Replication: On the Ecological Validity of Online Security Developer Studies: Exploring Deception in a Password-Storage Study with Freelancers

Anastasia Danilova, Alena Naiakshina, and Johanna Deuter, *University of Bonn;*
Matthew Smith, *University of Bonn, Fraunhofer FKIE*

https://www.usenix.org/conference/soups2020/presentation/danilova

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

August 10–11, 2020

978-1-939133-16-8

Open access to the Proceedings of the Sixteenth Symposium on Usable Privacy and Security is sponsored by USENIX.

Replication: On the Ecological Validity of Online Security Developer Studies: Exploring Deception in a Password-Storage Study with Freelancers

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

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

Johanna Deuter
University of Bonn
johannadeuter@posteo.de

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

Abstract

Recruiting professional developers for studies can be challenging and one major concern for studies examining security development issues is their ecological validity—does the study adequately reflect the real world? Naiakshina et al. [28] examined the ecological validity of a password storage study conducted with students [29, 30] by hiring freelancers from Freelancer.com. In the hope of increasing the ecologically validity, Naiakshina et al. used a deception study design wherein freelance developers were hired for a regular job using a company front created for the study, instead of openly telling the freelancers that they were taking part in a study. Based on their results, Naiakshina et al. propose the use of online freelancers to be examined further, to supplement other recruitment channels such as CS students and GitHub users. The deception in their study was used with the aim that results would reflect the real work of online freelancers. However, deception needs to be used with careful consideration, which can entail additional study design work and negotiations with ethical oversight bodies. In this paper, we take a closer look at the deception used in Naiakshina et al.'s study. Therefore, we replicate Naiakshina et al.'s work but announce and run it as a study on Freelancer.com. Our findings suggest that for this password storage study deception did not have a large effect and the open recruitment without deception was a viable recruitment method.

USENIX Symposium on Usable Privacy and Security (SOUPS) 2020.
August 9–11, 2020, Virtual Conference.

1 Introduction

Security is an issue many software developers struggle [2, 5, 11, 28–30, 47]. User studies with developers are a useful tool to examine misconceptions and issues of developers when faced with security issues. While human computer interaction (HCI) research has made great progress in investigating the security behavior of end users, it still lacks methodological research for the human factor of *software developers* [4]. One major issue with software developers is their limited availability as subjects for research studies. Professional developers often cannot set aside time, may not be locally available or have hourly rates that researchers cannot afford [3–5, 24, 25, 38, 49]. An option to recruit professionals for studies is cooperating with companies as can be seen in [13, 15]. However, it can be difficult to find companies willing to join such studies. Reasons we have encountered are, that companies are hesitant to allocate time of their developers to a study, both for time and cost reasons, as well as worries about disclosure of information about their company as part of the publication process. Therefore, many previous security developer studies were conducted with computer science (CS) students [3, 8, 29–31, 49] or developers recruited on GitHub [5, 20, 35, 47, 48]. To supplement these recruitment options, Naiakshina et al. [28] proposed to use platforms such as *Freelancer.com* for developer recruitment.

In 2017 and 2018, Naiakshina et al. [29, 30] conducted a qualitative and a quantitative password-storage study with CS students to provide more insights into developer's security behavior. The participants were asked to complete the user registration functionally of a university social networking platform. Half the participants were *prompted* for secure password storage in the task, while the other half were merely told the study is about API usability (*non-prompted*). The results showed that not a single participant stored user passwords securely without being prompted. Some students, however, noted that they would have saved the user passwords securely if they had been working on a project for a real company. In order to find out whether the previous results were a study

	CS Students [29, 30]	Freelancers [28]	This Study: Replication of [28] with Freelancers
Independent Variables (IV)	IV1: Security prompting (yes/no) IV2: Framework (JSF/Spring)	IV1: Security prompting (yes/no) IV2: Payment (100/200 euros)	IV1: Security prompting (yes/no) IV2: Payment (100/200 euros) IV3: Framework (JSF/Spring)
Study Context and Task	University researchers University social networking platform	Start-up Sports photo social networking platform	University researchers Sports photo social networking platform
Study Deception	No	Yes	No
Recruitment	University	Individually on Freelancer.com	Project on Freelancer.com
Post Security Request	No	SecRequest-P if plain text submission	SecRequest-P if plain text submission SecRequest-G if not implemented industry security standards

