract experienced developers. Moreover, for those interested in the topic, we invited them to participate in a workshop about

[3]Recruitment materials can be found in our replication package (see Appendix B).

[4]Based on €90 (€1 per minute), according to the USD/EUR exchange rate at the time.

TABLE I
PARTICIPANTS' DEMOGRAPHICS.

Gender					
Male	20	80.0%	Prefer not to answer	1	4.0%
Female	4	16.0%			
Country of Residency					
Germany	10	40.0%	Lebanon	2	8.0%
United States	4	16.0%	Other	7	28.0%
India	2	8.0%			
Age [years]					
Min.	24		Max.	60	
Mean (Std.)	35.2	±8.3	Median	33	
Industry Experience [years]					
Min.	3		Max.	30	
Mean (Std.)	11.7	±8.8	Median	10	
Education					
High school	1	4.0%	Graduate school	1	4.0%
College	2	8.0%	Master's degree	8	32.0%
Vocational degree	1	4.0%	Doctorate / PhD	2	8.0%
Bachelor's degree	8	32.0%	Prefer not to answer	2	8.0%

usable security free of charge. We stopped recruiting after our interviews reached saturation [11].

D. Demographics

We interviewed 25 participants; see Table I for an overview of participant demographics and Table III, and Appendix A for full participant details. The majority were from Germany and the US, and we also interviewed participants from India, Lebanon, Venezuela, and the United Arabic Emirates. We interviewed 20 men and 4 women. Participants' ages range from 24 to 60 years, with an average of 35.2 years. This tracks with a high level of professional experience in the software industry with an average of 11.7 years and a maximum experience of 30 years. Our participants also reported a high level of formal education; 19 participants (76.0%) hold at least a Bachelor's degree and 10 participants (40.0%) a Master's or Doctorate degree. While most interviewees were software engineers, architects, or developers, we also interviewed participants who held founding or C-level roles, as well as usability, design, and front-end professionals. Almost half of the participants were recruited via the authors' personal networks ($n = 12$). The majority of the rest was recruited via the freelancer platform Upwork ($n = 10$), while a small number were recruited via online communities and platforms ($n = 1$ for Twitter, Reddit, and Slack each).

E. Transcript Analysis

Data Processing. Members of the research team transcribed the interview recordings. In addition to audio, notable reactions visible on video (such as "air quotes" or "raised eyebrows") were also transcribed. During transcription, we replaced sensitive information like company names, personal information, or relationships with pseudonyms; some participants asked us to omit specific statements from our transcripts, a request that we honored. We matched the pre-questionnaire data to the appropriate transcript, then destroyed the recordings as promised in the consent form.

Qualitative Coding. We used both inductive and deductive approaches for data analysis: The first coder developed a code list based on the phases of software engineering as described by Pressman and Maxim [50], our research questions, the interview guide, and insights from the interviews. The codes were then iteratively refined over the process of coding nine transcripts independently with two authors, then discussing and adapting the codebook until all authors agreed on the codebook. All codes were formally defined ("operationalized") to ensure agreement by all coders. Two authors then coded all transcripts, re-coding those used for codebook creation, and merged their codes for analysis. The final codebook (cf. Table IV) addressed concepts like usability, security, usable security, concepts related to mental models (such as "misconceptions"), factors that influence the software development process (such as budget and resources), as well as the software development process itself. Throughout the coding process, we explored the relationship between the codes as well as patterns and themes. The authors regularly met and discussed the emerging concepts, which allowed the focus of this paper to develop: how software development processes influence usable security in software products.

F. Ethics and Data Protection

Neither of our institutions had an institutional review board (IRB) nor an ethics review board (ERB) that applied to our study. We adhered to the strict German and EU privacy laws, and the main author's institution's data protection officer reviewed and approved our study and our data handling process. Our consent form is compliant with the European General Data Protection Regulation (GDPR) and covered all information usually required for US IRB approval. Participants were informed that they could terminate the study at any time. We clarified all of their questions regarding the consent form, data processing and storage, and privacy ahead of the interview. All participants consented to our data handling, collection, usage practice, and publication strategy. We clarified that we would only evaluate de-identified data, publish aggregate data and de-identified quotes. We deleted all video and voice recordings post transcription to minimize unnecessary storage of PII, including information that was accidentally revealed in the interviews, and de-identified during the transcription process. To further prevent re-identification, we report results linked to development context independently from participant IDs. A more detailed discussion of our measures to ensure protection of participants' data can be found in Appendix A.

G. Limitations

This paper describes results from a qualitative study with a purposive sample. Therefore, all counts reported in our results are used to convey weight but should not be taken as quantitative results, statistics, or as claims to generalizability. Our goal was to understand a wide range of processes and conceptions about usable security. We are confident that we reached theoretical saturation in participants' answers for our purposive sampling strategy. Due to our advertising and the questions we asked, participants likely knew that we were interested in usability, security, and usable security – which could raise concerns about priming. However, most participants reported a lack of processes for usability, security, and usable security, as well as disinterest in the topic in their professional experience, so the concern is not warranted. This qualitative study aimed to explore current practices surrounding usable security and if/how it is supported. Possible follow-up work can build on this with quantitative methods and the goal of generalizability to verify our results. Additionally, while we aimed for a diverse sample, our sample skewed Western and male, aligning with human factors research studies with software professionals. While individual developers might tend to provide their personal opinions, our interviews prompted participants to report specific organizational experiences and context information. Individual recollections are, of course, subject to self-censorship and biases [6], most importantly self-reporting and social desirability bias, e.g., interviewees claiming that usability is important to them when it is actually not. However, for many aspects like usability, security, and others, our in-depth interviews and the fine-grained process explanations allow us to assess actual procedural focus on specific topics, as opposed to individual assessment of importance.

IV. Results

In this section, we present the results and themes that emerged from the analysis of the 25 semi-structured interviews. We first provide an overview of development contexts, i.e., companies and projects (for a detailed breakdown, see Table II). We assessed each company regarding usable security, identifying four different categories (cf. Figure 2): (1) *Usable Security Awareness and Follow-through*, (2) *Usable Security Awareness, Little or no Follow-through*, (3) *No Usable Security Awareness, some User-centered Measures*, and (4) *No Usable Security Awareness, few or no User-centered Measures*. We present the findings in each category within the four sections starting at Section IV-B. In this and the following sections, we report counts of interviews. However, as we conducted qualitative analysis, the counts should only give weight to the themes we found and therefore should not be interpreted as a quantitative result.

A. Software Development Contexts

Companies and Projects. Here we report the *contexts*, i.e., the companies, organizations, and projects that participants related to in the interviews. 21 of our 25 interviews covered at least one company context in detail, resulting in a total of 23 contexts (see Table II). The companies sizes' ranged from five companies being very small (1–9 employees), eight small (10–99), five medium sized (100–999), two large (1000–5000), to three companies being very large (>5000). In the remaining four interviews, participants shared their professional experiences and impressions across several projects and organizations, rather than focusing on a single context or company. While this did not allow us to investigate specific

TABLE II
SUMMARY OF THE CONTEXTS IN OUR INTERVIEWS.

	Product	Company Size	Awareness	User-Centered
C1	Passwordmanager	Very Small	●	●
C2	Office Suite	Very Large	●	●
C3	Cloud Project	Very Large	●	●
C4	Secure Communication	Small	●	●
C5	Service for Postal Deliveries	Very Large	●	●
C6	Fitness App	Small	○	●
C7	Access Control (Cars/Trucks)	Very Small	○	●
C8	Secure E-Mail	Small	●	○
C9	Document Processing Software	Small	●	○
C10	Secure Messaging	Small	●	○
C11	Cryptocurrency Web Wallet	Medium	●	○
C12	Secure Configuration IoT	Medium	●	○
C13	Secure E-Mail	Medium	●	○
C14	Secure Mobile App	Large	○	○
C15	Addon for CRM	Small	○	○
C16	Document & Data Management	Small	○	○
C17	Internal Administration Software	Very Small	○	○
C18	Document Signing	Medium	○	○
C19	PDA Delivery Assistant	Large	○	○
C20	Tracker medical devices	Very Small	○	○
C21	Social Distancing Wearable	Very Small	○	○
C22	Monitoring Trains	Small	○	○
C23	Security Product	Medium	○	○

Fig. 2. Categories that emerged from the contexts in our interviews.

contexts in-depth, these experienced participants (mean industry experience: 13.5 years) and how they compared different contexts they encountered in their professional career provided a cross-cutting perspective that helped comparing different contexts described in the other interviews.

