Code Reviewing as Methodology for Online Security Studies with Developers – A Case Study with Freelancers on Password Storage

Anastasia Danilova, Alena Naiakshina, and Anna Rasgauski, *University of Bonn;* Matthew Smith, *University of Bonn, FKIE Fraunhofer*

https://www.usenix.org/conference/soups2021/presentation/danilova

This paper is included in the Proceedings of the Seventeenth Symposium on Usable Privacy and Security.

August 9–10, 2021

978-1-939133-25-0

Code Reviewing as Methodology for Online Security Studies with Developers – A Case Study with Freelancers on Password Storage

Anastasia Danilova
University of Bonn
danilova@cs.uni-bonn.de

Alena Naiakshina
University of Bonn
naiakshi@cs.uni-bonn.de

Anna Rasgauski
University of Bonn
rasgausk@cs.uni-bonn.de

Matthew Smith
University of Bonn, FKIE Fraunhofer
smith@cs.uni-bonn.de

Abstract

While ample experience with end-user studies exists, only little is known about studies with software developers in a security context. In past research investigating the security behavior of software developers, participants often had to complete programming tasks. However, programming tasks require a large amount of participants' time and effort, which often results in high costs and small sample sizes. We therefore tested a new methodology for security developer studies. In an online study, we asked freelance developers to write code reviews for password-storage code snippets. Since developers often tend to focus on functionality first and security later, similar to end users, we prompted half the participants for security. Although the freelancers indicated that they feel responsible for security, our results showed that they did not focus on security in their code reviews, even in a security-critical task such as password-storage. Almost half the participants wanted to release the insecure code snippets. However, we found that security prompting had a significant effect on the security awareness. To provide further insight into this line of work, we compared our results with similar password-storage studies containing programming tasks, and discussed code reviewing as a new methodology for future security research with developers.

1 Introduction

Code reviewing is a technique applied at the end of the Software Development Life Cycle (SDLC), used as one of the final steps by software developers to ensure programming code quality before software release. Thus, software developers change their perspective from a code creator to a code inspector, which might affect their security awareness. While knowledge on code reviewing in the field of software engineering exists [9, 19, 32], only little is known about this methodology within a security context [13]. Acar et al. [4] called for more studies on developers' security behavior and on the study methodology of security developers. While most of the previous work within this context includes surveys (e.g., [7, 11, 21, 28, 37]), interviews (e.g., [6, 11, 18, 21, 24, 36–38]) or programming tasks (e.g., [2, 3, 17, 20, 22, 23, 25, 25–27, 30, 39–42]), we explored code reviewing as a promising methodology for developer security studies.

Conducting security studies with software developers can be difficult due to recruitment and study compensation challenges [3–5, 10, 12, 20–22, 25, 31, 42]. Programming tasks can also often take more time than professionals can afford. Naiakshina et al. [22–25] conducted a number of studies where computer science (CS) students, freelance developers and software developers from companies were required to complete the registration functionality in a web application. The authors investigated participants' security behavior with a focus on the storage of user passwords in a database. According to the study design and participants' feedback, the study lasted around 8 hours. Recruiting a high number of employed developers who had time outside of their normal working hours for a one-day study was reported to be extremely difficult. Thus, their sample size is not as large as they would have wished.

Therefore, researchers often tend to design programming tasks in such a way that software developers only need to solve small and short tasks (e.g., [2, 3, 5, 33]). For example, Acar et al. [5] conducted a security developer study with GitHub users and provided them with programming tasks which were "short enough so that the uncompensated participants would be likely to complete them before losing interest, but still complex enough to be interesting and allow for some mistakes." While Acar et al. did not mention security or privacy in the recruitment message at all, Naiakshina et al. tested if

USENIX Symposium on Usable Privacy and Security (SOUPS) 2021.
August 8–10, 2021, Vancouver, B.C., Canada.

explicitly asking participants for secure password storage (security prompting) would affect their solutions. They found that security prompting had a significant effect on participants solutions. Additionally, participants tended to concentrate on the functionality of the software first, before working on security aspects [22–24].

To provide deeper insights into this research field, we tested code reviewing as a promising methodology for security studies with developers. Instead of asking developers to program a piece of code, we showed them functional code snippets and asked them to write code reviews about the snippets. We based our code snippets on the participants' submissions from the previous freelancer study from Naiakshina et al. [23] and also recruited freelance developers. This allowed us to compare the results of the code review task with the findings of the previous study containing a programming task. While we offer insights on freelancers' behavior in code reviewing tasks on a primary level, we also discuss code reviewing as a methodology for developer security studies on a meta-level. Our main research questions are as follows:

RQ1: How do developers behave when reviewing code in a security-critical task such as password storage? We were interested to find which criteria the developers mostly base their reviews on in security-critical code, and whether they would be able to indicate the security issue, even when being presented with distraction tasks not related to security. Additionally, how detailed would their security problem description and suggestions for improvement be? Would they suggest to release the code even though it contains security-critical issues?

RQ2: Which factors have an influence on developers' security awareness? In particular, we investigated whether prompting, programming experience or code snippets with different password-storage issues (plain text, Base64, MD5) have an influence on whether developers find the security issues.

RQ3: How much time do developers dedicate to security and do they feel responsible for security in a code review? We asked the freelance developers to indicate how much time they spend with security in code reviews and whether they feel responsible for security. These results were compared to their self-reported responsibility and time spent on security in programming tasks.

RQ4: Comparing the results of a programming and a code reviewing task on password storage, which methodological implications can we conclude? To provide insights for security studies with developers, we discuss the advantages and disadvantages of code reviewing as a study methodology compared to programming tasks.

2 Related Work

Our related work section is divided into two parts. First, we discuss related work in the area of developer, security and password-storage studies. We then take a closer look at code-reviewing studies.

2.1 Developer and Password Studies

A qualitative study from Naiakshina et al. [24] in 2017 investigated the password-storage implementation behavior of 20 CS students. The participants were given a scenario where they were told they were working in a team to implement the registration functionality of a social networking platform. Half of the students were prompted beforehand and told to implement a secure solution. None of the non-prompted students implemented a secure solution. As often found with end users, there was also a tendency with developers to offer functionality over security. This study was extended by the authors in 2018, by inviting 20 additional CS students [25]. The authors acknowledged that they had challenges to recruit enough participants for the study. Out of a pool of 1600 CS students at their university, only 40 participated in the study. The exploratory quantitative study supported the findings of the qualitative study. However, CS students stated that they would have implemented a secure solution had the task been for a real company.

To find out whether the previous results were a study artifact, in 2019 Naiakshina et al. [23] conducted a similar study, but this time with 43 freelancers. Hired through Freelancer.com, the participants were asked to implement the registration functionality for a fictitious online sports-picture sharing platform. Here it was also found that prompting has an effect on the security of the end solutions. 15 of the 22 non-prompted, and even 3 of the 21 prompted freelancers received security requests after submitting insecure solutions [23]. Unlike the previous study, the participants believed they were working for a real company. This did not appear to have an impact on the security of the end solution. The same study was also conducted with 36 professional developers from diverse companies in 2020 [22] and again, prompting had an effect on the security of participants' submissions.

Another study on password-storage was conducted by Wijayarathna et al. [40]. Their study was similar to the one described in the previous papers. To explore the usability of the SCrypt password hashing functionality of Bouncycastle, the authors conducted a 2-hour study with 10 programmers and reported 63 usability issues with the Bouncycastle API. A further study from Wijayarathna et al. [39] investigated how security responsible programmers feel when writing code. The participants were expected to complete four programming tasks including a password storage task. They found that developers know they are responsible for security, but often have a difficult time implementing security measures.

A study investigating having GitHub members for developer studies was conducted by Acar et al. [5]. 307 participants completed three small python programming tasks and then filled out a survey, all without payment. One of the tasks was to store login credentials in a database. The researchers set out to explore how well GitHub users can replace IT-professionals in developer studies, who often do not have time for studies outside of their working schedules and who often have an hourly rate much higher than that offered for taking part in studies. They found that whether a participant is a professional or a student had no significant effect on the security perception of the participant or on the functionality or security of the solution implemented. Experience did however have an effect: each year of added python experience increased the chance of a secure solution by 5%.

Tahei et al. [34] reviewed literature looking at security from a developer perspective. They found there was generally a lack of research in developer-centered-security, and that security needed to become of higher business value. As found in the previously mentioned studies, security knowledge among software developers is often lacking and security is often a secondary requirement.

2.2 Code-reviewing Studies

Baum et al. [9] looked at code-reviewing practices in 19 firms, 11 of which regularly conduct some kind of code review. They found that some firms use multiple reviewers, particularly when making large changes to code. Aside from being used to improve the quality of the code and find defects, the researchers found that code reviews are also conducted to enable a learning process of the coder. Furthermore, it was found that some firms do not conduct code reviews: developers can often feel attacked by code reviews and need time to adjust to the new method. Bacchelli et al. [8] supported the finding that code reviews are often used as a learning process in firms. They also found that “finding defects” in code was a driving factor for conducting code reviews. In multiple papers the term “finding defects” included security issues, but mostly included other kinds of defects [8, 9, 29].