Table 1: Overview of password-storage studies for most relevant aspects.

artifact, Naiakshina et al. conducted a follow-up study with freelancers [28], which they did not announce as an academic study but as a real project. In the corresponding pilot study, the authors found that owing to the university context of task from the previous study, freelancers believed that they were working on university homework and also did not implement any security. To avoid this potential study bias, the authors invested additional study design work to provide a more realistic scenario to make the freelancers believe they were working for a real company and the code would be used in the wild. First, they removed the university social network context and posed as a start-up that had just lost a developer in their team and was searching for a new one. The job freelancers were asked to complete was to create code for a social networking platform for a sports photo sharing website. To make the scenario more believable, the researchers created a fake multi-page online web presence for the start-up. Second, the authors had to manually contact freelancers individually, instead of advertising the job on the service, and letting the freelancers apply for the job. The latter would have made recruitment easier, however, the Freelancer.com platform shows who has been hired for projects advertised in this way and pilot studies showed high dropout rates. This was due to freelancers being confused and suspicious about a single job being given to many freelancers. However, using the deception study design, Naiakshina et al. concluded that the CS students and the freelancers behaved similarly with regard to their password storage practices in this particular study setup and thus, suggested Freelancer.com as a promising platform for developer recruitment which warrants further examination.

In this paper, we specifically want to examine the use of deception in the context of this study. Deception can be a useful tool, if disclosing the study purpose would lead to a significant change in behavior in the participants, e.g., Naiakshina et al. were concerned that freelancers would not implement security in the same way in a study setting as they would for a real customer. However, deception should be used only after careful consideration. Three issues are relevant in our context 1) One of the fundamental principles of human subjects research is *informed consent* and deception can impact this principle. 2) If deception is used, participants who take part in multiple studies or have heard of deceptive studies might second guess what the study is about, potentially affecting their behavior in an unknown manner. 3) Deception studies can require additional study design work, e.g., in the case of Naiakshina et al. the creation of a fake company web presence and more elaborate scenarios.

In order to gain more insights into study methodology and ecological validity of developer studies, we replicated the freelancer study of Naiakshina et al. [28] except for the use of deception. While the study task was kept the same, we created a new profile on Freelancer.com, where we introduced ourselves as a university group conducting scientific research and the job as a study. Due to this we were able to post and manage our study as a single project and manage all freelancers from there, reducing the additional study design work needed to run the study.

In addition to examining the effect of deception, we add further insights into the effect of different application programming interfaces (API) offering cryptographic libraries. For this we replicated the comparison of two frameworks JSF and Spring, as examined in the studies [29, 30],

Finally, we investigated whether providing participants with password storage guidelines has an effect on the security of the submitted solutions. Table 1 provides an overview of the previous lab studies with computer science students [29, 30], the freelancer study [28], and the present study.

2 Related Work

There is a wide variety of studies investigating end-user password creation, password creation rules and their effects and password usage [11, 16, 22, 23, 26, 34, 36, 37, 39, 42–45]. Although developers play an important role in secure password storage too, rather little work has been done with them. To offer more insights into developers' security behavior, Naiakshina et al. conducted password-storage studies with CS students and freelancers [28–30]. A detailed description of the papers is available in the Introduction section. In order to explore the necessity of study deception as done in [28], we replicated the freelancer study of Naiakshina et al. by announcing our study context to participants. Section 2.1 outlines the work conducted on the ecological validity of study

design. Section 2.2 provides an overview of security developer studies in general and in the context of password storage. Finally, Section 2.3 summarizes studies conducted focusing on freelancers.

2.1 Ecological Validity

Acar et al. [4] argued that the examination of ecological validity–whether studies reflect the real world–is of great importance. Most work in this field was primarily conducted with end users. For example, Redmiles et al. [33] compared field data with survey reported data about software updates initiated with end users. They found that self-reported data in some cases varied from field data.