Usable Security Status. As a first step, we analyzed all 23 company contexts regarding their focus on and their awareness of usable security. If participants were aware of usable security and reported its relevance for the product or the company, we classified those contexts as being ***aware of usable security***. We made this assessment based on the parts of the interviews we had coded *Interplay Usability & Security*, *Product*, *Importance*, and *Responsibility*. Contexts were also classified in this category when usable security was not directly mentioned, but the underlying concept was understood. We were able to reconstruct each company's SDP from the codes *Communication & Modeling*, *Construction*, *Deployment*, *Staff & Team*, *Budget & Resources*. We analyzed whether ***user-centered development measures*** were taken, whether (1) participants mentioned systematic usability testing, and whether (2) they conducted user research, or performed strong and active engagement with their user community. We found various forms of empirical user testing, expert evaluations, or field observations. To assess whether the reported development process was user-centered, we relied on the key principles for user-centered system design by Gulliksen et al. [20]. Based on the classification along those two axes, we obtained the four categories shown in Figure 2. We provide an in-depth description of the categories and their characteristics below.
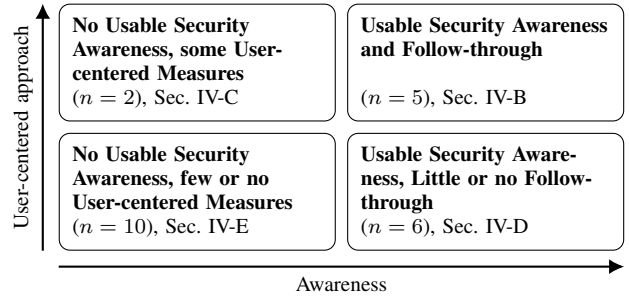
B. Usable Security Awareness and Follow-through

We placed five of 23 contexts (C1–C5) into the top-right category in Figure 2 because they followed a process that is characterized by user-centered methods, and they were aware of the underlying principles of usable security:

> *"Actually, I think that the security issues should be solved as much as possible technically and should not bother the user. Most of the time, the user is not an expert [. . .] That is, all the decisions that can be made for him, should be made in advance."* (C1)

We also found that in all contexts of this group, at least one expert for either usability or user interaction and security worked closely with the development team. Even if they were not members of the software development team, they were actively involved in the process.

Within C5, the UX expert was the pivot point for creating the interaction design. They actively involved a variety of stakeholders and pushed the user-centered thinking forward:

> *"Usually, with the stakeholders or directors, you are not really in direct contact with them. But for me, I developed a relationship with each of them [. . .] I made sure that they were more to date every week, [. . .] sometimes the developers get left out [. . .] so I made sure that the developers and the stakeholders were very involved in the design process."* (C5)

In addition to the availability of expert knowledge, we found an organizational structure that supports close interdisciplinary collaboration between different expert groups to be one of the key factors enabling usable security. In the above example (C5), the motivation to invest in a redesign of a security feature was based on past experiences with problems, i. e., customer complaints. Good usable security was thus the main driver for the project, and most stakeholders were aware that was the case. This participant also reported that user-centered processes were common in this company context: The team systematically examined the old application, and the problems users had with it. Therefore, the UX team was in continuous exchange with the development team to check which improvements were feasible. They conducted laboratory testing with real users to evaluate their design prototypes early. Moreover, they involved various experts from other teams like privacy experts or content designers to improve their product continuously. This is similar for the other four contexts in which we also observed that usability, including

usability of security features, was part of the business goals. Compared to the contexts from the other categories, we found that these five companies had user-centered methods like *user testing*, *A/B tests*, *expert evaluation*, or *prototyping* integrated within their SDP. Their development was also characterized by producing small and incremental artifacts. Nearly all of our 25 participants claimed that the companies they worked for employed beta-testing. However, not all companies actively collected feedback from them and used them for further testing.

C. No Usable Security Awareness, some User-centered Measures

Within this category, we found two contexts (C6, C7) that showed elements of a user-centered development culture – without concrete awareness of usable security concepts as such. Both participants reported that usability had high priority. One company developed and improved its product incrementally based on user feedback. Therefore, they had specific testing groups for usability and actively gathered feedback, recorded usability issues, and prioritized them in development. By contrast, security ranked lower in importance, and neither security experts were involved nor were security measures applied. Security features in the user interface were treated similarly to other features without additional effort:

> *"They were more concerned about usability. Like I said, they even made decisions which sort of decreases security, develop the product just so that it is more usable. [...] So, when we decide to go ahead with implementing a security feature, we then do not look at the original idea of the usability of it at all."* (C6)

For the other company, the customer explicitly requested security, and functional feature requests often included a security-relevant function (e. g., access control of cars). Still, their main focus was on usability. They did not involve designers and only brought in external security experts in certain development phases. Driven by specific customer requests, security mechanisms were adapted to customer needs (e. g., trying to make authentication as easy as possible for truck drivers). While their processes were user-centered, we found fundamental misconceptions about the interplay between security and usability (e. g., the trade-off myth). These misconceptions were present both in product descriptions given by the participants as well as in their direct statements: *"if we created an easy authentication method, I think its more likely to be vulnerable at some point, because usability cause security... security... don't know, security risks..."* (C7). We focus on misconceptions in Section IV-G2.

D. Usable Security Awareness, Little or no Follow-through

We observed that the development teams in six out of 23 contexts (C8–C13) may have been aware of usable security and may have even been interested or motivated to implement it. However, there were no supporting structures. All except one of these companies (C9) had dedicated security experts working on the projects, with a strong focus on security. One participant spoke of a *"very strong security focus"* (C10), and another said they *"put security at the forefront, and privacy and data protection"* (C8).

Only one of those companies (C11) had a designer or UX expert actively involved in the development process. In the other five contexts, design decisions were often made by developers and project leads. All participants reported that they followed an agile philosophy in their company. However, we found neither specific methods for evaluating usability nor dedicated measures for user research. Compared to the two previous categories, the interviewees did not report notable user-centered approaches. While some of them reported having tried some user-centered methods, they were used to a lesser extent. Two contexts, C8 and C10, however, had at least a public community of users. Nevertheless, compared to some of the companies described in Section IV-B who actively used their communities to request feedback, the relationship here can be described as more passive. However, participants were aware of concepts underlying usable security and sometimes even the terminology. Most attributed importance to usable security in their product but had not implemented mature measures for usable security in their processes. In the other companies in this category, designers or UX experts were not available. One participant of a medium-sized security company C13 described how they design interfaces themselves, because they were unable to hire an employee with skill sets in UX design and security:

> *"No, actually, we had for a very long time an open position for a UI/UX designer... but if you are looking for a UI/UX designer with additional qualification in the security environment... then, yes, you have to make one yourself."* (C13)

Other statements suggested that there is no need for a dedicated UX expert or designer: *"We are relatively small as a company and [...] don't have a usability expert now, unfortunately. But mainly ..., those are things that everyone has to learn themselves to a certain extent."* (C11).

We also observed that, unlike security, there were no specific requirements for usability. Usability was partly described as a handiwork of the developers:

> *"We give the developers a lot of freedom to design and implement features, so there are corporate design guidelines for colors and fonts and stuff like that, but interaction patterns and so on are more up to the developers themselves."* (C11)

Unlike the vague description of usability requirements, security was specified precisely. One participant told us that certification by a national office for software security means meeting concrete security requirements is a priority, even if this negatively affects usability. Other participants also mentioned that complying with security standards had an adverse impact on usability:

> *"So a thing, in general, is the interplay between security, especially in terms of approval, certification, and user experience, because I often find myself having to argue against a user experience thing because of some security thing, some box that has to be checked in order to get certification."* (C13)

We discuss conflicts created by security compliance standards in Section IV-G4.

We made two key observations: Firstly, some participants considered their processes to be sufficient for achieving usable security. Secondly, some participants openly admitted that they did not know how to implement certain security requirements without compromising usability.

E. No Usable Security Awareness, few or no User-centered Measures

In 10 of 23 contexts (C14–C23), we found that companies were unaware of the interplay between usability and security and spent little or no resources on the usability of their product. In five of those contexts (C14, C18–C20, C23), security was an important part of their business. In contrast, usability was taken much less seriously:

> *"I think the effort that goes into security-related topics and implementing them perfectly is many times greater than the ambition in the usability area, which is based more on the fact that the team has a claim to deliver. With security, it's about delivering [laughs]."* (C14)

Four companies (of which two were focused on security products) had actively involved experts for UX and design within their SDP. For all other contexts, even when products contained multiple security features, they did not see a need for usability experts. For example, one software team developed software for document signing (C18). This assessment was independent of the fact that usable security was well-understood: *"I don't think there was a designer involved. [laughs] That's how it developed over time – like a plant grows, that's how I always imagined it."* (C18).

Some participants did not comprehend the concept of usable security, even when explicitly asked about the relationship between security and usability: *"Oh is that [usable security] a phrase? I've never heard the phrase combined, no."* (C17).

Here, security is attributed extraordinary importance, with security measures in place, but usability is regarded as entirely independent from and less important compared to security. Accordingly, the process is not very user-centered: *"Other companies focus on usability. For us, security is important."* (C14). Their customers pay for technical security, but the system possibly being compromised due to users struggling with unusable security mechanisms was not a concern. However, we also found one case where an explicit request for usability from the customer compromised security. A participant described that a customer requested an "uncomplicated authentication process" for a managed mobile device and insisted on using employees' telephone numbers as the "secret" for unlocking the devices. The need for usable security was clearly evident to those involved but despite knowing better, the security company gave in to the request: *"And if you don't, you insist. They tell you 'go away', and they hire another company. Here, there are many companies waiting in the queue just to have a piece of the cake."* (C23).