Using the largely open source Mozilla project, Kononenko et al. [19] studied the quality of code-reviews. 54% of bugs were not found during the code-review. The authors found a positive correlation between reviewing experience and number of bugs found. Coding experience did not have an impact on the number of bugs found. They found larger patches and an increased number of files to be reviewed increased the chance of introducing “buggy” patches. The use of super reviewers, a reviewer who is a highly experienced developer, decreases the probability of introducing patches with bugs in them.

What effect availability bias has on code reviews was investigated by Spadini et al. [32]. Using a browser-based reviewing tool chosen by the researchers, the participants were expected to review a code change. The change contained three errors: two errors were of the same type and were bugs that reviewers do not usually look for, whereas as the third was of another type. Half of the participants were non-primed and received a code change where none of the bugs had been commented on, whereas as the other half were primed and received a code change where one of the errors had been commented on by another reviewer. The researchers found a correlation between priming the participants and the participants finding the other error of this type. 80% of participants mentioned they were influenced by the comments in the code. The non-primed bug was found by both groups at a similar rate.

A study investigating the optimal number of code-reviewers for a given task was conducted by Edmunson at al. [13]. The participants were web developers all with various backgrounds and security experience. They were presented with and expected to review an existing open-source project, which had some security issues. Edmunson et al. found a correlation between the number of correct vulnerabilities found and the number of false vulnerabilities found. Having more than 15 reviewers did not bring any further improvement. Interestingly, the researchers found a negative correlation between the number of years of experience and the number of correct vulnerabilities found. Our study differs from Edmunson et al.’s study in several aspects. First, while Edmunson at al.’s participants were informed about security issues within the web application they had to review, we did not mention that our code had issues at all. Additionally, half of our participants were advised to consider security for the code review, while the other half were not. Second, in addition to the security issues, we also added general issues within the code, so we were able to see which issues participants are more aware of, if they recognized them at all. Third, we showed participants one simple code snippet in the context of secure password storage, whereas Edmunson et al. investigated Cross-Site Scripting and SQL injection vulnerabilities within an entire web application project.

Most work using code reviewing as a study methodology was conducted in the field of software engineering. It is unclear yet, whether the findings are transferable to a security context. For example, Kononenko et al. [19] found a positive correlation between reviewing experience and number of bugs found, unlike the study from Edmunson et al. [13]. With our study we aim to provide deeper insights into code reviewing as a methodology for developer studies within a security context.

3 Methodology

Past work showed that software developers perceive security as more of a secondary task during programming and thus need to be explicitly asked to consider security aspects in the task requirements [22–24, 27]. It is however unclear, whether this holds true for code reviewing. The fact that code reviews are usually written at the end of the SDLC might affect developers' security awareness and thus improve the quality and security of the code.

In this work, we investigated whether software developers think of security when writing a review for security critical code, such as user password storage. For this we set up an online survey, for which we recruited freelance developers on Fiverr.com [1]. The complete survey can be found in the Appendix A. Half the participants were prompted for password-storage security prior to writing the review, and the other half were asked without being prompted to write the review. Hence, we explored the independent variable (IV) *security prompting* with the two values prompting and non-prompting. Additionally, we investigated whether different password storage implementations affect the code reviews. Thus, the participants were shown at random one of three insecure code snippets (plain text, Base64, MD5). Hence, our second IV variable was the *code snippet* each participant received, leaving us with a total of six conditions within our study. We conducted a between-subjects study, where each participant randomly received one of the three snippets.

Apart from prompting the prompted groups of participants to ensure the password is stored securely, we did not give any criteria to complete the review. We wanted to find out which criteria the participants chose and which issues they found without further requests. We recorded all questions received from the participants during the task and recorded our answers to these in a play-book to avoid giving more information to some participants than to others (see Appendix B). In addition to that, we provide further insights on a meta-level for using code reviewing as a new, promising methodology for developer studies within a security context.

We conducted a pilot study with one participant to test the survey and to get a better time estimation for the task. The participant finished within two hours. After correcting minor issues with the survey, we conducted the actual study between May and July 2020. We asked our participants to complete the code reviewing task within one week. Thus, we hoped to increase the number of participants.

3.1 Survey

In the survey, we showed the participants one of three code snippets, each of which contained a password storage implementation. The participants were asked to write code reviews for the snippets. We also asked them to list criteria on which they based their reviews and if they would release the code as it is, with minor adjustments or not at all. After the participants finished their code review, they were requested to explain the concepts of hashing and salting. Further, we asked them about their code reviewing experience and how much of a priority security is to them.

We switched off the back button to prevent the participants from changing their code reviews after being asked for code security. That way we aimed to get an unbiased view on the code reviews and to avoid priming participants for security by the survey.

Since we did not ask participants to write but only review programming code, we included a small programming test to the survey, which was also used by Danilova et al. in [11]. Danilova et al. recruited software developers for an online survey study on security warnings. To assure that their participants really had programming skills, the authors designed a multiple choice question with a code snippet where "hello world" was printed out backwards. About 74% of the participants recruited online on the survey platform Qualtrics failed the test, although all of them indicated to have programming skills. To ensure data quality, our participants were also shown this multiple choice question. Finally, the participants had to answer demographic questions.

3.2 Code Snippets

For the code review task, we chose code snippets submitted by freelancers from the study conducted by Naiakshina et al. [23] for two reasons. Firstly, it was programming code created by freelance developers who believed they were working on the registration functionality for a real company which was to be submitted for release. Secondly, it allowed us to compare our code review results with the findings of the programming code analysis from Naiakshina et al. with regards to both the effectiveness of security prompting and the accuracy of the submissions.

Additionally, we wanted to test whether different programming code snippets have an influence on our participants' submissions. We concentrated on the bad practices used by freelancers in [23]. While we decided upon a plain text code snippet as a baseline, we also added one snippet using MD5 as a hashing function and another using Base64 encoding, as these were prevalent within the freelancers' submissions in [23]. We made sure that the three snippets had an approximately similar length (120-130 lines). To further improve their comparability, we adjusted the selected snippets in the following way. First, to reduce the risk of comments influencing the code review, we deleted all comments from the chosen snippets. To look more realistic, but still stay comparable, we added generic comments which were the same for all snippets. We added comments to the head of the class as well as in the main functions, but not to the trivial setters and getters. Second, we rearranged the order of the functions so that all participants would see the functions in the same order when reviewing the code, as we did not want the function order to influence our results. Third, we added two distraction tasks to all of the snippets:
1) *Exception swallowing*: An empty catch block is considered to be bad practice as possible exceptions would be ignored.
2) *Logical mistake*: Within an if-loop we used only one "="

Table 1: Demographics of participants (n = 44)

Age	min: 19, max: 35	sd: 3.83	median: 24.0	mean: 25.06
General Programming Experience	min: 1, max: 12	sd: 2.43	median: 4.0	mean: 4.46
Java Experience	min: 0.5, max: 10	sd: 2.3	median: 2	mean: 3.19
Gender	Female: 3	Male: 40	Prefer to self-describe: 1	
Occupation	Freelance Developer: 27 Academic Researcher: 1	Industry Developer: 8 Industry Tester: 1	Undergraduate: 2 Graduate: 1	Other: 3 Freelance Tester: 1
Country of Residence	Pakistan: 21, India: 8 Nigeria: 1, Turkey: 1	UK: 3, Portugal: 1 Malaysia: 1, Italy: 1, US: 1	Burkina Faso: 1 Bangladesh: 1	Morocco: 1 Sri Lanka: 2 , NA: 1

in the condition. The condition is therefore always true since an assignment is executed instead. We expected that since the participants are eager to find mistakes in a study on code reviewing, these distraction tasks could divert the attention of the participants from the password storage implementation. The three code snippets can be found in the Appendix C.

3.3 Participants

Like Naiakshina et al. [23], we wanted to recruit freelance developers on Freelancer.com. However, our project was repeatedly denied by the platform with generic explanations such as: "Your project shows behavior that is contrary to our Code of Conduct." We contacted the platform's support service several times to clarify that we wanted to conduct a scientific study with freelancers. However, we were not able to solve our issues at that time and thus decided to use another freelancer platform for the recruitment of participants: Fiverr.com [1].[1]

In the freelancer study by Naiakshina et al., participants received either €120 or €220 for participating in a study of six to eight-hours. No significant difference was found on the security of the submissions between the different payment groups. Since our study was estimated to take one hour and the participant in the pilot study needed two, we decided on a compensation of $50.

We posted our project in four iterations receiving up to 15 applications per posting. With each iteration, the number of repeated offers increased. On Fiverr.com it is required to attach categories and subcategories to the post. Our post was included in the categories "Programming and Tech", "Web-programming" and "Java." In total we received 61 applications to take part in our study. All except four were invited to the study; reasons for not being invited to take part included being under 18 or not having any programming experience. Four freelancers did not respond upon our invitation, six did not want to participate, and two participants canceled after invitation before starting with the study. Finally, 45 freelancers completed our survey. To assure data quality, we excluded one participant from our data set who was not able to answer the "hello world" backward question from Danilova et al. [11].