Furthermore, Wash et al. [45] compared 134 self-reports of end users' password behavior with their actual behavior and found only a weak correlation of self-reported intentions with reality. In [46], Wash et al. aimed at finding out which security behaviors were accurately self-reported by the end users. For this, they collected survey responses and behavior data from 122 participants. The authors concluded that security self-reports oftentimes do not reflect users' actual behaviors, e.g., if the behavior involves awareness.

Fahl et al. investigated the ecological validity of a password study [17]. They compared the study-observed behavior of 645 participants with their real-life password choices. They conducted an online and laboratory research under priming and non-priming conditions. The authors found that around 20% of the participants behaved differently in the study compared to their real-life password behavior. They concluded that ecological validity is an important criterion, as it can reveal a high index of the irrelevance of laboratory studies to the real-life behavior.

Simultaneously, Mazurek et al. [27] studied the guessability of passwords used by members of a university in comparison to passwords that were previously collected in experiments or were leaked from low-value accounts. Having the same password policy, the authors found the real university passwords to be more similar to the passwords from research studies than when comparing university passwords to the leak passwords.

In order to increase the ecological validity of developer studies, Stransky et al. [40] designed an online platform which enables developers to participate in a study using their own equipment and allows them to participate from anywhere they would like to. The authors used the platform to conduct two studies and found that participants created code that was equally good as the code created in previous lab studies and noticed that participants were willing to spend more time when working online.

Finally, Naiakshina et al. [28–30] investigated whether deception task design–prompting participants to secure password storage–would change their security behavior in comparison to non-prompting. The study results showed that prompting has an effect on the security of participants' solutions. We replicated the freelancer study of Naiakshina et al. [28] in order to explore what impacts removing the deception emulating ecological validity might have.

2.2 Security Developer and Password Studies

Balebako et al. [7] conducted semi-structured interviews with 13 app developers. They investigated their view on security- and privacy-related topics. Many participants stated that both were part of their development process but that they were not their top priority. The authors found that security and privacy are positively correlated.

Acar et al. [3] invited 54 Android developers to take part in a lab study comparing different kinds of resources (free choice of resources, Stack Overflow only, official Android documentation only, books only). The participants only using Stack Overflow wrote significantly less secure code than the others, while the ones using only books wrote significantly less functional code than the others.

In a further study [2], Acar et al. compared the usability of five different APIs. They asked 256 GitHub users to solve various tasks concerning symmetric and asymmetric encryption. Twenty percent of the users who solved the task functionally believed to have solved it securely while it was not. The researchers recommended better documentation with secure, easy-to-use code examples.

With regard to password policies, Bonneau and Preibusch [12] analyzed 150 websites which offer free user accounts for various purposes. They found a great variety of security implementations. Websites with few security-critical features had the worst security practices. Other websites storing more sensitive data, such as financial data, implemented better security.

In 2007, Prechelt [32] arranged a contest where teams of web developers took part using different web development platforms (Java EE, Perl, PHP). They all had 30 hours to implement the same requirements for a web-based application. Their results were compared along various factors like usability, functionality and security. They found that PHP was in many aspects "at least as safe" as the other platforms and that it tended to have the smallest within-platform variations. Finifter and Wagner [18] conducted further analysis on the code of [32]. The authors investigated the relation between programming language and its number of vulnerabilities and the frameworks' support for security features and number of vulnerabilities. They did not find a relation between choice of programming language and application security, but they noticed that the developers almost never made use of the frameworks' built-in security support, e.g., for password storage. Like Finifter and Wagner, we also investigated whether freelancers would make use of the Spring frameworks' built-in libraries for secure password storage.

In a further study, Acar et al. [5] invited 307 GitHub users to work on security related tasks (e.g., password storage) in

Python and to take part in a survey afterwards. The authors found a positive correlation between performance regarding security and functionality and years of programming experience.

Another password-storage study was conducted by Wijayarathna and Arachchilage [47] with 10 developers. The authors explored usability issues of the Bouncycastle API to provide insights on how to develop, design and improve security/cryptographic APIs. They identified 63 issues in the SCrypt implementation of Bouncycastle.