While usable security was at least considered in the example, in other contexts, it was not attended to during implemen-

tation: *"I'll be honest. I don't think they would even mention the usability of the security. [...]"* (C17).

We generally heard that development teams are under pressure to deliver and that their resources are extremely limited. Good development practices such as clean requirements gathering, unit testing, and mature technical designs often fall by the wayside: *"We need a solution in half a year. So there will be nothing like making a concept first, how everything should look, and making it well usable."* (C16) We discuss issues around resources and budgets in Section IV-G1.

F. Usable Security Decisions

As reported by participants, corporate structures and corporate cultures varied, as did the decision-making processes that affect usable security. For example, early in the development process, customer requirements and suggestions from marketing or other stakeholders often set the direction for the development team. The level of detail provided to the development team varied, which implicitly means developers make decisions relevant for usable security. For example, P11 mentioned:

> *"That is not defined. It's up to us. It's [...] sort of an unspoken rule that if you have login, the developers will implement 'forget password' for you and everything. It's never really defined in the requirements [...]."* (P11)

In development teams without dedicated designers, software developers often perform design activities. While designers, when involved, contributed wireframes and mockups for general design decisions, even with designer involvement, not every exception was included (e. g., the password recovery routine) and important usable security details (e. g., number of allowed login attempts) were missing. Even though designers take some responsibility, the development teams still stay in charge of making usable security decisions. The user interface design of a product often emerges from an interaction between developers and designers who try to interpret user requirements. Designers were actively involved in eleven software development contexts (C1–C6, C8, C11, C15, C19, C22). Three other development teams (C17, C18, C23), reported the ability to consult designers but never or only very rarely made use of this option. In the remaining contexts, no designers were available, leaving developers to make their own decisions. Since not specifying requirements explicitly might impact the distribution of resources within the development team, we asked whether customers requested usable security features. According to our participants, some customers do not care about usable security. Five participants (P8, P13, P17, P18, P22) stated that their customers did not consider usable security important in the past as they assigned little or no budget. For example, P8 said:

> *"Usually, you have like three [login] attempts or something. Sometimes the client doesn't even want that, so you can just keep entering the password until you get it because it's simpler to develop, you know. They don't have to spend too much money on development, I guess."* (P8)

G. Structures that Hinder Usable Security

Below we present obstacles and challenges to usable security based on insights from interviews dealing with concrete development contexts (cf. Section IV-A).

1) Limited Resources: All development organizations have to use their resources carefully. The two key resources overall, budget and time to delivery, are often limited, which has an adverse impact on usable security. The focus lies on delivering functionality on time and budget; security and usability are not specified in detail. Hence, they are assigned a lower priority and fewer resources. It affects usability and security, but usable security – which requires more than one skill set – is not owned by anyone and considered last – if there is any budget and time left.

Budget. Eight participants (P4, P11, P14, P17, P18, P19, P23, P25) said that very limited budgets impacted steps related to usable security in the SDP. For example, usable security cannot be achieved when even basic security cannot be realized within existing budgets, as P18 illustrated:

> *"But in many cases, if the customer doesn't have enough budget for development, you can't set up that kind of security. [...] They have budget for main functionality but not for security or usability."* (P18)

While limited budgets were a common theme throughout the interviews, justifications varied. P17 stated that features must primarily pay for themselves and that usable security is not a way to make money: *"And then you also sometimes discuss whether a usability feature is in proportion to its cost. [...] because in the end, the majority of features have to pay for themselves."* (P17). P23 agrees: *"[...] there are limited resources, right? What's gonna happen is they are testing other features that have broader use, have broader impact [...]"* (P23). Another participant illustrated the prioritization of security awareness over usability: *"Yes, yes. Enormous amount of spending on security. [...] [But on] the usability of it. No, no. That would be... that would be very, very low, considering the usability."* (P4). This is in line with P11 and might cause a security paradigm that cannot achieve usable security as it is skewed towards technical security, neglecting the human factors.

Time. Strongly related to budget, time is another limiting resource. Ten participants (P3, P5, P10, P11, P14, P16–P19, P24) explicitly mentioned general time pressure during their work as well as deadlines. This lack of time inevitably leads to higher-priority tasks being attended to first. Usable security rarely has priority. P18 explicitly described this *"On fast and easy projects [...], you never find that time of detail of security level or usability level."* (P18) as well as P24 who said: *"When you do have to get that thing done in that time, quality suffers, and so some of the things that can suffer is security [...] and then also usability"* (P24). Besides P24, also P3, P5, and P11 each reported that due to time pressure, testing is neglected. All this highlights the time budget's impact on which measures and steps in the SDP will get attention.

Functionality First. A common theme across eight interviews (P3, P10, P15, P18, P20, P24, P25) is the prioritization

of features and functionality – to the detriment of usable security. This *functionality first* principle is also an indicator of the low awareness of usable security's importance: *"against security, against usable security as well. Yes. The resources, the priorities were just mostly to new features"* (P15). Participants described that this is related to time or budget:

> *"There's a point where you will sacrifice a usability or security feature. [...] You cannot touch the main functionality. But you can remove a security feature or a UI part or whatever to remain inside a budget."* (P18)

Following that principle, some participants (P3, P10, P15, P25) told us that the features, including their security, only have to be functional regardless of how usable they are: *"I absolutely don't care what the function looks like. The main thing is that the function is in there, and the customer can use it."* (P3). Furthermore, those participants reported that improving usable security is something they plan to attend to sometime later. But since budgets are always too tight, the resources are never there. While functionality is the key selling point for software and the deciding factor when competing with other companies (P24), the fact that its usable security is never attended to before budgets run out should raise concerns in development companies and their customers.

2) Misconceptions: The analysis of the interviews revealed several misconceptions about usable security and also about usability itself. If software developers do not understand what usable security is, we cannot expect them to carry out the steps needed to deliver it. P9, for example, mentioned a usable authentication method where users overlooked the security but the participant himself did not realize that this is an aspect of usable security: *"But otherwise, I think we really don't have [usable security]. Because the login happens [...] [transparently for users] and what we do there in terms of security things has no influence on how the normal user uses it."* (P9). However, while this would allow accidental or unconscious usable security, other misconceptions would hinder usable security adoption. For example, P21 blamed users for being unable to cope with the security technology and forgetting usernames and passwords: *"[This is] not related to usability, mostly it's related to lack of technology skills. [...] we can't do anything about [authentication]"* (P21).

Another participant did not realize that even well-educated users are more likely to make mistakes when presented with a complex mechanism: *"Then security is my first priority because they are already educated and they know exactly what's going on in the system."* (P1).

As we have already seen, usability may be assigned a low priority because it is not taken seriously: *"So it's kind of, I would say, a healthy mix of best practice and common sense."* (P5). A designer (P19) told us that her expertise is not perceived as such:

> *"But there is a totem pole, and that's you don't always get people to trust your expertise, I guess. There are some people... they trust their opinion over the expertise in the field. What are you gonna do about that?"* (P19)

This was evidenced in P21's opinion that a UX expert was not needed on their team:

"We didn't need any expert to know that... that this type of... of usability... is needed, actually we don't put usability on the... on standards. Usability has to be like this or we can identify as that... we don't know this type... or this... we don't take it from the academic view. We... do deal with usability as a sense. I don't know, as a fifth, fourth, seventh sense, I don't know." (P21)

Similar to other research [9], [53], [54], we found that participants (P1, P6, P21, P22, P24) believed in the trade-off myth for usable security. Around half of our participants stated that usability and security have to be weighed against each other: *"I think the gain of security is worth the slight dip in usability."* (P24) or *"I mean if you want to create something easy to use, you must pay for it at some point. [...] it puts you on a risk of being vulnerable on the security level"* (P21).

Usable security research has provided ample evidence that and how security systems that are difficult to use end up being circumvented or compromised by users. Our results indicate that developers do not yet understand this. P22, P4, and P7 told us that they consciously traded security for more usability: *"In order to make the application more usable, we had to remove all of that... all of that security-related code [...]"* (P4).

In summary, we found fundamental misconceptions about usable security, not least a complete lack of awareness that being usable is a necessary condition for a security mechanism to be effective.

3) Communication Barriers: Implementing usable security requires knowledge and skills in both security and usability. Individuals rarely have knowledge and skills in both, so development organizations have to take steps to ensure a development project has access to those skills. We found several contexts in which expertise was available but not accessible when it was needed. Incorporating other teams or experts in the SDP creates an overhead within organizations. P5 described that developers may have access to experts but do not consult them because it would cost valuable time. Instead, they would only resort to doing so in case of severe problems. This mental barrier may be heightened due to problems in understanding the respective other profession resulting from fractured knowledge. P10, P13, and P23 highlighted that especially designers lack knowledge and understanding of security concepts: *"I think they really put a lot of effort into it already. But what you wonder is if the designer was even able to grasp the front-end developer"* (P10). P22 surmised that designers might not be interested in technical aspects, such as security: *"I didn't meet any designers that like to be involved in technical decisions. I think they feel bored or something."* (P22).