Table 2: Number of participants per group (n = 44)

	Plain text	MD5	Base64
Prompted	8	7	8
Non-prompted	6	7	8

Table 1 summarizes the demographics of our participants. Out of 44 participants, 40 reported to be male, 3 female, 1 preferred to self-describe. They reported to be between 19 and 35 years old (mean: 25.06 years, median (md): 24 years, sd: 3.83). Further, most of the participants reported to live in Pakistan or India. The general programming experience ranged between one and 12 years with a median of 4 years. All except 2 had at least 2 years of general programming experience and all but 8 reported to have at least 2 years of Java experience (min: 0.5, max: 10, md: 2, mean: 3.19). Most (27) named freelancing as their main profession. All except two participants reported to have reviewed code by others in the past. Table 2 shows the number of valid participants in each group.

3.4 Evaluation

3.4.1 Security

We evaluated the security of participants' code review submissions in the following way. First, we introduced a binary variable *found password storage issue* with two values: 1: participants stated in their reviews, that they found some issues with password storage security; 0: participants did not state in their reviews, that they found some issues with password storage security.

Second, to identify how accurate our participants' code reviews were with regard to the secure password-storage parameters, we used the *security score* of Naiakshina et al. in [23]. Participants received 2 points for hashing and salting the user

[1] Another reason the support service of Freelancer.com offered us was: "Academic cheating is not allowed." As pointed out by Naiakshina et al. [23], it seems that students often use this platform for hiring freelancers to do their university homework. We are still in contact with the enterprise department of Freelancer.com, which reassured us that university studies are welcome to use their platform. It seems that an enterprise self-service would have solved our previous issues.

passwords, another 2 points if the salt was randomly generated and at least 32 bits in length, and another 3 points for iterations, a memory-hard hashing function and if the hash's derived length was at least 160 bits long [24]:

1. The end-user password is salted (+1) and hashed (+1).
2. The derived length of the hash is at least 160 bits long (+1).
3. The iteration count for key stretching is at least 1 000 (+0.5) or 10 000 (+1) for `PBKDF2` and at least $2^{10} = 1\,024$ for `bcrypt` (+1).
4. A memory-hard hashing function is used (+1).
5. The salt value is generated randomly (+1).
6. The salt is at least 32 bits in length (+1).

3.4.2 Qualitative

The code reviews were evaluated qualitatively with inductive content analysis [14]. We decided upon an inductive coding method, as opposed to a deductive coding method, as we did not want to assume what our results would be. The evaluation process included open coding and creating categories. Since we used the "independent parallel coding" approach of David R. Thomas [35], we compared two sets of categories and report the inter-coder agreement for them. The two sets of categories were subsequently merged into a combined set. We calculated the inter-coder agreement Cohen's Kappa (κ) and received an agreement of 0.82. Fleiss et al. considered a value above 0.75 a good level of coding agreement [16].

3.4.3 Quantitative

We additionally conducted an exploratory quantitative analysis. We established the variable from the reviews (found password storage issue) and evaluated the effect of our two IVs (security prompting, code snippet) on this variable using Fisher's exact tests (FET) [15, p. 816]. To test for correlations, we used the Pearson's correlation coefficient. To examine effects in continuous data, we used Wilcoxon Rank sum tests. All tests referring to the same dependent variable (found password storage issue) were corrected using the Bonferroni-Holm correction. The corrected p-values are referred to as $cor-p$.

3.5 Ethics

The institutional review board of our university reviewed and approved our project. We provided the participants of our study with a consent form outlining the scope of the study, the data use and retention policies; we also complied with the General Data Protection Regulation (GDPR). The participants were informed of the practices used to process and store their data and that they could withdraw their data during or after the study without any consequences. Also, the participants were asked to download the consent form for their own use and information.

3.6 Limitations

The participants who took part in this study were recruited on Fiverr.com. They may not be representative for all developers and results may even differ among different freelance platforms. Code reviews are usually performed in companies, so the freelancers might not have had experience with code reviews. However, we aimed to test code reviewing as a study methodology for developer studies as opposed to studying the code review experience.

The majority of participants were non-native English speakers and their responses were not always so clear to understand. Some participants may have had trouble understanding the questions. While this is not desirable for a study, it still represents a realistic scenario, since freelancers with the same issues are hired for real life projects. Moreover, all participants were informed that this project is part of a study. It could be that the participants would have behaved differently had they been writing a code review for a real life project.

We conducted an a priori power analysis with an effect size from Naiakshina et al. from [22] to calculate the necessary sample size to prevent type II errors of falsely rejecting null hypotheses. The required sample sized turned out to be 45 persons per group. However, we were not able to recruit enough freelancers on Fiverr.com, even though we used multiple rounds of recruitment and posted the project repeatedly on the platform. The recruitment of software developers is a challenging task and small sample sizes can limit the method's potential and the generalizability of results. Therefore, our analysis needs to be considered as an explanatory first glance on the problem.

4 Results

The results section is structured as following. First, we present the findings of our qualitative analysis. Second, we report the results of our exploratory quantitative analysis. Third, we compare our results with a similar study containing a programming task on password-storage. We report statements of specific participants by labeling them according to their conditions. The first letter of the label refers to **P**rompting or **N**on-Prompting. The second denotes the code snippet used (**B**ase64, **MD**5, or **P**lain text). While qualitative analysis is more frequently used to explore phenomena, we still provide the numbers of participants to give an indication of the frequency and distribution of themes. An overview of the evaluation of participants' submissions can be found in the Appendix D.

4.1 Qualitative Analysis

The reviews of our participants differed in quality, word count and content. The majority of participants (36 of 44) reported to have reviewed the snippet manually, 4 said that they used an IDE (Eclipse, NetBeans, Visual Studio Code) to check the code, and 4 reported to have used a static analyzer (PMD, sonarlint, findbugs, codacy). With such a small sample size, we could not draw conclusions but it might be worth exploring the use of static analyzers in future studies. On average participants needed a median of 83 minutes to complete the survey. The fastest was submitted after 12 minutes. Some participants took more time since the deadline to complete the project was set to one week.

4.1.1 Participants' Review Criteria

To provide insights into the security awareness and focus of freelancers, we evaluated the criteria which participants mentioned to have looked for in their code reviews. A detailed list of criteria mentioned by our participants is available in the Appendix E. We categorized the answers as follows:

Implementation: A total of six participants said that functionality was one criteria they looked for. Logic was mentioned by eight participants to be an important topic to look for in source code. One participant reported to have looked for maintainability and two mentioned performance and efficiency (PB5, NP4). A large number of participants (14/44) reported to also have checked for error handling.

Testing and bugs: NM3 said that quality assurance was one criteria to check for, while PB7 checked for unit tests and whether all scenarios are considered. NM1, NM5, and NP4 said they looked for bugs or errors in the code. A number of participants mentioned syntax to be a criterion to look for.

Standards and validation: NB1 and PB1 said that they checked whether code standards are met in the code. Three participants mentioned that code format was a criterion they checked for (PB2, NM4, PP5). Nine participants reported that they looked for the correct usage of get and set methods. NM1, NP2, NB8 mentioned the inspection of the model view controller architecture. Further, several participants reported to check imported packages and libraries. Some participants mentioned input validation and null checks (PM6) as criteria they checked the code for. Additionally, several freelancers reported to have checked for code style issues; e.g., camel case or naming conventions in the code. PB1 and PB2 wrote that they looked for duplicated code or unused code. Furthermore, four participants reported to assess the code complexity. Some participants said that they checked the code for readability and comments.

Security: Security in general was mentioned as a criterion by 10 participants. Eleven participants specifically included password storage security in their criteria. Data security was mentioned by 4 participants.

4.1.2 Found Password Storage Issue

Thirteen participants specifically mentioned in their reviews that secure password storage is an issue. Of these, only 2 were non-prompted. To solve the issue, PP8 suggested to use an external authentication service:

> "Depending on the application it may also be better in this case to simply use an external auth service such as that offered by google" (PP8).

Some prompted participants misinterpreted our prompting task description "Please ensure that the user password is stored securely" as password validation (NM3, PP4, PB8). For example, PB8 included secure password policies in the review but failed to detect the insecure password storage method:

> "The password must be at least 8 characters long. The password must have at least one uppercase and one lowercase letter. The password must have at least one digit. This needs to be updated" (PB8).

NM3 mentioned another password validation policy issue:

> "The most important part is the one you are not verifying the password what if it is equal to username. You should know that any person who is trying hit and try on the passwords will definitely first try to enter same username and password and he might be successful in your code and it's the worst part in security risks."

We also found that a number of participants used "password encryption" as a suggestion in their review, which is a concept not recommended for secure user password storage in a database. Furthermore, the prompted participant PB6 wrongly stated that the code snippets contained SQL and JAR injections, but did not mention insecure password storage as an issue. Finally, PM3 perceived the password storage implementation as "too complex" and asked in their review for more comments in the code snippet. A detailed list of code issues reported by our participants is available in the Appendix F.

4.1.3 Security Score

A number of participants commented on the password storage methods of the code snippets. Some participants explicitly said that the password storage functions were sufficient (NM4, NM5, PB8), others saw issues with the functions.