2.3 Developer Studies with Freelancers

Ahmed and van den Hoven [6] discussed the freelancers' responsibility and ability to cause harm. They pointed out that most freelancers only do exactly what they are asked to do, which can cause security issues with employers who do not have a computer science background–an observation which was also found in Naiakshina et al.'s studies with freelancers [28] and students [29, 30] indicating that this issue cannot be reduced to freelancers only. Another problem Ahmed and van den Hoven found was that freelancers use malicious code from the Internet without testing it to ensure functionality and security. The researchers wanted to encourage freelancers to accept their responsibility. In our study, we acknowledge these aspects by providing participants password storage standard sources, if they submitted non-secure solutions.

In a further study, Bau et al. [9] compared the websites developed by start-up developers with websites they asked freelancers from Elance.com and Freelancer.com to develop. They wanted to investigate how the employment status, the developer's security knowledge and the programming language influence web application security. Nineteen start-ups and 8 freelance developers were invited and all of them were interviewed and took part in a security quiz. Their analysis showed that the code written by freelancers had more weaknesses than the code written by the start-ups. There was a huge discrepancy between the freelancers' security knowledge and their implementations. Furthermore, they found that code written in PHP had more injection vulnerabilities than code written in other programming languages.

While Bau et al. referred to freelancers as being rather unreliable, in another freelancer study, Yamashita and Moonen [50, 51] conducted a study with 85 freelancers on code smells and acknowledged the flexibility, the access to a wide population, and the low costs of Freelancer.com. Furthermore, in contrast to Bau et al., Naiakshina et al. [28] reported freelancers to be very dependable in their study. Most of the subjects delivered their solutions within the time frame they promised; they were also reliable in their communication and showed a high interest in the study results.

3 Methodology

With the aim of improving the ecological validity of their study, Naiakshina et al. [28] concealed the context of their scientific research and hired freelancers for a "real project." The authors invested additional study design work into the deception by creating a fake start-up with a web presence, creating a fake profile on Freelancer.com, designing a task description as authentic as possible and contacting and hiring individual developers based on their skills.

To test whether similar results are achieved without the use of deception, we conducted a replication-extension [10] of Naiakshina et al.'s [28] freelancer study, which adopted the study design from the original CS student studies [29, 30]. While we replicated the study of Naiakshina et al. with freelancers [28], we also extended it by the methodology (study announcement) and the framework variable, which we adopted from the previous student studies [29, 30] as well as an additional security request (see Table 1). The task was to complete a Java registration functionality that facilitated the storage of user properties (including user passwords) from a web form in a database. We used the task description given in [28] and hired participants from Freelancer.com. Similar to the previous studies, *prompting* was one variable in our study: half of the participants were tasked to securely store the passwords while the other half was not explicitly asked to do so.

When the project was published on Freelancer.com, the participants bid on it, and we contacted them via private messages. To communicate with the participants, we used the playbook from [28] and extended it when new cases appeared. The changes we made are given in the Appendix E. After completing the programming task, participants were asked to fill out an online survey to obtain their opinions on the used frameworks, their demographics, and their feedback about the task. In the following section, we provide a detailed description of all the design changes we made in contrast to the previous study [28].

3.1 Study Design Changes

Recruitment. One major change from the previous work was the public posting of the project on the freelancer platform as part of a scientific study. In the freelancer study of Naiakshina et al. [28], the researchers sent private messages with the project offer to freelancers who had mentioned Java knowledge in their profiles. Due to the limited filter features, the researchers needed to manually inspect freelancers' profiles to verify whether they actually had the required knowledge. Eighty of the 340 selected subjects did not have the required knowledge. Additionally, the remaining 260 freelancers were contacted by the researchers, but a total of 211 did not accept the offer for different reasons, such as their lack of experience or time. In our study, developers had to submit an application

for the project and thus, it was ensured that all the applicants were available for the study. Our public announcement for the study is available in the Appendix E.1.