P13 and P23 both told us about conflicts with the security team while fighting for better usability. For example, the usability expert (P13) said:

"We were constantly at odds with the [cryptography team] because [...] we had to basically consistently push back against that cryptography. Well, in their mind, it was against their cryptography because we had to tell them that, like, 'Listen, and you can't do it this way because, like, users will not accept, this is unacceptable, from the user perspective'." (P13)

However, it is essential to combine both in the SDP to actually achieve usable security. Conflicts, therefore, have to be overcome by respecting and understanding each other.

4) Requirements, Guidelines, Compliance: Usability requirements in our participants' organizations were vague and rarely, if ever, written down. In contrast, we noticed that specific security requirements had their origin either in compliance standards or they emerged naturally, as P2, P6, P11, and P15 described: *"Actually, they came pretty naturally."* (P6).

We hoped to obtain concrete requirements for usable security, but as a rule, the processes analyzed tend to indicate that usable security is a requirement, which also arises naturally, but where the basis is not the individuals themselves, but rather the project focus and the corporate culture. To the companies that we have classified as aware it was rather clear to the team members that the project focus was on usable security. However, we have seen at certain companies (cf. Section IV-D) that were aware but failed to implement usable security measures. Neither participants who talked about a specific case nor participants who gave us a more high-level view of their experiences mentioned any compliance standards or guidelines related to usable security. In contrast, we heard about security guidelines or standards more frequently. P19 described a case where she was in a conflict with a decision-maker about removing security questions from the product for multiple reasons. In the end, she lost the fight because she did not have any evidence to convince her superior:

"I didn't have evidence to bring to the table [...]. That would have been a lot easier. [...] Trying to explain usability without a ton of evidence isn't gonna get you very far. I wasn't able to find anything [...] worthwhile. You know, sometimes you win, sometimes you lose." (P19)

H. Structures and Attitudes that Contribute to Usable Security

This section highlights and relates the structures and attitudes that we identified in the above classification as more positively related to the ability to deal with usable security challenges. We consider a holistic understanding of usable security as necessary so that suitable measures can be established. We have seen that some decisions have been made contrary to results of previous usable security research due to fundamental misconceptions (e.g., some development teams still believing the trade-off myth).

1) Communication Pivot: In all contexts showing user-centered characteristics that we classified as being aware of usable security, we found a person who formed the communication bridge between both worlds, usability and security:

"Yeah, so that was one security part of it [...] Especially in the beginning [...], what can the developers actually do on that team, and what are their capabilities in the time frame and the money that we had. So, yeah, the developers were included in the discussion with all the security issues from the beginning." (P14)

Those people with background knowledge in security were not left out but actively involved in the design process. We also observed a strong collaboration between UX experts and security experts in the case of P13. For example, the

designers were actively involved during threat assessment with the security experts. However, the designer or UX expert does not always have to be the driver for usable security. In three of five cases in Section IV-B, a security expert was the pivot for communication about usable security in the software teams. We also observed that the domain knowledge of security and usability was more pronounced in either one of the areas. P13 described that the design team was involved during a threat assessment.

2) Open Attitude and Commitment Towards Usability: We have seen that the development teams of most companies whose processes we classified as having user-centered characteristics were given the time and resources to apply user-centered methods. In most of the contexts, as described in Sections IV-B and IV-C, participants showed an open attitude towards usability and its value for the customer and the business: *"The main and the most important request from the management was: they need an easy-to-use software or app or interface to compete with other competitors"* (P21).

3) Access to Real Users and Feedback: Participants reported two products, one partly (P6), one completely open-source (P2). Both described how they leveraged their user communities to collect feedback. P2 said that they had access to communities all over the world and dedicated contact persons who are in contact with specific user groups via forums and emails. Those communities provided not only comprehensive feedback but were also the source of substantial new requirements. In other contexts which had developed usable security, users were also actively involved in the development. For example, in C5, users were invited to a laboratory to test the software prototype.

4) Knowledge About User-centered Methods and (Usable) Security: In six contexts, we found awareness of the importance of usable security, but the processes were not very user-centered (Section IV-D). Although the lack of resources was often mentioned as a reason for lacking measures (Section IV-G1), participants reported individual user-centered measures, such as personas or prototyping, being used because they were considered rather low-cost. In another case, beta-testing with users and collecting feedback in high-security environments was not possible due to confidentiality reasons: *"No, unfortunately [a beta group is] not possible. For the same reason, we don't get feedback from users."* (C13).

V. Implications

In this section, we discuss the implications of our findings for academia and the software industry. First, we summarize our results and discuss the major factors that enable or prevent usable security in development organizations as well as identify actions that organizations that espouse usable security as a goal could take to enable and support developers.

A. Factors Impacting Usable Security

Lack of Awareness. The development processes our participants described revealed fundamental misconceptions of usable security. Misconceptions are not necessarily shortcomings of them as individuals but can be a shared lack of understanding in software development contexts (Section IV-G2). Many of our participants had an apparent misunderstanding of the connection between usability and security (Section IV-E) or suggested having to sacrifice one to achieve the other (see section IV-G2). Some participants stated usability not to be a specific skill set, but some common knowledge developers would implicitly have. In these development contexts, user-centered methods were neither available nor applied (Section IV-C).

Missing Knowledge of User-Centered Methods. Some interviewees were aware of the potential benefits of making security features usable. However, they lacked the means to implement usable security (Section IV-D). Across all our interviews, only a few user-centered methods were mentioned. Hence, there seems to be a lack of usability security knowledge in many software teams (Section IV-H4). The methods the participants were aware of were often not applied due to a lack of resources (Section IV-G1). Even in the face of limited resources and time constraints, developers could engage with usability for security features by applying established heuristics such as Nielsen's "10 Usability Heuristics for User Interface Design" [42].

Communication Barriers. In some contexts (Section IV-G3), usability expertise was available in the form of specialists, but individual development teams did not consult them. When usability was seen as easy, consulting experts was considered a waste of time. Usable security is an interdisciplinary field where knowledge from both domains must be brought together to craft an effective security mechanism that does not overburden end-users (Section IV-H1).

B. Decision Making

In the case of development teams that miss resources to implement usable security or that have the required skills available but do not apply them, adapting structures and processes can help. Functional requirements are frequently defined during requirements engineering. Depending on the level of requirements details, developers need to make their own decisions. Fine-grained requirements evoke the interaction between end-users as well as the system and reduce the likelihood of inappropriate assumptions made by developers.Some participants reported that more rigorously specified design drafts were created in collaboration with one or more technical experts (cf. Section IV-B). Decisions affecting the usability of the product were made by different stakeholders. We rarely identified active decisions for or against usable security: Decisions affecting usable security are often not debated or a source of conflict; they do not seem important enough to invest much effort. These findings illustrate that many of our participants (see Sections IV-C, IV-E) lack usable security awareness and the consequences of not taking care of it.

C. Recommendations for Industry

Development organizations need to recognize that usable security requires effort and does not come for free with security alone. On one level, making something usable seems easy enough – but it requires basic resources and skills to deliver it. Assuming that developers will take care of a quality attribute that is not clearly specified, and for which nobody has been given responsibility, is wishful thinking – especially when time and resources are always tight. It requires changes to the development process that need to be managed and given resources. In the following section, we present two approaches that could support companies in taking a step further towards usable security.

1) Climbing the Competence Curve: Different types of interventions are needed to help software teams climb the competence curve:

1) Build up awareness for usable security.
2) Impart knowledge to those involved.
3) Integrate and consolidate measures into the company's daily routine.

Changing employees' habits and routines is complex and requires resources. However, it is not about redrawing entire organizational structures, but rather about moving holistically and purposefully one step further towards usable security.

Create Awareness. As we illustrated in the results, awareness of usable security is crucial. While we identified companies that focused on user-centered requirements or needs and implemented respective methods into their processes, they were not aware of the important interplay of usability and security. Instead, they made usable security decisions unconsciously (cf. Sections IV-F, V-B). The fact that decisions about usable security are often made unconsciously reinforces our assumption that awareness is fundamental to start a conversation about usable security and, consequently, to actively make decisions adapting the SDP towards usable security. Nevertheless, awareness alone is not sufficient to change behavior and establish habits that support usable security (cf. Section IV-D).

Understanding Users and Their Context. Some contexts did not integrate measures for usability in their SDP. In order to make a system, and especially its security components, usable, it is crucial to understand the users' needs, capabilities, goals, and most importantly, the context in which they interact with the software. We have seen that all of the contexts from Section IV-B were actively engaged in learning more about their users to build a more usable and secure product. To establish a better user understanding, we recommend sharing this information with other stakeholders, as the participant from C5 did while building an authentication mechanism for a postal delivery company. For a systematic approach, human-computer interaction research has established an extensive knowledge base (e. g., [10]) and toolbox (e. g., [42], [43]) that software professionals can use to improve usable security.

Improve Communication between Experts. In all contexts from Section IV-B, we noticed the gap between usability and security expertise being much smaller than in the other contexts. We attribute this to extensive communication between security and usability experts. Tools such as personas or scenarios, known for decades as communication and modeling tools, are particularly suitable for helping software professionals to understand their users' needs, experiences, behaviors, and goals. Even further, P13 described that they performed threat modeling as a group activity with both security and usability experts.