For example, a number of participants recommended using secure hashing functions like bcrypt (PM1, PM5), PBKDF2

(PM6), ARGON2 (PM1) or scrypt (PM5). In contrast, others recommended less secure functions such as MD5 (PP4), SHA-1 (PM4, PP4) or SHA-2 (PM1) (without mentioning iterations). It is noted that SHA-1 and SHA-2 can be secure when used with a key derivation function like PBKDF2, but we cannot assume that the participants mean this if they do not include it in their reviews. The code reviews of the 13 participants who identified user password storage security as an issue often lacked details. Thus, we only were able to grade 11 participants by using the security scale. Table 3 summarizes all the participants who found an issue with user password storage and their score according to the security scale. Only two participants specifically mentioned that a salt should be used.

4.1.4 Distraction Tasks

Some participants found the issues we introduced as distraction tasks. The logical mistake was found by 8 participants and the exception swallowing was mentioned by 12 participants.

NM5 falsely stated that error handling was done correctly. Out of all the 16 participants who found at least one distraction issue, 13 did not find the password storage issue, and 3 did (PM7, PB1, PM4). This might indicate that the distraction issues could have indeed distracted the participants from security. However, from the 13 participants who found the password storage issue, only 3 found at least another distraction issue.

4.1.5 Ready for Release?

We asked our participants to choose whether the code can 1) be released, 2) be released but the issues mentioned in the review need to be fixed for the next update, or 3) whether the code did not pass the review. Out of 44 participants, 2 said that the code can be released. Both found no security issues. Another 17 said that the code can be released but the issues should be fixed for the next update. 12 of these 17 participants did not find the security issue, so they referred to non-security related issues which needed to be fixed. The remaining 25 said the code did not pass the review and should not be released until the issues were fixed. 17 of these 25 participants, however, did not find the security issue with password-storage, so they have based their decision to not release the code on issues other than security. A detailed overview can be found in the Appendix D.

4.1.6 Participants' Definition of Hashing and Salting

After the review completion, we asked our participants to give the definition of hashing and salting passwords. We wanted to find out how many participants were able to correctly define hashing and salting of passwords. On average the participants took 8 minutes to answer this question (md: 5). We evaluated the responses and also tracked whether the participants left the tab inactive while answering the question. Out of 44 participants, 20 participants left the tab inactive for some time while 24 did not leave the tab to answer the question. The participants who left the tab inactive spent a median of 3 minutes outside the tab (mean: 6 minutes). This might indicate, that participants were searching the Web for the answer.

The majority of participants, 63% (28 of 44), gave a correct definition of hashing and salting for password storage. We checked whether the responses matched with definitions from the Internet indicating they were copied and pasted. We found that 8 participants copied the entire definition or parts of their definition from the Internet.

Table 3: Security score of participants who found the password storage issue

	Snippet	P	Score	Salt	Suggestion
NP1	Plaintext	n	0	-	"hashcode generation or convert in Hexa or other formats"
NP5	Plaintext	n	0	-	-
PB1	Base64	p	0	-	"an encoded format"
PB2	Base64	p	2	Y	-
PM1	MD5	p	3	-	"SHA2, Argon2, bcrypt"
PM4	MD5	p	1	-	"SHA-1"
PM5	MD5	p	3	-	"bcrypt, scrypt"
PM6	MD5	p	4	Y	"PBKDF2"
PM7	MD5	p	0	-	-
PP1	Plaintext	p	1	-	-
PP2	Plaintext	p	0	-	-
PP3	Plaintext	p	1	-	-
PP4	Plaintext	p	1	-	"SHA, MD5"

p = Prompted, **n** = Non-prompted

4.1.7 Security Responsibility

We asked our participants whether they felt responsible for end-users' security when writing or reviewing code. Figure 1 visualizes the responses. The "disagree" options 1,2,3 are summarized on the left. By contrast, the "agree" options 5,6,7 are summarized on the right. The percentages next to the bars are the sums of participants' ratings for each side. Neutral (4) is counted as an own point. For both tasks, the participants reported to strongly agree with the statements. However, after completing their reviews, we asked the participants whether they had ensured that the user password was stored securely. Out of 44 participants, 29 answered that they ensured that they had, which did not correspond to the evaluation of their

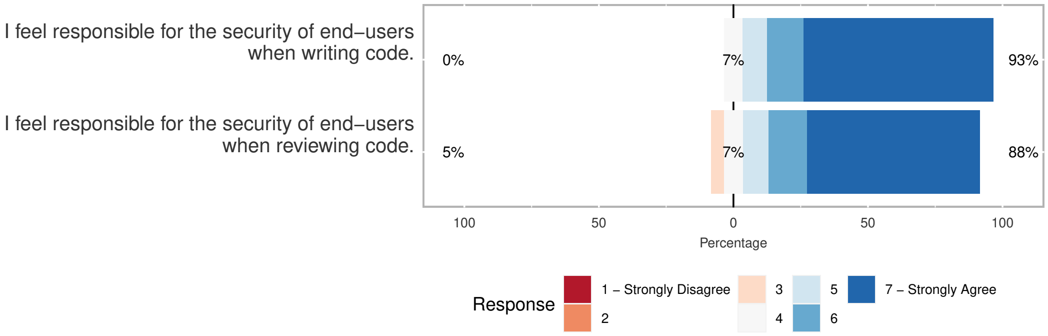

Figure 1: The responses on whether the participants feel responsible for security while code reviewing and writing.

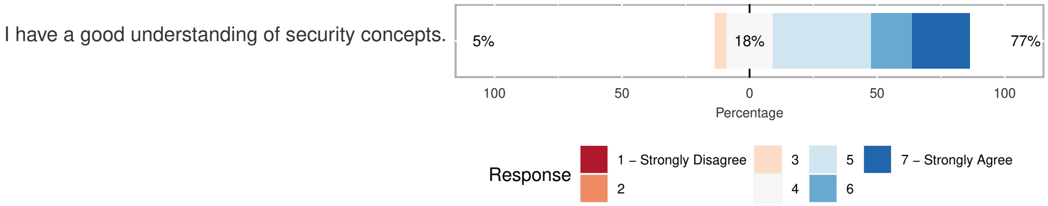

Figure 2: The responses on whether the participants reported to have a good understanding of security concepts.

reviews. In fact, only 13 participants correctly mentioned that insecure password storage was an issue in the snippets. This suggests a social desirability bias while answering survey questions, which was also reported by Naiakshina et al. [23] in their programming study with freelancers.

Additionally, we found an overconfident self-representation of the freelance participants. The degree of agreement to the statement around the freelancers' understanding of security concepts can be found in Figure 2. PP1, NB2, NM2, NP1, and PM6 specifically noted that security is very important in the optional feedback field. For example, NB2 noted:

> "Security is always important, developers put it in the background."

However, 3 participants explicitly noted in the optional feedback field that security is not always important, e.g., "for robotics control" (PP8). Further, PM4 stated:

> "In my experience, sometimes its not all about the security. Some occasions we have to provide hot fixes for urgent customers without thinking about the security. Yes security is an essential factor but it is not something that should be burden to a developer."

4.2 Quantitative Analysis

In this section we report the results of our exploratory quantitative analysis. We tested whether prompting, programming experience or different insecure code snippets had an effect on finding the password-storage security issue. We also tested whether prompting affected the word count of the code reviews and whether the self-reported time participants spend on security differs between programming and code-review tasks.

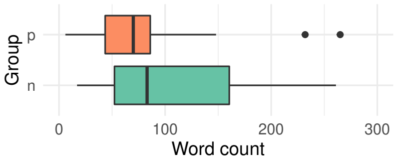

Figure 3: Word count within the reviews for the prompted and non-prompted group.
p: Prompted **n**: Non-prompted

4.2.1 Effect of Prompting on the Word Count

On average the reviews contained 100 words (md: 78 words) with the smallest review containing 6 words and the largest 443 words. We did not find a significant effect of prompting on the word count (Wilcoxon rank-sum, W = 300.5, p = 0.17). Figure 3 visualizes the word count in both groups. Both medians were in a similar range, however, the non-prompted group has a higher variable spread than the prompted group. The number of codes emerging from a code review did not necessarily rely on the word count in the review. Short reviews could cover different issues, while elaborate reviews

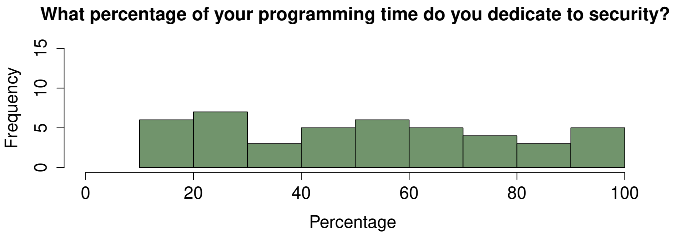

Figure 4: Percentage of *programming time* freelancers dedicate to security (self-reported)

with more details could discuss only one minor issue.

4.2.2 Effect of Prompting on Finding the Password Storage Issue

We evaluated whether participants correctly indicated that there was an issue with password storage security in the snippet (*found password storage issue*). We excluded participants who wrote that the passwords should be "encrypted" if no hashing function was recommended. This might mislead the developer receiving the review to implement encryption instead of hashing the passwords. Naiakshina et al. [23] reported some encryption solutions, which shows that this might be a problem. Eleven prompted and two non-prompted participants correctly stated that password storage was not solved securely in the code snippet. Thus, prompting had a significant effect on finding the issue in the code snippet (FET: p = 0.008*, $cor-p$ = 0.02*, CI = [1.44, 89.85], OR = 8.28).