Study Announcement. In the previous study, the researchers presented themselves as a start-up company and revealed their academic aims only at the end of the programming task. To make the project as realistic as possible, the original task presented in [29, 30], needed to be changed from a university setting to a company setting. The authors created a fake company profile on Freelancer.com and a web presence for that company and shared that with the subjects. While we used the same task description as used in the previous freelancer study, we omitted all the other steps and introduced ourselves as academic researchers conducting a scientific study.

Framework. While in the original freelancer study only JSF was used, we randomly assigned the participants to use either JSF or Spring as introduced in the previous student studies [29, 30]. The JSF participants had to implement secure password storage on their own. In contrast, Spring offers supporting libraries.

Security Requests. In their CS student studies, Naiakshina et al. [29, 30] accepted the participants' initial solutions and evaluated them based on a security score. To sum up, participants received a score of 0-7 points for security, based on their implementation choices for the password storage security, such as the hashing algorithm, salt generation, iterations etc. (see Section 3.4). In the freelancer study, the authors extended the security score and included a security request:

- *SecRequest-P(laintext)*: If the submissions included plain text password storage, subjects were asked to revise their submissions and to securely store user passwords.

We adopted this procedure and extended it by a further security request:

- *SecRequest-G(uideline)*: If the security score of participants' solutions was less than 6 points, we asked them to store the passwords by following industry's best practices.

As in the previous studies by Naiakshina et al., we accepted 6 points, instead of the full 7, for security, because Springs' default implementation of the bcrypt scores 6 points; thus, no one received the full 7 points by using memory-hard hashing functions. To investigate whether developers could obtain the full 7 points when given the appropriate source, we provided our participants with website links to the password security guidelines of the Open Web Application Security Project (OWASP) [1] and National Institute of Standards and Technology (NIST) [21] for the SecRequest-G. The exact wording and an illustrated procedure of the security requests can be found in the Appendix A.

Compensation. In the freelancer study, Naiakshina et al. [28] tested the impact of a payment variable on security by recruiting subjects either with a payment of €100 or €200. After revealing the study context, freelancers were invited to fill out a survey for additional €20. To allow researchers to conduct scientific studies in an economical manner, we began recruiting participants with a lower compensation amount of €120 for the project, including the programming task and the survey. Naiakshina et al. did not find a significant effect between the two payment levels on security. Therefore, after facing difficulties to recruit over 18 participants with €120, we started offering €220.[1]

Performance Based Payment. Based on the insights obtained from our pilot study (see Section 3.2), our payment method resembled the prior freelancer password storage study [28] but was split into compensation milestones. Since we included up to three possible iterations of the code review and possible security requests, we divided the payment process based on three milestones: €50 for the first code submission, €50 for the final code release after our review including possible security requests, and €20 the survey completion. Accordingly, participants received €100, €100, and €20 respectively for each milestone for a payment of €220.

3.2 Pilot Study

We recruited two freelancers via private messages to participate in our pilot study. We offered €220 to each. One participant reported that he wanted further instructions instead of just the mention of OWASP and NIST. Therefore, we included the links to the security guidelines in the final study. Since one participant appeared to be bothered by the requests, we divided the payment process based on milestones to clearly indicate that there were additional steps in our study. One participant stated that it took him six hours to finish the task, but he submitted the solution after six days. The other participant worked for three and a half hours on the task. Since both the participants reported to have finished the task in a relatively short period of time we started the recruitment process with €120.

3.3 Participants

Recruitment. We posted our project in 5 iterations over a time-span of almost two months. On each public project, freelancers could place bids with their payment offer, which ranged from €120/€220 to €250. In total we received 101 applications, of whom we invited 73 to the study. Since we limited our payment to €220, participants with higher bids were not invited to the study. In total, we excluded 28 applicants for requirement reasons, such as participants with a bid higher than we offered (12), already participated in our

[1] As in the previous freelancer study [28], payment did not show an effect on the decision to include security in the initial solution (FET: $p = 0.55$, OR = 1.54, CI = [0.39, 6.19]). We also regarded the prompting vs. the non-prompting group and did not find any significant effect in both cases.