Shift Usable Security Left. Usable security should become part of the SDP early on. We observed that usable security requirements are currently not specified concretely in almost all contexts. That said, usable security should be explicitly included in requirements engineering and addressed in early discussions with customers and other stakeholders – ideally from the very beginning. The analysis of decision-makers and decision chains (Section V-B) revealed that the requirements which are handed over to development teams, along with their level of detail, substantially determine the extent of available resources and the scope for interpretation of usable security.

Measure and Track Usable Security. The celebration of short-term goals is a common activity to drive change forward [29]. Hence, we recommend tracking usable security progress for both the process and the product. Furthermore, this helps to understand and improve usable security in existing processes. Performing regular usable security measurements as part of the SDP may also prevent a *trial-and-error* approach and save resources. To the best of our knowledge, there are no explicit measures to assess usable security. Instead, we recommend using conventional usability evaluation methods for security features to assess products' usable security. This could include but is not limited to A/B testing, beta tests, or active feedback gathering, as prevalent in the contexts of Section IV-B. There may be other indicators or proxies; in C5 the reduced number of support tickets opened by users after a major redesign was an indicator of improved usable security.

Usable Security Champions. In the context of security, *security champions* are a common role within teams. Similarly, we identified *usable security champions* as having interdisciplinary knowledge in usability and security as well as taking care of usable security. Usable security champions were available in three out of five contexts (C2–C4) in Section IV-B. Those champions are rare, like in C11 (Section IV-D), or could not be found or hired as in C13. Champions were unavailable in the other two contexts that also achieved usable security. Instead, in C5, the designer was the pivot point and initiator of a successful usable-security-driven process – without having security knowledge. However, this needs extensive communication, which can be a higher burden compared to a usable security champion as there will be no communication overhead. Therefore, we conclude that a usable security champion may not be needed when there is enough knowledge and skills for usability and security in the team and the respective members are collaborating.

2) Usable Security Defaults and Tooling: We identified a gap between usability and security for developers and

stakeholders as expertise in both is often unavailable at the same time. A general approach from the security area is the *security by default* principle. We suggest transferring this to *usable security by default*. For example, a web framework might have a module for user authentication. Extending this by usable security principles would allow software creators to easily create a secure and usable software feature out of the box, requiring little or no effort. Consequently, this approach appears suitable for those unaware of usable security or unable to climb the competence curve.

To the best of our knowledge, the current software development ecosystem lacks this support, as there is no tooling, no special documentation, guidelines, or standards for usable security. As a starting point, we propose creating guidelines and checklists for usable security in specific use cases. These then can be incorporated into tools and frameworks. Even if these measures may only be directly related to individual and not all stakeholders (e.g., a framework that only front-end developers use), they consider other contextual factors such as limited resources and knowledge. Nevertheless, the approach is not as holistic as the competence curve. However, this could be a way to systematically address the simpler usable security pitfalls. Overall, we recommend more research to explore this avenue.

D. Recommendations for Human Factors in Security Research

Researching human factors in security has been an active field in the security community since the 1990s. However, we identified a low awareness (Sections IV-C, IV-E) and fundamental misconceptions (Section IV-G2) for usable security among practitioners. While it remains unclear how practitioners learn about usable security, it shines through all our observations that the past decades of usable security research contributions have not received widespread attention in the software industry so far. Hence, there seems to be a huge gap between the status quo in academic research and the adoption by the software industry. However, we are convinced that our community should try to close this gap by better transferring our expertise and knowledge to practitioners who build software. That way, research could increase its impact on society and improve security for millions of users. Furthermore, we discuss the following.

Emphasize Context. Prior research has investigated pitfalls created by individual developers and other software professionals as highlighted in Section II. However, we found that context factors and culture are essential for success with usable security. This highlights the need for further research not limited to individual developers and extends the focus more towards context factors. This includes supporting factors and structures for usable security but also blockers and challenges posed by context factors. Therefore, research should move beyond developer training, education, and support, which is indeed an important approach. However, we also advocate for the research and development interventions that target the production context as a whole and not only single steps from the SDP like the coding by developers. Such a holistic approach should consider the context, including the company and all stakeholders, and processes as important factors influencing usable security. We hypothesize that a holistic approach could also be more sustainable as it does not focus on a single group of individuals, e.g., developers that may leave the company anytime, or technology and defaults changing over time.

Tooling vs. Mindset & Knowledge. Two main approaches can be identified in our research community. One is tooling and technical solutions whose main idea is to support humans, for example, by providing suitable tools [32], [33], [41] to enable non-experts to develop secure software. Beyond that, some approaches aim to "have an impact on the human", e.g., the mindset, knowledge, or skills of stakeholders. Both have advantages and limitations; for example, using a tool is often easier than learning new skills or changing a mindset, but building a mindset for usable security might be more sustainable. We argue that both are orthogonal approaches, and both can be beneficial for different contexts. However, choosing one of those approaches in a specific scenario is not obvious and requires more research.

Develop Measures. To support the adoption of usable security research results in the software industry, development teams need actionable measures. Researchers can support arguments for usable security in teams by developing measurable goals, actions, and interventions. Without those, blockers might be too challenging, as described by P19. We recommend measures to be lightweight in order to to take the limited resources and economic pressures into account (see Section IV-G1) and remain in line with general security research [66], [67].

VI. CONCLUSION

In this work we provide novel insights into how professional software development organizations make usable security using 25 semi-structured interviews with software industry workers from different professions. Our results suggest a high impact of individual as well as contextual software development process factors on software products' usable security, such as the commitment to usable security and knowledge of user-centered methods. Our study highlights the need for more awareness and a holistic understanding of usable security. Based on blockers and structures that contribute to usable security, we propose interventions for industry, such as deliberately strengthening the communication between security and usability experts. To successfully integrate usable security into software products, we recommend that researchers and practitioners take a holistic view on software development, also taking into account contextual factors.

ACKNOWLEDGMENT

REFERENCES

[1] Y. Acar, M. Backes, S. Fahl, S. Garfinkel, D. Kim, M. L. Mazurek, and C. Stransky, "Comparing the Usability of Cryptographic APIs," in *Proc. 38th IEEE Symposium on Security and Privacy (SP'17)*. IEEE, 2017.

[2] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, "You Get Where You're Looking For: The Impact of Information Sources on Code Security," in *Proc. 37th IEEE Symposium on Security and Privacy (SP'16)*. IEEE, 2016.

[3] Y. Acar, S. Fahl, and M. L. Mazurek, "You are Not Your Developer, Either: A Research Agenda for Usable Security and Privacy Research Beyond End Users," in *Proc. 2016 IEEE Secure Development Conference (SecDev'16)*. IEEE, 2016.

[4] A. Adams and M. A. Sasse, "Users are not the enemy: Why users compromise security mechanisms and how to take remedial measures," *Communications of the ACM*, vol. 42, no. 12, pp. 40–46, 1999.

[5] D. Akhawe and A. P. Felt, "Alice in Warningland: A Large-Scale Field Study of Browser Security Warning Effectiveness." in *Proc. 22nd Usenix Security Symposium (SEC'13)*. USENIX Association, 2013.

[6] H. Alshenqeeti, "Interviewing as a Data Collection Method: A Critical Review," *English Linguistics Research*, vol. 3, no. 1, pp. 39–45, 2014.

[7] H. Assal and S. Chiasson, "'Think Secure from the Beginning': A Survey with Software Developers," in *Proc. CHI Conference on Human Factors in Computing Systems (CHI'19)*. ACM, 2019.

[8] H. Assal and S. Chiasson, "Security in the Software Development Lifecycle," in *Proc. 14th Symposium on Usable Privacy and Security (SOUPS'18)*. USENIX Association, 2018.

[9] D. D. Caputo, S. L. Pfleeger, M. A. Sasse, P. Ammann, J. Offutt, and L. Deng, "Barriers to Usable Security? Three Organizational Case Studies," *IEEE Security & Privacy*, vol. 14, no. 5, pp. 22–32, 2016.

[10] A. Cooper, R. Reimann, D. Cronin, and C. Noessel, *About Face: The Essentials of Interaction Design*, 4th ed. John Wiley & Sons, Ltd., 2014.

[11] J. M. Corbin and A. L. Strauss, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 2014.

[12] A. Danilova, A. Naiakshina, J. Deuter, and M. Smith, "Replication: On the Ecological Validity of Online Security Developer Studies: Exploring Deception in a Password-Storage Study with Freelancers," in *Proc. 16th Symposium on Usable Privacy and Security (SOUPS'20)*. USENIX Association, 2020.

[13] E. Derr, S. Bugiel, S. Fahl, Y. Acar, and M. Backes, "Keep me updated: An Empirical Study of Third-Party Library Updatability on Android," in *Proc. 24th ACM Conference on Computer and Communication Security (CCS'17)*. ACM, 2017.

[14] S. Egelman, L. F. Cranor, and J. Hong, "You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings," in *Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI'08)*. ACM, 2008.