4.2.3 Effect of Experience on Finding the Password Storage Issue

In [13], Edmundson et al. did not find a significant effect of years of programming experience and whether the reviewers are more accurate or effective. We also investigated whether programming experience had an effect on whether participants found security issues with password storage. We counted how many of our 3 issues the participants were able to find (password storage, 2 distraction tasks). Similar to Edmundson et al., we did not find a significant correlation between the number of issues and the years of general experience (r = 0.05, p = 0.73). Further, we did not find a significant correlation between the number of issues and years of Java experience (r = 0.06, p = 0.72). We also did not find a correlation between Java experience and whether participants found the password storage issue (r = 0.06, p = 0.71).

4.2.4 Effect of Different Insecure Code Snippets on Finding the Password Storage Issue

All the three code snippets (plain text, Base64, MD5) showed an example for insecure user password storage in a database.

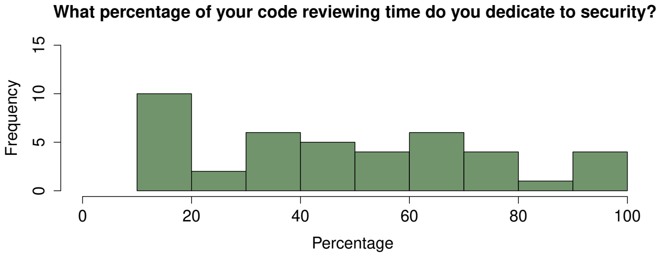

Figure 5: Percentage of *code reviewing time* freelancers dedicate to security (self-reported)

We tested whether the different snippets had an effect on whether participants found an issue with secure password storage. For example, participants presented with an MD5 example might rather report an issue with password-storage security than participants presented with a code snippet where passwords were stored as plain text. We found, however, no significant effect within each subsample, neither the non-prompted (FET: p = 0.07, $cor-p$ = 0.14) nor the prompted group (FET: p = 0.26, $cor-p$ = 0.26).

4.2.5 Time for Security

While Figure 4 visualizes the distribution of percentages of *programming* time dedicated to security, Figure 5 summarizes the distribution of percentages dedicated to security during *code reviews*, according to our participants. The reported median percentage of time that the participants dedicated to security during code reviewing was 50 (mean: 51.64, min: 15, max: 100, sd: 26.54). The reported median percentage of time that the participants dedicated to programming was 55 (mean: 54.89 min: 10, max: 100, sd: 26.31). We did not find a significant difference between both reported time estimations using the sign-rank Wilcoxon test (V = 247, p = 0.12).

5 Discussion

RQ1: Developers' behavior in a security-critical code-reviewing task: Code reviewing is a technique applied at the end of the SDLC, used as one of the final steps by software developers to ensure programming code quality before software release. In comparison to programming code creators, developers take roles as programming code inspectors, which might increase their security awareness. Our study results showed, however, that this is not necessarily the case. It is alarming that almost half the participants wanted to release the insecure code snippets, although security issues with password storage can endanger millions of end-users' data. Even if participants indicated secure password storage as an issue, they often suggested poor techniques or weak hashing algorithms to improve the code. Such poor suggestions, however, might initiate the code creator to revise the code without really improving security. It seems freelance developers need to be reminded of security during code reviewing, otherwise they might be too focused on other issues such as logical mistakes, conventions etc.

RQ2: Factors influencing developers' security awareness: We did not find an effect between which insecure programming code snippet the participant received and whether the participant reported issues with secure password storage. This is especially interesting, considering the fact, that they not only involved plain text password storage, but also Base64 encoding and even MD5 as a hashing function. Furthermore, we did not find an effect of programming experience on finding the insecure password-storage issue, which might indicate that more experience does not necessarily mean that the reviewers are more security aware or effective in finding security issues.

However, similar to the programming studies of Naiakshina et al. [22–24], we found that prompting for security in the reviewing task had an effect on finding the password storage issue in the code snippet. This means, that only if security requirements were mentioned in the task description, do participants consider security issues with password storage in their code reviews. Our results suggested that similar to programming, security needs to be part of the task during code reviewing as well. Therefore, we recommend to prompt for security when a code review is required.

RQ3: Developers' security responsibility in a code review: Our participants reported to spend half their programming and code reviewing time on security. We did not find a significant difference of the reported time spent on security between programming and code reviewing. Additionally, almost all the participants indicated to strongly agree with the statement that they feel responsible for security during programming and code reviewing and that they have a good understanding of security concepts. 66% of the participants also indicated to have ensured that the user passwords were stored securely after their code reviewing task. However, considering that only 13 of 44 (30%) participants reported a security issue with password-storage and the fact that almost all of them were prompted, this might indicate a social desirability bias in surveys. Our qualitative analysis showed that a number of participants had misconceptions and outdated knowledge of secure password storage. This might also suggest that APIs and libraries need to provide safe security defaults instead of requiring software developers to choose security mechanisms.

RQ4: Methodological implications: There is only limited knowledge of using code reviewing as a methodology for security studies with developers. While we provide insights into freelancers' behavior in code-reviewing tasks, we also wanted to explore which advantages, disadvantages and parallel insights a code-reviewing study can have in comparison to a programming study with developers. While we cannot conduct a direct comparison to the study of Naiakshina et al. [23] due to methodological differences, we still discuss the methodology of code reviewing for developer security studies by comparing the advantages, the disadvantages and some parallel insights of both the study types.

One disadvantage was the lack of certain information. We were not able to calculate the security scores of participants in such detail as Naiakshina et al. did. We could not find all information for the security scores in the reviews since participants simply did not mention them. Checking whether participants found the password storage issue was, however, still possible.

Moreover, we found that prompting had an effect on participants' solutions. This indicates that researchers investigating the security awareness of freelance developers might not need to hire them for longer programming tasks. Short and focused code reviews can offer similar results. With a median of 83 minutes to complete the survey, our participants required less time than Naiakshina et al.'s participants, who worked about 6-8 hours on the programming tasks.

Furthermore, code reviewing tasks can give indications to problems with code writing. Similar to Naiakshina et al., we were able to identify different issues developers experienced with password storage. For example, MD5 and encryption were often mentioned as adequate solutions to solve the password-storage issue. However, MD5 is an outdated hash function, which is not recommended any more for secure password storage. With encryption, participants might have referred to symmetric encryption [23], which is, as mentioned before, a discouraged practice for secure password storage. This suggests that code reviewing studies can offer valuable insights into participants' security behavior. We acknowledge though, that code reviewing is a different process to writing code and therefore it is not possible to prove which suggested solutions to the issues developers would really implement.

Similar to Naiakshina et al.'s password-storage study with students, we found that "security knowledge does not guarantee secure software" [24]. Although only 30% of our participants indicated that the user passwords were stored insecurely, 63% were able to provide a correct definition for hashing and salting after their code reviewing task. We have to note, however, that we had only limited possibilities to prove that their definitions were not simply copied and pasted from the Web. Its seemed that 8 participants copied the entire definition or parts of their definition from the Internet, which indicated that knowledge questions should be treated with caution in surveys.

To sum up, we found that security prompting had a significant effect regardless of whether participants completed a programming or a code-reviewing task on password storage. Additionally, we were able to identify participants' misconceptions and outdated knowledge about secure password-storage and which criteria they believe are important in programming code. One disadvantage, however, was that we were limited in the comparison of the participants' security scores, which Naiakshina et al. introduced in their programming study. In our study, the code reviews did not offer enough details to calculate them. Still, code reviewing tasks can help investigate programming knowledge and decrease the time developers need to spend on a task. Participants needed less time to complete the study compared to the studies of Naiakshina et al. while still finding similar results with regard to security awareness and security prompting.

While we do not argue to replace programming tasks with code-reviewing tasks in security developer studies, funding is often limited within academia and smaller tasks yielding similar effects could enable more future research with developers. Therefore, we encourage the community to conduct further research into this line of work.

6 Conclusion

We conducted an online code reviewing study with 44 freelance developers showing each of them an insecure password storage code snippet. We investigated how participants behave in a code-reviewing study by considering which criteria they base their reviews on, whether they would find the security issue and most importantly, whether they would release the insecure code snippets. Additionally, we explored different factors, which might influence their behavior. For example, we explored the effect of prompting for security in the task on whether participants reported password storage security issues within their code reviews. We also explored whether participants feel responsible for and how much time they dedicate to security. Finally, we discussed the methodological implications of a code reviewing study for developer security studies.

Not even one third of our participants reported the security issue with password storage. Almost all the participants who reported an issue were prompted for security. Thus, prompting had a significant effect on participants' behavior. Still, almost half the participants wanted to release the code as it is, which is alarming since insecure password-storage is a major issue endangering millions of users. Finally, our findings suggest that code reviewing studies could be an interesting approach for conducting security developer studies.