General Information:			
Gender	Male: 32	Female: 11	NA: 0
Age	min: 18, max: 46	mean: 28.95, median: 29	SD: 6.21
Country of Residence	China: 11, Hong Kong: 1, The Netherlands: 1, Palestine: 1,	India: 10, Gaza: 1, New Zealand: 1, Romania: 1,	Pakistan: 6, Russia: 3, Lithuania: 1, Vietnam: 1, Germany: 1, Malaysia: 1
Profession and Programming Experience:			
Profession	Freelance Developer: 28 Employed full time: 1	Industrial Developer:12 Undergraduate full-time Student: 1	Finance Professional:1
General Development Experience [years]*	min: 1, max: 20	mean: 7.42, median: 7	SD: 4.47
Java Experience [years]*	min: 1, max: 16	mean: 6.19, median: 5	SD: 3.69

* = There were no significant demographic differences between the groups.

Table 2: Demographics of participants (n = 43)

study (9), and other reasons (7), such as missing Java skills in their profile or large time frames. Forty-three accepted our study invitation and participated in the study.

The first 3 iterations were posted with a payment of €120 and received a total of 60 bids from which we invited 46 freelancers to the study. Fourteen were removed for requirement reasons. Eight participants did not answer to the study invitation and another 8 were not interested anymore after reviewing the study material. Two wanted a higher payment for the project, one had no Java skills, two declined without seeing the study materials and one did not believe that we were conducting a study. Finally, 24 agreed to participate in our study. In the last 2 iterations, we offered a payment of €220. We received 41 bids and contacted 27 potential participants, of which 19 agreed to participate in the study. Fourteen freelancers were removed for requirement reasons. Six participants did not respond and one told us he is not interested anymore. Another participant wanted a higher payment for the project. In total 43 freelancers participated in our study and were randomly assigned to one of the 4 conditions: Spring-Prompting (FSP), Spring-Non-Prompting (FSN), JSF-Prompting (FJP), and JSF-Non-Prompting (FJN).

Demographics. Table 2 summarizes the demographics of our participants. While the demographics were comparable to Naiakshina et al.'s [28] freelancers in general, more female participants were involved in this study. 74% (32 of 43) were male, and 26% (11 of 43) were female. Most of the participants claimed to be from China (11), India (10), and Pakistan (6). Their ages ranged between 18 and 46 years (mean: 28.95, median: 29, SD: 6.21). The general programming experience of the participants ranged from 1 to 20 years (mean: 7.42, median: 7, SD: 4.47). For Java, the programming experience was reported to be between 1 and 16 years (mean: 6.19, median: 5, SD: 3.69). Almost all the participants reported to be experienced in developing web applications (41 of 43) and desktop applications (27 of 43). Thirty-eight participants reported to have a university degree. The freelancers had the option of indicating a minimum hourly wage in their Freelancer.com profile. The lowest rate among our participants was €5/hour, while the highest was €46/hour (mean: 21.71, median: 19, SD: 11.3, NA: 2).

3.4 Evaluation

Code Analysis. When releasing the milestones, we accepted only functional solutions. We adopted the extended version of the security scale [29] for the password storage security used by Naiakshina et al. [28–30] to score the code submissions. To sum up, we used a binary variable *secure* that indicated whether a participant included at least some kind of security. An ordinal variable *security score* was used to assess the security of the solution according to the password security scale [29] from 0-7; the *security score* considered the hash algorithm, iteration count for key stretching and salt generation. Submissions in which the user passwords were stored as plain text in the database received 0 points. Base64 and symmetric encryption are not suitable methods for secure password storage, and thus, any submissions that included these methods were rated being insecure (0 points). The security scale can be found in the Appendix B.

Two coders independently reviewed all the programming code submissions and evaluated them for security. Disagreements were resolved by consulting a security expert and discussing the algorithm specifications. We had 7 cases of disagreement, e.g., if one coder assessed the iterations incorrectly or misread the hash length. However, after a discussion all cases were resolved. With the rigid scoring system and the strict algorithm specifications, full agreement was achieved among the researchers.