[15] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith, "Why Eve and Mallory love Android: An analysis of Android SSL (in)security," in *Proc. 19th ACM Conference on Computer and Communication Security (CCS'12)*. ACM, 2012.

[16] A. P. Felt, A. Ainslie, R. W. Reeder, S. Consolvo, S. Thyagaraja, A. Bettes, H. Harris, and J. Grimes, "Improving SSL Warnings: Comprehension and Adherence," in *Proc. 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15)*. ACM, 2015.

[17] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android Permissions: User Attention, Comprehension, and Behavior," in *Proc. 8th Symposium on Usable Privacy and Security (SOUPS'12)*. ACM, 2012.

[18] M. Golla, M. Wei, J. Hainline, L. Filipe, M. Dürmuth, E. Redmiles, and B. Ur, ""What Was That Site Doing with My Facebook Password?": Designing Password-Reuse Notifications," in *Proc. 25th ACM Conference on Computer and Communication Security (CCS'18)*. ACM, 2018.

[19] 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.

[20] J. Gulliksen, B. Göransson, I. Boivie, S. Blomkvist, J. Persson, and A. Cajander, "Key principles for user-centred systems design," *Behaviour & Information Technology*, vol. 22, no. 6, pp. 397–409, 2003.

[21] M. Gutfleisch, J. H. Klemmer, N. Busch, Y. Acar, M. A. Sasse, and S. Fahl, "Replication Package: "How Does Usable Security (Not) End Up in Software Products? Results From a Qualitative Interview Study"," https://doi.org/10.25835/0089554, 2021, version 1.0.

[22] M. Gutfleisch, M. Peiffer, S. Erk, and M. A. Sasse, "Microsoft Office Macro Warnings: A Design Comedy of Errors with Tragic Security Consequences," in *Proc. 2021 European Symposium on Usable Security (EuroUSEC'21)*. ACM, 2021.

[23] J. Hallett, N. Patnaik, B. Shreeve, and A. Rashid, ""Do this! Do that!, And Nothing will happen": Do specifications lead to securely stored passwords?" in *Proc. 43rd IEEE/ACM International Conference on Software Engineering (ICSE'21)*. IEEE, 2021.

[24] J. M. Haney, M. Theofanos, Y. Acar, and S. S. Prettyman, ""We make it a big deal in the company": Security Mindsets in Organizations that Develop Cryptographic Products," in *Proc. 14th Symposium on Usable Privacy and Security (SOUPS'18)*. USENIX Association, 2018.

[25] N. Huaman, S. Amft, M. Oltrogge, Y. Acar, and S. Fahl, "They Would do Better if They Worked Together: The Case of Interaction Problems Between Password Managers and Websites," in *Proc. 42nd IEEE Symposium on Security and Privacy (SP'21)*. IEEE, 2021.

[26] N. Huaman, B. von Skarczinski, C. Stransky, D. Wermke, Y. Acar, A. Dreißigacker, and S. Fahl, "A Large-Scale Interview Study on Information Security in and Attacks against Small and Medium-sized Enterprises," in *Proc. 30th Usenix Security Symposium (SEC'21)*. USENIX Association, 2021.

[27] L. L. Iacono, H. V. Nguyen, and H. Schmitt, "Usable Security — Results from a Field Study," *i-com*, vol. 15, no. 2, pp. 203–209, 2016.

[28] L. Kocksch, M. Korn, A. Poller, and S. Wagenknecht, "Caring for IT Security: Accountabilities, Moralities, and Oscillations in IT Security Practices," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, pp. 92:1–92:20, 2018.

[29] J. P. Kotter, *Leading Change*. Harvard Business Review Press, 2012.

[30] K. Krombholz, K. Busse, K. Pfeffer, M. Smith, and E. von Zezschwitz, ""If HTTPS Were Secure, I Wouldn't Need 2FA" - End User and Administrator Mental Models of HTTPS," in *Proc. 40th IEEE Symposium on Security and Privacy (SP'19)*. IEEE, 2019.

[31] K. Krombholz, W. Mayer, M. Schmiedecker, and E. Weippl, ""I Have No Idea What I'm Doing" - On the Usability of Deploying HTTPS," in *Proc. 26th Usenix Security Symposium (SEC'17)*. USENIX Association, 2017.

[32] S. Krüger, S. Nadi, M. Reif, K. Ali, M. Mezini, E. Bodden, F. Göpfert, F. Günther, C. Weinert, D. Demmler, and R. Kamath, "CogniCrypt: Supporting Developers in Using Cryptography," in *Proc. 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE'17)*. IEEE Press, 2017.

[33] S. Krüger, K. Ali, and E. Bodden, "CogniCrypt$_{GEN}$: Generating Code for the Secure Usage of Crypto APIs," in *Proc. 18th ACM/IEEE International Symposium on Code Generation and Optimization (CGO'20)*. ACM, 2020.

[34] F. Li, L. Rogers, A. Mathur, N. Malkin, and M. Chetty, "Keepers of the Machines: Examining How System Administrators Manage Software Updates For Multiple Machines," in *Proc. 15th Symposium on Usable Privacy and Security (SOUPS'19)*. USENIX Association, 2019.

[35] T. Lopez, H. Sharp, T. Tun, A. K. Bandara, M. Levine, and B. Nuseibeh, ""Hopefully We Are Mostly Secure": Views on Secure Code in Professional Practice," in *Proc. 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE'19)*. IEEE Press, 2019.

[36] S. Nadi, S. Krüger, M. Mezini, and E. Bodden, ""Jumping Through Hoops": Why do Java Developers Struggle With Cryptography APIs?" in *Proc. 38th IEEE/ACM International Conference on Software Engineering (ICSE'16)*. ACM, 2016.

[37] A. Naiakshina, A. Danilova, E. Gerlitz, and M. Smith, "On Conducting Security Developer Studies with CS Students: Examining a Password-Storage Study with CS Students, Freelancers, and Company Developers," in *Proc. CHI Conference on Human Factors in Computing Systems (CHI'20)*. ACM, 2020.

[38] A. Naiakshina, A. Danilova, E. Gerlitz, E. von Zezschwitz, and M. Smith, ""If You Want, I Can Store the Encrypted Password": A Password-Storage Field Study with Freelance Developers," in *Proc. CHI Conference on Human Factors in Computing Systems (CHI'19)*. ACM, 2019.

[39] A. Naiakshina, A. Danilova, C. Tiefenau, M. Herzog, S. Dechand, and M. Smith, "Why Do Developers Get Password Storage Wrong?: A Qualitative Usability Study," in *Proc. 24th ACM Conference on Computer and Communication Security (CCS'17)*. ACM, 2017.

[40] A. Naiakshina, A. Danilova, C. Tiefenau, and M. Smith, "Deception Task Design in Developer Password Studies: Exploring a Student Sample," in *Proc. 14th Symposium on Usable Privacy and Security (SOUPS'18)*. USENIX Association, 2018.

[41] D. C. Nguyen, D. Wermke, Y. Acar, M. Backes, C. Weir, and S. Fahl, "A Stitch in Time: Supporting Android Developers in Writing Secure Code," in *Proc. 24th ACM Conference on Computer and Communication Security (CCS'17)*. ACM, 2017.

[42] J. Nielsen, "Enhancing the Explanatory Power of Usability Heuristics," in *Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI'94)*. ACM, 1994.

[43] J. Nielsen, *Usability engineering*. Morgan Kaufmann Publishers, 1993.

[44] M. Oltrogge, Y. Acar, S. Dechand, M. Smith, and S. Fahl, "To Pin or Not to Pin—Helping App Developers Bullet Proof Their TLS Connections," in *Proc. 24th Usenix Security Symposium (SEC'15)*. USENIX Association, 2015.

[45] M. Oltrogge, N. Huaman, S. Amft, Y. Acar, M. Backes, and S. Fahl, "Why Eve and Mallory Still Love Android: Revisiting TLS (In)Security in Android Applications," in *Proc. 30th Usenix Security Symposium (SEC'21)*. USENIX Association, 2021.

[46] N. Patnaik, J. Hallett, and A. Rashid, "Usability Smells: An Analysis of Developers' Struggle With Crypto Libraries," in *Proc. 15th Symposium on Usable Privacy and Security (SOUPS'19)*. USENIX Association, 2019.

[47] S. Pearman, J. Thomas, P. E. Naeini, H. Habib, L. Bauer, N. Christin, L. F. Cranor, S. Egelman, and A. Forget, "Let's Go in for a Closer Look: Observing Passwords in Their Natural Habitat," in *Proc. 24th ACM Conference on Computer and Communication Security (CCS'17)*. ACM, 2017.

[48] O. Pieczul, S. Foley, and M. E. Zurko, "Developer-centered security and the symmetry of ignorance," in *Proc. 2017 New Security Paradigms Workshop (NSPW'17)*. ACM, 2017.

[49] 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 *Proc. 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17)*. ACM, 2017.

[50] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 8th ed. McGraw-Hill Education, 2015.

[51] A. Ruef, M. Hicks, J. Parker, D. Levin, M. L. Mazurek, and P. Mardziel, "Build It, Break It, Fix It: Contesting Secure Development," in *Proc. 23nd ACM Conference on Computer and Communication Security (CCS'16)*. ACM, 2016.