For future work we recommend testing a hybrid between a code reviewing and a code writing developer study: a participant could receive functional insecure code and be asked to write a review and if necessary to correct the issues within the code. This could combine the advantages of both the methodologies. However, it might also increase the time of solving the study for the participants again.

Acknowledgments

This work was partially funded by the ERC Grant 678341: Frontiers of Usable Security.

References

[1] Fiverr.com. Accessed: September 2020.

[2] Y. Acar, M. Backes, S. Fahl, S. Garfinkel, D. Kim, M. L. Mazurek, and C. Stransky. Comparing the Usability of Cryptographic APIs. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 154–171, May 2017.

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

[4] Yasemin Acar, Sascha Fahl, and Michelle L Mazurek. You are Not Your Developer, Either: A Research Agenda for Usable Security and Privacy Research Beyond End Users. In *Cybersecurity Development (SecDev), IEEE*, pages 3–8, Piscataway, NJ, USA, 2016. IEEE, IEEE Press.

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

[6] Hala Assal and Sonia Chiasson. Security in the Software Development Lifecycle. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pages 281–296, 2018.

[7] Hala Assal and Sonia Chiasson. 'Think Secure from the Beginning': A Survey with Software Developers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 289:1–289:13, New York, NY, USA, 2019. ACM.

[8] Alberto Bacchelli and Christian Bird. Expectations, Outcomes, and Challenges of Modern Code Review. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, page 712–721. IEEE Press, 2013.

[9] Tobias Baum, Olga Liskin, Kai Niklas, and Kurt Schneider. Factors Influencing Code Review Processes in Industry. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, page 85–96. Association for Computing Machinery, 2016.

[10] Deanna D Caputo, Shari Lawrence Pfleeger, M Angela Sasse, Paul Ammann, Jeff Offutt, and Lin Deng. Barriers to Usable Security? Three Organizational Case Studies. *IEEE Security & Privacy*, 14(5):22–32, 2016.

[11] Anastasia Danilova, Alena Naiakshina, and Matthew Smith. One Size Does Not Fit All: A Grounded Theory and Online Survey Study of Developer Preferences for Security Warning Types. In *Proceedings of the 42nd International Conference on Software Engineering (ICSE'20)*, 2020.

[12] Cleidson R. B. de Souza, David Redmiles, Li-Te Cheng, David Millen, and John Patterson. Sometimes You Need to See Through Walls: A Field Study of Application Programming Interfaces. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, pages 63–71, New York, NY, USA, 2004. ACM.

[13] Anne Edmundson, Brian Holtkamp, Emanuel Rivera, Matthew Finifter, Adrian Mettler, and David Wagner. "An Empirical Study on the Effectiveness of Security Code Review". In Jan Jürjens, Benjamin Livshits, and Riccardo Scandariato, editors, *Engineering Secure Software and Systems*, pages 197–212, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[14] Satu Elo and Helvi Kyngäs. The qualitative content analysis process. *Journal of advanced nursing*, 62(1):107–115, 2008.

[15] Andy Field, Jeremy Miles, and Zoë Field. *Discovering statistics using R*. Sage publications, 2012.

[16] Joseph L Fleiss, Bruce Levin, and Myunghee Cho Paik. *Statistical methods for rates and proportions*. John Wiley & Sons, 2013.

[17] Peter Leo Gorski, Luigi Lo Iacono, Dominik Wermke, Christian Stransky, Sebastian Möller, Yasemin Acar, and Sascha Fahl. Developers Deserve Security Warnings, Too: On the Effect of Integrated Security Advice on Cryptographic API Misuse. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pages 265–281, 2018.

[18] Julie M Haney, Mary Theofanos, Yasemin Acar, and Sandra Spickard Prettyman. "We make it a big deal in the company": Security Mindsets in Organizations that Develop Cryptographic Products. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pages 357–373, 2018.

[19] O. Kononenko, O. Baysal, L. Guerrouj, Y. Cao, and M. W. Godfrey. Investigating code review quality: Do people and participation matter? In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 111–120, 2015.

[20] Katharina Krombholz, Wilfried Mayer, Martin Schmiedecker, and Edgar Weippl. "I Have No Idea What I'm Doing" - On the Usability of Deploying HTTPS. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1339–1356, Vancouver, BC, 2017. USENIX Association.

[21] Thomas D. LaToza, Gina Venolia, and Robert DeLine. Maintaining Mental Models: A Study of Developer Work Habits. In *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, pages 492–501, New York, NY, USA, 2006. ACM.

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

[23] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, Emanuel von Zezschwitz, and Matthew Smith. "If You Want, I Can Store the Encrypted Password": A Password-Storage Field Study with Freelance Developers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 140:1–140:12, New York, NY, USA, 2019. ACM.

[24] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith. Why Do Developers Get Password Storage Wrong?: A Qualitative Usability Study. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 311–328, New York, NY, USA, 2017. ACM.

[25] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, and Matthew Smith. Deception Task Design in Developer Password Studies: Exploring a Student Sample. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pages 297–313, Baltimore, MD, August 2018. USENIX Association.

[26] Duc Cuong Nguyen, Dominik Wermke, Yasemin Acar, Michael Backes, Charles Weir, and Sascha Fahl. A Stitch in Time: Supporting Android Developers in Writing Secure Code. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1065–1077, 2017.

[27] Daniela Oliveira, Marissa Rosenthal, Nicole Morin, Kuo-Chuan Yeh, Justin Cappos, and Yanyan Zhuang. It's the

psychology stupid: how heuristics explain software vulnerabilities and how priming can illuminate developer's blind spots. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 296–305, 2014.

[28] Marten Oltrogge, Yasemin Acar, Sergej Dechand, Matthew Smith, and Sascha Fahl. To Pin or Not to Pin—Helping App Developers Bullet Proof Their TLS Connections. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 239–254, 2015.

[29] Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. Modern Code Review: A Case Study at Google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '18, page 181–190. Association for Computing Machinery, 2018.

[30] Masha Sedova. Comparing Educational Approaches to Secure programming: Tool vs.TA. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, 2017.

[31] Dag IK Sjoberg, Bente Anda, Erik Arisholm, Tore Dyba, Magne Jorgensen, Amela Karahasanovic, Espen Frimann Koren, and Marek Vokác. Conducting realistic experiments in software engineering. In *Proceedings international symposium on empirical software engineering*, pages 17–26, Piscataway, NJ, USA, 2002. IEEE, IEEE Press.

[32] Davide Spadini, Gul Calikli, and Alberto Bacchelli. Primers or reminders?: The effects of existing review comments on code review. In *Proceedings of the 42nd International Conference on Software Engineering (ICSE '20)*, 2020.

[33] Jeffrey Stylos and Brad A Myers. The implications of method placement on api learnability. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 105–112. ACM, 2008.

[34] Mohammad Tahaei and Kami Vaniea. A survey on developer-centred security. *2019 IEEE European Symposium on Security and Privacy Workshops*, pages 129–138, 2019.

[35] David R Thomas. A general inductive approach for analyzing qualitative evaluation data. *American journal of evaluation*, 27(2):237–246, 2006.

[36] Tyler W Thomas, Madiha Tabassum, Bill Chu, and Heather Lipford. Security During Application Development: An Application Security Expert Perspective. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

[37] Sven Türpe, Laura Kocksch, and Andreas Poller. Penetration Tests a Turning Point in Security Practices? Organizational Challenges and Implications in a Software Development Team. In *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*, 2016.

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

[39] Chamila Wijayarathna and Nalin AG Arachchilage. Am I Responsible for End-User's Security? Baltimore, MD. USENIX Association.

[40] Chamila Wijayarathna and Nalin AG Arachchilage. Why Johnny Can't Store Passwords Securely? A Usability Evaluation of Bouncycastle Password Hashing. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, pages 205–210, 2018.

[41] Jing Xie, Heather Lipford, and Bei-Tseng Chu. Evaluating interactive support for secure programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2707–2716, 2012.

[42] Khaled Yakdan, Sergej Dechand, Elmar Gerhards-Padilla, and Matthew Smith. Helping Johnny to Analyze Malware: A Usability-Optimized Decompiler and Malware Analysis User Study. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 158–177, San Jose, CA, USA, 2016. IEEE, IEEE.

A Survey

1. Thank you very much for your interest in our study. We are researchers from the University of Bonn. We are investigating the process of code reviewing for software developers. By taking part in our study you will help us understand how developers explore issues in source code written by other developers.

   Since we will be showing you code snippets please make sure to participate in the survey using your computer instead of a mobile device.

2. You and your colleagues are developing a social networking website to share pictures with family and friends. People need to register to this website in order to share their pictures.

   The front end has already been developed using the Model-view-controller (MVC) architectural pattern. A colleague of yours committed the following class and you have been asked to do a code review for it, before it gets accepted and released.

The class implements a registration functionality in the web application: a user with the following attributes (email, name, gender, birth date, username and password) is stored in the database.

Please review the snippet as you would for a company. You get to decide if this code is deployed or if it goes back to the developer for improvement. In both cases please explain your decision. You can review the code manually or use tools if you prefer. (Prompting: Please ensure that the user password is stored securely.)
[CODE SNIPPET]

3. What did you check for? *[Text box]*

4. Please write your review for your colleague. Please suggest actions for all issues. *[Text box]*

5. Do you approve the release of the code?

    - The code has passed the review. The code can be released
    - The code can be released but the issues mentioned above need to be fixed for the next update.
    - Code did not pass review, please fix issues as mentioned above.