Statistical Testing. We evaluated the same hypotheses as Naiakshina et al. [28, 30]. We were able to examine the effect of prompting vs. non-prompting (H-P1), framework (H-F1), years of Java experience (H-G1), and password storage experience (H-G2) on security. We also used the same statistical tests as in [28, 30]. All the tests on the same dependent variable were corrected using the Bonferroni-Holm correction. We denoted all corrected tests with "family = N," where N is the family size, and reported both the initial and corrected p-values (cor-p). An additional description of the hypotheses and a summary of our statistical analysis can be found in the Appendix C and D.

Qualitative Analysis. The open-ended questions from the follow-up survey were analyzed by two researchers using inductive coding [41]. The two researchers independently searched for codes categories and themes emerging in the raw

data. The codes were compared and the inter-coder agreement was calculated by using the Cohen's kappa coefficient (κ) [14]. The agreement was 0.83. A value above 0.75 is considered a good level of coding agreement [19]. We found a large number of similar codes as in the previous freelancer study [28]. In order to provide novel insights, we only report new codes and findings in this work.

4 Limitations

Sample. Similar to Naiakshina et al. [28], our sample consisted of developers from Freelancer.com. This sample is not be representative for all developers. Other freelancer hiring services could be used by other developers, and the results may vary. Further, since we publicly announced the project, participants needed to actively contact us, leading to a possible self-selection bias.

Recruitment Payment. We used two payment levels to recruit participants. This might have led to a self-selection bias. However, we tested the effect of payment on security in the initial solution and did not find any significant effect between both payment levels.

Deception. Similar to Naiakshina et al.'s previous password storage studies, this work examined the prompting vs. non-prompting effect in the task description. This resulted in concealing the security-focused research from half of our participants. However, due to our security requests, participants were able to improve their submissions. We received only positive feedback from our participants.

Generalizability. Finally, our findings are based on a single example study and thus further studies are needed to see if our results replicate in other types of studies.

5 Ethics

The institutional review board of our university approved our project. The participants were sent a link to a consent form in the first message of the conversation. We informed them of our data-storage policies and that they could withdraw their data at any time. To treat all the participants equally, we compensated the participants who had initially been offered a lower pay amount (€120) with additional €100 at the end of our study. Thus, all our participants received €220 for their efforts.

6 Results

In this section, we present an analysis of the present study. Additionally, we compare the results of the present study with results from the previous studies [28, 30]. To enable this comparison, we investigated the effect of the same factors on whether participants decided to store user passwords securely in the database considering the initial submissions. Further, we compared submissions from the previous freelancer sample [28] with our sample.

6.1 Security

As in the previous freelancer study [28], our participants used three techniques to store user passwords in the database: (1) hashing (+ salting); (2) symmetric encryption; and (3) Base64 encoding. We rated the solutions according to the security scale introduced in Section 3.4.

Table 3 shows the summary of the initial password storage solutions and the solutions handed in after SecRequest-P (in bold). To submit their initial solutions, participants took on average 4 days (min: 2h 20 min, max: 29 days, median: 2 days, SD: 5.4 days). In total we received 23 non-secure and 20 secure initial solutions from our 43 participants. Seventeen of the 23 participants with non-secure submissions received SecRequest-P as they stored the user passwords in plain text. Most of these requests were sent to non-prompted participants (15/17). Including security prompting in the initial task description meant that there was (almost) no need to remind participants not to save passwords in plain text. For SecRequest-P, they needed on average 1.8 days (min: 15 min, max: 19 days, median: 2h 30 min, SD: 4.4 days). After SecRequest-P all participants except one at least hashed the user passwords.

Overall, 25 participants received SecRequest-G because their first or second submission achieved less than 6 points on the security scale. For SecRequest-G, participants needed on average 2 days (min: 15 min, max: 19 days, median: 1 day, SD: 3.7 days). We provide a deeper analysis of submissions after SecRequest-G in Section 6.5.

Participants needed on average 5.8 days for their final submissions (min: 2h 20 min, max: 32 days, median: 3 days, SD: 7.9 days). In contrast to the previous work [28], we wanted to have a more accurate time specification. Thus, we asked participants in the survey how much time they actually needed to finish the task. On average, they stated that it took them 14 hours and 30 minutes to complete the task (min: 1 h, max: 72 h, median: 10 h, SD: 14 h 50 min).