[52] A. Sasse, "Scaring and Bullying People into Security Won't Work," *IEEE Security & Privacy*, vol. 13, no. 3, pp. 80–83, 2015.

[53] M. A. Sasse and M. Smith, "The Security-Usability Tradeoff Myth [Guest editors' introduction]," *IEEE Security & Privacy*, vol. 14, no. 5, pp. 11–13, 2016.

[54] M. A. Sasse, M. Smith, C. Herley, H. Lipford, and K. Vaniea, "Debunking Security-Usability Tradeoff Myths," *IEEE Security & Privacy*, vol. 14, no. 5, pp. 33–39, 2016.

[55] B. Schneier, "Stop Trying to Fix the User," *IEEE Security & Privacy*, vol. 14, no. 5, pp. 96–96, 2016.

[56] A. Shostack and M. E. Zurko, "Secure Development Tools and Techniques Need More Research That Will Increase Their Impact and Effectiveness in Practice," *Communications of the ACM*, vol. 63, no. 5, pp. 39–41, 2020.

[57] B. Shreeve, J. Hallett, M. Edwards, M. Ramokapane, R. Atkins, and A. Rashid, "The best laid plans or lack thereof: Security decision-making of different stakeholder groups," *IEEE Transactions on Software Engineering*, 2020.

[58] B. Shreeve, J. Hallett, M. Edwards, P. Anthonysamy, S. Frey, and A. Rashid, ""So If Mr Blue Head Here Clicks the Link..." Risk Thinking in Cyber Security Decision Making," *ACM Transactions on Privacy and Security*, vol. 24, no. 1, 2020.

[59] R. Stevens, D. Votipka, E. M. Redmiles, C. Ahern, P. Sweeney, and M. L. Mazurek, "The Battle for New York: A Case Study of Applied Digital Threat Modeling at the Enterprise Level," in *Proc. 27th Usenix Security Symposium (SEC'18)*. USENIX Association, 2018.

[60] C. Stransky, O. Wiese, V. Roth, Y. Acar, and S. Fahl, "27 Years and 81 Million Opportunities Later: Investigating the Use of Email Encryption for an Entire University," in *Proc. 43rd IEEE Symposium on Security and Privacy (SP'22)*. IEEE, 2022.

[61] T. W. Thomas, M. Tabassum, B. Chu, and H. Lipford, "Security During Application Development: An Application Security Expert Perspective," in *Proc. CHI Conference on Human Factors in Computing Systems (CHI'18)*. ACM, 2018.

[62] C. Tiefenau, M. Häring, K. Krombholz, and E. von Zezschwitz, "Security, Availability, and Multiple Information Sources: Exploring Update Behavior of System Administrators," in *Proc. 16th Symposium on Usable Privacy and Security (SOUPS'20)*. USENIX Association, 2020.

[63] D. van der Linden, P. Anthonysamy, B. Nuseibeh, T. T. Tun, M. Petre, M. Levine, J. Towse, and A. Rashid, "Schrödinger's Security: Opening the Box on App Developers' Security Rationale," in *Proc. 42nd IEEE/ACM International Conference on Software Engineering (ICSE'20)*. ACM, 2020.

[64] D. Votipka, K. R. Fulton, J. Parker, M. Hou, M. L. Mazurek, and M. Hicks, "Understanding security mistakes developers make: Qualitative analysis from Build It, Break It, Fix It," in *Proc. 29th Usenix Security Symposium (SEC'20)*. USENIX Association, 2020.

[65] D. Votipka, R. Stevens, E. Redmiles, J. Hu, and M. Mazurek, "Hackers vs. Testers: A Comparison of Software Vulnerability Discovery Processes," in *Proc. 39th IEEE Symposium on Security and Privacy (SP'18)*. IEEE, 2018.

[66] C. Weir, I. Becker, J. Noble, L. Blair, M. A. Sasse, and A. Rashid, "Interventions for long-term software security: Creating a lightweight program of assurance techniques for developers," *Software: Practice and Experience*, vol. 50, no. 3, pp. 275–298, 2020.

[67] C. Weir, L. Blair, I. Becker, J. Noble, A. Sasse, and A. Rashid, "Interventions for Software Security: Creating a Lightweight Program of Assurance Techniques for Developers," in *Proc. 41st IEEE/ACM International Conference on Software Engineering (ICSE'19)*. IEEE, 2019.

[68] C. Weir, B. Hermann, and S. Fahl, "From Needs to Actions to Secure Apps? The Effect of Requirements and Developer Practices on App Security," in *Proc. 29th Usenix Security Symposium (SEC'20)*. USENIX Association, 2020.

[69] A. Whitten and J. D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," in *Proc. 8th Usenix Security Symposium (SEC'99)*. USENIX Association, 1999.

[70] K. Yakdan, S. Dechand, E. Gerhards-Padilla, and M. Smith, "Helping Johnny to Analyze Malware: A Usability-Optimized Decompiler and Malware Analysis User Study," in *Proc. 37th IEEE Symposium on Security and Privacy (SP'16)*. IEEE, 2016.

[71] M. E. Zurko and R. T. Simon, "User-Centered Security," in *Proc. 1996 New Security Paradigms Workshop (NSPW'96)*. ACM, 1996.

APPENDIX A

PARTICIPANTS

Table III gives an overview of our participants. We did not include sensitive information such as age and gender on a per-participant basis to protect our participants' privacy. However, this data is reported in aggregated form in Table I. We purposely do not link contexts to individual interviews for participant and company privacy, as some participants were recruited from our professional network. Identifying product, company size, experience, etc. could help re-identify those who volunteered to participate in our study. We think this should not impact the paper: Our coding and interpretation included our contextual knowledge; thus, our results are informed by this link. For privacy reasons, we omit the link in Tables II and III. Consequently, when we quote participants where their context is relevant, we identify them with the context; whenever their personal experience is most relevant, we quote them with their participant ID.

APPENDIX B

REPLICATION PACKAGE

To improve the reproducibility of our work and allow easy access for meta-research, we publish a replication package [21]

TABLE III
OVERVIEW OVER OUR PARTICIPANTS.

#	Highest Degree	Country[1]	Current Job Status	Current Job Title	Industry Experience [years]
P1	Master	IN	Self-employed/Freelancer	Consultant, CTO, Full Stack Architect	16
P2	High School	DE	Employed, full-time	Software Design and Development	30
P3	Bachelor	DE	Employed, full-time	Senior Full-Stack Developer	14
P4	College	US	Employed, full-time	Software Engineer	3
P5	Master	DE	Prefer not to say/NA	Senior Software Developer	6
P6	—	DE	Employed, full-time	Founder	—
P7	Master	LB	Employed, full-time	Web Developer	10
P8	Master	BA	Self-employed/Freelancer	Embedded Systems Developer	4
P9	Vocational Degree	DE	Employed, full-time	Programmer	5
P10	College	US	Employed, full-time	Software Engineer	6
P11	Bachelor	IN	Employed, full-time	Lead Architect	5
P12	Doctorate	DE	Employed, full-time	COO	8
P13	Graduate School	DE	Employed, full-time	Founder	10
P14	Bachelor	CA	Employed, part-time	UX Research, Design Specialist	4
P15	Master	LU	Prefer not to say/NA	Software Developer	3
P16	Bachelor	UA	Employed, full-time	Tech Lead	18
P17	Master	DE	Employed, full-time	Senior Software Engineer	3
P18	Master	VE	Prefer not to say/NA	Software Architect	29
P19	Bachelor	US	Self-employed/Freelancer	UX Designer	6
P20	Doctorate	DE	Employed, full-time	Head of Development	13
P21	Bachelor	LB	Employed, full-time	Software Developer	10
P22	Bachelor	AE	Self-employed/Freelancer	Chief Software Architect	12
P23	Master	US	Employed, full-time	Technical Staff	30
P24	Bachelor	GB-SCT	Employed, full-time	Principal Front End Engineer	23
P25[2]	Doctorate	DE	Employed, full-time	Lead Engineer and Architect	—

[1] ISO 3166-1 encoded.
[2] Interview with three participants, reported only main participant.

containing the following documents: (1) The pre-questionnaire for demographic and quantitative questions; (2) The interview guide with the main questions and follow-up prompts for the semi-structured interviews; (3) The materials we used to contact and recruit participants; (4) The consent forms; (5) A table of the software used in this study; (6) The operationalized codebook with codes contributing to this paper. The replication package is available at https://doi.org/10.25835/0089554.

APPENDIX C
PRE-QUESTIONNAIRE

Usability

Q1.1. Which of the following statements best describes your usability experience?

- ○ No experience with usability
- ○ Little experience with usability
- ○ Some experience with usability
- ○ Considerable experience with usability
- ○ Prefer not to answer

Q1.2. Was usability part of your training or studies?

- ○ Yes
- ○ No
- ○ Prefer not to answer
- ○ Other (please specify): _____

Q1.3. During your professional activity have you participated in any usability course or training?