6. Did you check for security? *[Yes; No]*

7. Did you ensure that the user password was stored securely? *[Yes; No]*

8. Can you please explain what hashing and salting for passwords is? *[Text box]*

9. How did you review the code snippet? *[Manually; Using the following tools: [Text box]*

10. In the past I have reviewed code written by others. *[Yes; No]*

11. If Yes: How many times have you reviewed code written by others in the past year? *[Text box]*

12. What percentage of your code reviewing time do you dedicate to security? *[Text box]*

13. What percentage of your programming time do you dedicate to security? *[Text box]*

14. I have a good understanding of security concepts. *1 - Strongly Disagree - 7 Strongly Agree*

15. Please rate the following items: *1- Never - 7 Always*

    - How often do you ask for help when faced with security problems?
    - How often are you asked for help when others are faced with security problems?

16. I feel responsible for the security of end-users when writing code. *1 - Strongly Disagree - 7 Strongly Agree*

17. I feel responsible for the security of end-users when reviewing code. *1 - Strongly Disagree - 7 Strongly Agree*

18. Please enter your age: *[Text box]*

19. Please select your gender. *[Male; Female; Prefer not say; Prefer to self-describe: Text box]*

20. What is your current occupation? *[Freelance developer; Industry developer; Freelance tester; Industry tester; Academic researcher; Undergraduate student; Graduate student; Other:]*

21. What type(s) of software do you develop/test? (Multiple answers possible) *[Web applications; Mobile/App applications; Desktop applications; Embedded Software Engineering; Enterprise applications; Other (please specify):]*

22. In which country do you mainly work / study? *[Text box]*

23. How many years of experience do you have with software development in general?

24. How many years of experience do you have with Java development?

25. How many people work in your team? Please enter 1 if you work on your own.

26. Please select what is more important to you. *[Functionality - Security* (Slider between both, Middle: Equally important)

```
main{
 print(func("hello world"))
}

String func(String in){
   int x = len(in)
   String out = ""
   for(int i = x-1; i >= 0; i--){
       out.append(in[i])
   }
   return out
}
```

Figure 6: Test for software developing skills [11]

27. Please select the returned value of the pseudo code above [see Figure 6]:

- hello world hello world hello world hello world
- world hello
- hello world
- hello world 10
- HELLO WORLD
- dlrow olleh

28. As a non-profit academic institution we are interested in offering fair compensation for your participation in our research. How do you rate the payment of the study? *[Way too little; Too little; Just right; Too much; Way too much;]*
29. How many minutes did you actively work on this survey?
30. Thank you for taking part in our study! We really appreciate your time and effort. We hope our results will help improving security awareness in code reviewing. If you have any comments or suggestions, please leave them here and then please click on "Continue" to complete the survey.

B Play Book

During the study we conducted a play book to ensure all participants received the same information. When a seller contacted us, there were three cases: the offer is the correct amount, the offer is too expensive or the offer is too cheap.

- Hello! Thank you for your interest. We would be delighted to have you participate in our java code reviewing study. Do you have experience programming in Java? If you agree to proceed we would send you a link, from which you can then complete our online survey. We expect the survey to take no more than two hours. To complete the survey you would have a week. Would you like to proceed? Kind regards, XXX
- If the offer was not $50 we added the following question:
- If you would like to participate could you increase your payment requirement and send us a custom offer of $50?
  or
  We do however have a budget of $50 per participant. If you would like to participate could you send us a custom offer of $50?
- Once the participants had sent us a custom offer, they received the answer:
  Thank you! I will confirm your offer and then send you a link to the survey
- When you have completed the survey, we would appreciate it if you do not write any specific comments regarding the survey in your rating of us on Fiverr. As the study is currently ongoing this can lead to inconsistent results. Thank you for your understanding!

Below is a list of questions we were asked and our responses to them (P = Participant):

- P: I have very little programming experience in Java albeit.
  Us: We are looking for people with experience programming in Java. If you feel you fulfill this requirement you are welcome to take part.
- P: Is clicking on that link mean that I must start?
  Us: You should be able to continue where you left off, if you happen to want to continue the survey at a later point.
- P: I hope that the answers are to be in English?
  Us: Yes the survey is in English.
- P: I don’t even know what is the problem and what is it about your research?
  Us: This is a Java code reviewing study. You will be required to complete a code review and then answer some questions.
- P: Why is that obligatory? (to get paid)
  Us: It is important for the study that each participant is treated the same.
- P: How many files / classes are and LOC (Lines of Codes) will be there in the code base? (roughly)
  Us: There are three files, two with roughly 100 lines of code and the third with 15.
- P: Do these two hours have to be without intervals?
  Us: You’re welcome to take breaks as and when you need them.
- P: Just to get to know, do we need to do the survey straightaway for 2 hours or can we save the part that we have done and continue it later?
  Us: You don’t need to complete the survey immediately. You can complete it at any point in the week after your offer is accepted.
- P: No personal information?
  Us: All data will be processed pseudonymously and stored anonymized after the study; there will be no identifying information published in any form.
- P: Seems a little sketchy to be honest, I’d like to make sure this is legit.
  Us: If you would like to participate could you send us a custom offer of $50? We would then accept your offer and send you the link, thus ensuring no risk for you.

- P: I wish to complete your online survey but Unfortunately paying you $50 is stopping me to participate.
  Us: You would be receiving the money.

- P: No I don't have any experience in Java.
  Us: Ok, thank you for your response!

- P: But how you know I take a survey and how you pay me?
  Us: You have sent us an offer of $50. I would confirm this offer and send you a link to our survey. Upon completion you will get paid.

- P: But I don't have any project if I don't deliver how it is possible to send money?
  Us: You have sent us an offer. As mentioned, I would confirm this offer, send you the link to our survey, which you would then complete. You would then confirm that you have delivered the service. We would then check that you have completed the survey. If this is the case, we will confirm completion and you will receive your payment.

- P: And what is the deadline? Is it limited by time?
  Us: You have a week to complete the survey.

- Participant claims to be finished, but the response is not submitted.
  Us: We have not received your response. Can you check that you have completed the survey?

- Participant mentioned word 'security' in review.
  Us: Thank you very much for your kind review. We have however noticed that you mentioned the word "security" in your review. As this study is ongoing, we would rather not have any comments regarding security on our profile. Is it possible you could change your review message? Kind regards, XXX

- P: They are asking for my review will you give me review otherwise I will mention that you haven't given me review after all work.
  Us: I have completed your review already.

- P: Please tell me and type here what review you want from me as seller.
  Us: Telling you what to review us is not in compliance with Fiverr's terms and conditions. We would appreciate it if you do not mention any specifics to the survey, but you are welcome to comment on the experience as a whole working with us.

- P: Do you have something new for me?
  Us: I'm afraid we don't have any more work for you at this time.

- P: What was the survey for? (After completion)
  Us: We are researchers working in the field of software usability. The survey is to be used to better understand how freelancers work and what benefits and disadvantages a code review has.

- By reapplication: Thank you for your interest in our survey, unfortunately we need new participants for the survey.

- Review: Very good communication, delivered on time. It was nice working with *name*!

C Code Snippets

Participants were shown at random one of three insecure code snippets. The code snippets for the study can be found here:

Plaintext

https://gist.github.com/u-cec/54e79635ec44234f8aa8ae4514d3d9e9

MD5

https://gist.github.com/u-cec/d25963ac45569962fca2291661f6e2f8

Base64

https://gist.github.com/u-cec/3fedf84f64918d9cafab61042cee8658

D Evaluation of Participants' Code Reviews

Table 4 shows an overview of the evaluation of participants' submissions.

E Participants' Review Criteria

Criteria mentioned by participants are summarized in Table 5.

F Found Password Storage Issue

Participants who reported issues with the code are summarized in Table 6.