- ○ Yes
- ○ No
- ○ Prefer not to answer

Q1.4. Which usability methods have you applied? [Free text field]

Q1.5. Who do you think is responsible for the usability of a software product? [Free text field]

Security

Q2.1. Which of the following statements best describes your security experience?

- ○ No experience with security
- ○ Little experience with security
- ○ Some experience with security
- ○ Considerable experience with security
- ○ Prefer not to answer

Q2.2. Was security part of your training or studies?

- ○ Yes
- ○ No
- ○ Prefer not to answer
- ○ Other (please specify): _____

Q2.3. During your professional activity have you participated in any security course or training?

- ○ Yes
- ○ No
- ○ Prefer not to answer

Q2.4. Who do you think is responsible for the security of a software product? [Free text field]

Q2.5. How many of your recent projects had security requirements? [Numerical field]

Experience

Q3.1. How old are you? [Numerical field]

Q3.2. What is your gender?

- ○ Male
- ○ Female
- ○ Non-binary
- ○ Prefer not to answer
- ○ Other (please specify): _____

Code	Description	Example Quote
Individual (*)	Codes that describe the participant individually, not the context or company.	—
Usability	General statements and attitudes towards usability.	*Yes, that UX and usability is quite the extensive and broad topics but from my experience basically, I just divide it into two simple classifications.*
Security	General statements and attitudes towards security.	*When it comes to the security it's something that should be architected in from the very beginning*
Interplay Usability & Security	General statements and attitudes towards the relationship and interplay of security and usability, including usable security.	*I think it's equal. They go hand in hand. Security as is very important but also usability.*
Context Information (*)	Context specific information (company, project, and product, including processes).	—
Product	Description of the software product, including functionality, use cases, customers, requirements, etc.	*We're a little bit special with the password manager product.*
Staff & Team	Descriptions of the team working on the software, including roles, skills, expertise, and size. Covers external and internal team members.	*And the CEO of course… he knows what he wants. He knows the users better than we do.*
Budget & Resources	Resources influencing security and/or usability. This includes time and money but also statements on limited resources or time pressure.	*Yes, so as I said, designers explicitly don't, because it's not important enough for that, and no one would want to pay him.*
Responsibility	Responsibility for security, usability, and usable security, but also related steps in the SDP. This includes decision-makers.	*The, the people in the design team, they had little voice or no voice at all.*
Importance for Company & Motivation	Motivators and focus in the context's company. Including but not limited to usability, security, and usable security.	*So, our goal was to make it as efficient and usable and quick as possible for the user.*
Guidelines & Standards	Usage and availability of guidelines and standards in the SDP, also internal/context specific. Also includes checklists and laws.	*Yes, I mean, of course, we sometimes look into the material design guidelines.*
SDP (*)	In-depth descriptions of the software development processes.	—
Communication & Modelling	Modeling of products, as well as everything belonging to the communication phase (e. g., requirement elicitation). This includes requirement modeling, architecture modeling and design modeling.	*We have a user experience designer. He made these suggestions for a redesign based on the suggestions we had already collected as a team.*
Planning	Information on the planning phase and project management. Also includes general procedures and processes spanning the whole SDP or other SDP steps as well as techniques (e. g., agile).	*Yes, all three things I talked to you about were agile [makes air quotes].*
Construction	General and usability/security specific measures and descriptions of the construction phase, including implementation and testing.	*The static analysis tool takes care of that.*
Deployment	Measures and process steps that take place after construction. Includes maintenance, bug handling and fixing, customer support, as well as beta/alpha testing. Also includes everything belonging to operations.	*We also did a beta phase there.*
General (*)	General information and statements not related to a specific context.	—
Misconceptions	Misconceptions about security, usability, and usable security.	*Security over usability, they were like 'no no no we gotta have this time out' and I was just like 'accessibility issues and other things' and they like 'no no no we need to do this'.*
Staff & Team	(same as above, but covering non context specific statements)	*Bigger companies… Mid sized companies normally have a security guy.*
Responsibility	(same as above, but covering non context specific statements)	*You know it doesn't come to much from the product team or the users […]*
Importance for Company & Motivation	(same as above, but covering non context specific statements)	*I cant think of a time. […] they've mentioned the usability of the security.*
Budget & Resources	(same as above, but covering non context specific statements)	*And the smaller companies have money too […]*
Guidelines & Standards	(same as above, but covering non context specific statements)	*But the biggest companies normally are more tied to the plan, following guidelines, following […]*
SDP – General Thoughts (*)	(Including the same subcodes as the above SDP code)	—

Q3.3. What is the highest level of school you have completed or the highest degree you have received?

- ○ Less than high school / GCSE or equivalent
- ○ High school or equivalent / A level or equivalent
- ○ Some college, currently enrolled in college, or two-year associate's degree, completed part of a higher education course, or currently enrolled
- ○ Vocational degree
- ○ Bachelor's degree
- ○ Some graduate school, or currently enrolled in graduate school
- ○ Master's or professional degree
- ○ Doctorate degree
- ○ Other (please specify): _____

Q3.4. In which country do you live? [Free text field]

Q3.5. What is your current employment status?

- ○ Employed full-time
- ○ Employed part-time
- ○ Prefer not to answer
- ○ Other (please specify): _____

Q3.6. What is your current job title? [Free text field]

Q3.7. How many years have you been in the software industry? [Numerical field]

Q3.8. How many different products have you worked on in your professional career? [Numerical field]

APPENDIX D
INTERVIEW GUIDE

We include only the English version which is identical in content with the German version. A version with follow-up questions can be found in the replication package (Appendix B).

Individually

1) Usability: What is usability for you in a software product?
2) Security: What role does security play in software development for you?
3) Usable Security: How do you think usability and security are related to each other?

Case Example

Usability.

1) Have you ever implemented something that affected the usability of the product? (e. g., anything that changes the UI)
2) Can you tell me a little bit about the background of the product and feature you have implemented?
3) How did you make your decision about usability? Can you walk me through the process?
4) Did the specification you were given cover usability aspects?
5) Were you given any specific usability criteria to meet (when would the software be "usable enough")?
6) Who did you assign usability tasks to? Did you assign the role of "usability champion" to any team member?
7) Did you consult other team members, or other usability resources in the company, at any stage?
8) Did you consult any external usability resources – literature, design guidelines – at any point?
9) What were the specific usability goals/requirements for this project?
10) Were any usability assessments (e. g., heuristic evaluation, user test) carried out during the project?
11) Did you know exactly who the users are and how they behave with the product?
12) How do you handle usability bugs?
13) Was usability directly or indirectly part of your goals within your contract?
14) What role does usability play for your team?
15) What usability knowledge and skills does the team have?
16) How many of you previously worked on a project with usability requirements?
17) How many of you had usability training in this company, or in a previous position?
18) How important was it for the company that software delivered is usable? What are the consequences if it is not?
19) How well do you think this company delivers usability, compared to other companies you know?
20) How does the company deliver usability?
21) How does the organization define usability?
22) Can you give me some more Background information to this company?

Security.

1) Have you ever implemented something that affected the security of the product? (e. g., login field, network interface, some API which used cryptographic functions)
2) Can you tell me a little bit about the background of the product and feature you have implemented?
3) How did you make your decision about security? Can you walk me through the process?

4) Did the specification you were given cover security aspects? For instance, was a risk/threat analysis provided? Were security mechanisms specified?
5) Were you given any specific security criteria to meet (when would the software be "usable enough")?
6) Who did you assign security tasks to? Did you assign the role of "security champion" to any team member?
7) Did you consult other team members, or other security resources in the company, at any stage?
8) What were the specific security goals/requirements for this project?
9) Were any security assessments (e. g., code walkthrough, pen testing) carried out during the project?
10) How do you select an API for security purposes?
11) How do you deal with security bugs?
12) Was security directly or indirectly part of your goals within your contract?
13) How do members of the team define security?
14) What security knowledge and skills did the team have?
15) How many of you previously worked on a project with significant security requirements?
16) How many of you had security training in this company, or in a previous position?
17) How important is it for the company that software delivered is secure? What are the consequences if it is not?
18) How does the organization deliver secure software?
19) How well do you think this company delivers usability, compared to other companies you know?
20) How does the organization define security?
21) How well does the software deliver by the organization meet security goals, compared to other organizations?
22) Can you give me some more background information of this company?

Usable Security.

1) Did you personally have to deal with conflicting usability and security requirements at any stage?
2) Can you tell me a little bit about the background of the product and conflict you dealt with?
3) When you look at all your projects in the past, how many of them had bad usable security requirements?
4) Did you consult any internal or external resources – colleagues, developer forum – to help you solve the problem? If so, what information was helpful to you?
5) Do you follow a particular software development process? If so, how would you describe it? And how easy is it to deliver usable security as part of this process?
6) Did you consult any internal or external resources – colleagues, external experts, developer forum – to help you achieve usable security in the project? If so, what information was helpful to you?
7) Were there specific goals for usability of the security mechanisms in the software?
8) How important is it for the company to deliver usable security? What are the consequences if it is not?
9) What does the company do to ensure it does deliver usable security? Who in the company has responsibility for this?