Participant	Code Snippet	Prompted	Survey duration [minutes]*	Time for review [minutes]	Found password storage issue	Ready for release?	Review word count
NB1	Base64	n	31	18	0	✗	127
NB2	Base64	n	86	26	0	✗	56
NB3	Base64	n	24	4	0	✗	17
NB4	Base64	n	316	44	0	✓ (!)	87
NB5	Base64	n	34	14	0	✗	128
NB6	Base64	n	29	12	0	✗	205
NB7	Base64	n	23	11	0	✗	63
NB8	Base64	n	36	27	0	✗	261
NM1	MD5	n	6119	30	0	✗	41
NM2	MD5	n	12	3	0	✓ (!)	33
NM3	MD5	n	326	3	0	✗	443
NM4	MD5	n	62	38	0	✗	177
NM5	MD5	n	83	50	0	✓	117
NM6	MD5	n	254	195	0	✓ (!)	79
NM7	MD5	n	85	49	0	✓ (!)	40
NP1	Plaintext	n	83	37	1	✓ (!)	74
NP2	Plaintext	n	151	118	0	✗	228
NP3	Plaintext	n	37	19	0	✗	73
NP4	Plaintext	n	3511	582	0	✗	171
NP5	Plaintext	n	40	29	1	✗	157
NP6	Plaintext	n	13	3	0	✓ (!)	24
PB1	Base64	p	206	102	1	✗	265
PB2	Base64	p	934	46	1	✗	232
PB3	Base64	p	87	50	0	✓ (!)	28
PB4	Base64	p	12	2	0	✓ (!)	6
PB5	Base64	p	2407	132	0	✗	77
PB6	Base64	p	9045	38	0	✓ (!)	82
PB7	Base64	p	68	41	0	✓ (!)	30
PB8	Base64	p	198	122	0	✓ (!)	70
PM1	MD5	p	25	14	1	✗	82
PM2	MD5	p	5798	1	0	✓	45
PM3	MD5	p	35	14	0	✗	45
PM4	MD5	p	129	105	1	✓ (!)	148
PM5	MD5	p	6153	55	1	✓ (!)	47
PM6	MD5	p	66	15	1	✗	84
PM7	MD5	p	39	15	1	✗	49
PP1	Plaintext	p	23	3	1	✗	25
PP2	Plaintext	p	2490	2	1	✓ (!)	42
PP3	Plaintext	p	4444	67	1	✓ (!)	87
PP4	Plaintext	p	25	12	1	✗	85
PP5	Plaintext	p	76	38	0	✗	107
PP6	Plaintext	p	198	14	0	✓ (!)	89
PP7	Plaintext	p	53	5	0	✗	36
PP8	Plaintext	p	5910	21	0	✓ (!)	68

Table 4: Evaluation of participants' code reviews

* Some participants started the survey and probably left it for some days since the deadline was to complete it within one week. ✗: Code did not pass review, please fix issues as mentioned above. ✓ (!): The code can be released but the issues mentioned above need to be fixed for the next update. ✓: The code has passed the review. The code can be released. **Found password storage issue**: insecure password storage was mentioned as an issue in the review.

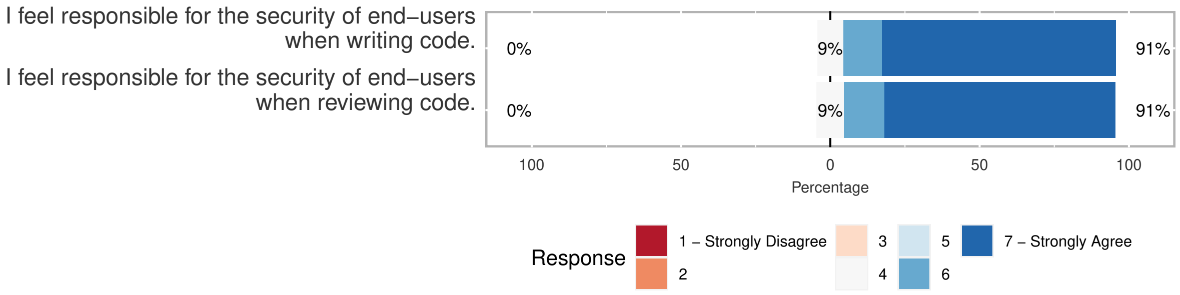

Figure 7: The responses on whether the participants feel responsible for security while code reviewing and writing (group: prompted)

	Criteria	Total Count	Participants	
			Prompted	Not prompted
Implementation	Functionality	6	PM4, PB6 NM5	NB6, NB1, NP1
	Logic	8	PM2, PM3, PP5, PB7	NB4, NP3, NB6, NM6
	Maintainability	1		NB2
	Performance / Efficiency	2	PB5	NP4
	Error Handling	14	PB1, PP3, PB3, PM3, PM5, PP5, PM7	NB1, NM2, NM4, NM5, NB6, NP4, NB8
Testing and Bugs	Quality Assurance	1		NM3
	Unit tests	1	PB7	
	Bugs/ Errors in the code	3		NM1, NM5, NP4
	Syntax	10	PM1, PB3, PM4, PP5 PB7, PP8, PM7	NM6, NP1, NP3
Standards and Validation	Code standards	2	PB1	NB1
	Code format	3	PB2, PP5	NM4
	Correct usage of get and set methods	9	PM1, PB3, PM5, PP5	NM2, NP1, NB3, NB4, NB5
	Model view controller architecture	3		NM1,NP2, NB8
	Imported Packages and Libraries	5	PP3, PB6	NP2, NB1, NB8
	Input validation	6	PM6, PP7, PP5	NB6, NM7, NM2,
	Null checks	1	PM6	
	Style issues	7	PB2, PM4, PP5, PP8	NB1, NP3, NP4, NB7
	Duplicated/ Unused code	2	PB1,PB2	
	Code complexity	4	PM4, PP5, PB5, PB7	
	Readability	5	PM4, PB5	NM1, NM4, NM5
	Comments	5	PB1, PM4, PB7	NB4, NP4
Security	Security	10	PB1, PP2, PP8, PM7, PM4, PP4	NM4, NP3, NP4, NP6
	Password Storage Security	11	PP1, PM1, PP2, PB2, PB6, PM6, PB7, PB8, PM7	NP1, NM4,
	Data Security	4	PB1, PP4, PM7	NM4

Table 5: All criteria mentioned by participants

	Found Issue	Total Count	Participants	
			Prompted	Not prompted
Found Password Storage Issue	Secure password storage	13	PP1, PM1, PB1, PP2, PB2, PP3, PM4, PP4, PM5,, PM6, PM7	NP1, NP5
	Password validation	3	PP4, PB8	NM3
	Password encryption	4	PP1, PP2, PP3	NP5
	SQL and JAR injections	1	PB6	
	Password storage to complex	1	PM3	
Security Score	Storage sufficient	3	PB8	NM4, NM5
	Function issue	5	PM1, PM4, PP4, PM5, PM6	
Distraction tasks	Logical Mistake	8	PM3, PP5, PP7	NM3, NM4, NB5, NP4, NB8
	Exception swallowing	12	PM4, PB1, PB3, PM3, PM7, PP5	NB1, NB2, NB6, NB8, NM4, NP3

Table 6: Issues found by participants

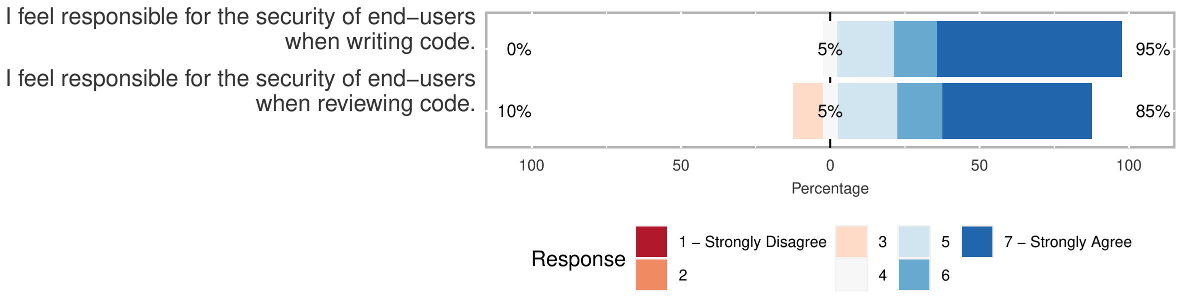

Figure 8: The responses on whether the participants feel responsible for security while code reviewing and writing (group: non-prompted)

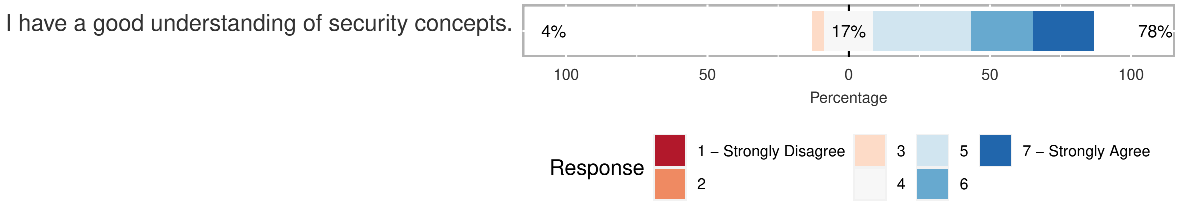

Figure 9: The responses on whether the participants reported to have a good understanding of security concepts (group: prompted)

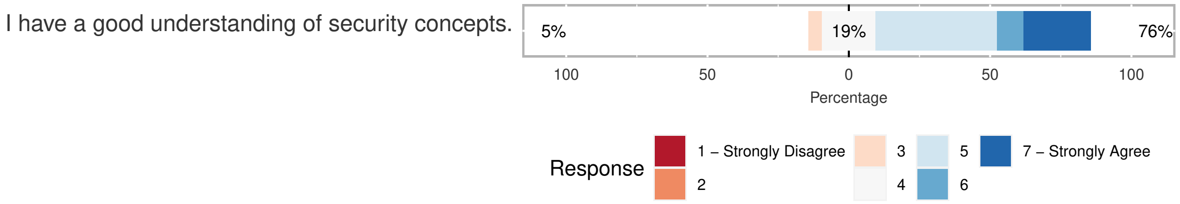

Figure 10: The responses on whether the participants reported to have a good understanding of security concepts (group: non-prompted)