6.2 Prompting effect (H-P1)

As done with all the previous studies on password storage of Naiakshina et al. [28–30], we examined the effect of prompting for security in the task. We also found a significant effect of prompting on the security of participants' submissions (FET: $p = 0.006$*, $cor - p = 0.01$*, OR = 6.51, CI = [1.51, 33.18], family = 2). The majority of non-prompted participants submitted a non-secure solution (16 of 21); only 5 participants considered security. Of the 22 prompted participants, 15 at least hashed the passwords.

Participant	Prompting	Framework	Payment	Working Time	Include SecRequest-P	Active Working	Security Requests	Function	Length in bits	Iteration	Salt	Secure	Score	Copied
FJN1	**0**	**JSF**	**120**	**3 Days**	**3 Days**	**7h**	**P + G**	**SHA-1**	**160**	**1**		**0/1**	**0/2**	
FJN2	**0**	**JSF**	**120**	**3 Days**	**2 Days**	**4h**	**P**	**bcrypt**	**184**	2^{10}	**SR**	**0/1**	**0/6**	
FJN3	**0**	**JSF**	**120**	**18 Days**	**2h 30min**	**11h**	**P + G**	**MD5**	**128**	**1**		**0/1**	**0/1**	✓
FJN4	0	JSF	120	3 Days		8h		PBKDF2 (SHA-512)	512	20 000	SR	1	6	
FJN5	**0**	**JSF**	**220**	**2 Days**	**30min**	**12h**	**P + G**	**MD5**	**128**	**1**		**0/1**	**0/1**	
FJN7	0	JSF	120	3 Days		25h	G	MD5	128	1		1	1	
FJN8	0	JSF	120	7 Days		4h	G	PBKDF2 (SHA-512)	512	65 536	St	1	5	
FJN9	**0**	**JSF**	**220**	**12 Days**	**1 Day**	**30h**	**P + G**	**SHA-1**	**160**	**1**		**0/1**	**0/2**	
FJN10	0	JSF	220	3 Days		8h	G	pgCrypto(xdes)	64	725	pgC	1	3	
FJN11	0	JSF	220	2h 20min		4h		bcrypt	184	2^{10}	SR	1	6	
FJN12	**0**	**JSF**	**220**	**2 Days**	**35min**	**8h**	**P + G**	**MD5**	**128**	**1**		**0/1**	**0/1**	
FJP1	1	JSF	220	1 Day		5h		bcrypt	184	2^{10}	SR	1	6	
FJP2	1	JSF	120	3 Days		48h	G	sym. Encryption				0	0	
FJP3	1	JSF	120	1 Day		10h	G	PBKDF2 (SHA-1)	128	65 536	SR	1	5	
FJP4	1	JSF	220	1 Day		18h		bcrypt	184	2^{14}	MR	1	6	✓
FJP5	1	JSF	120	6 Days		16h		PBKDF2 (SHA-1)	160	20 000	SR	1	6	✓
FJP6	1	JSF	120	2 Days		3h	G	MD5	128	1		1	1	✓
FJP7	**1**	**JSF**	**120**	**3 Days**	**15min**	**10h**	**P + G**	**MD5**	**128**	**1**		**0/1**	**0/1**	✓
FJP8	1	JSF	120	4h		1h	G	MD5	128	1		1	1	✓
FJP9	1	JSF	120	12 Days		60h	G	Base64				0	0	
FJP10	1	JSF	120	1 Day		12h	G	sym. Encryption				0	0	
FJP11	1	JSF	220	2 Days		10h	G	sym. Encryption				0	0	
FSN1	**0**	**Spring**	**120**	**3 Days**	**5h**	**15h**	**P + G**	**SHA-512**	**512**	**1**	**SR**	**0/1**	**0/5**	
FSN2	**0**	**Spring**	**220**	**8h 30min**	**20min**	**4h**	**P**	**bcrypt**	**184**	2^{10}	**SR**	**0/1**	**0/6**	
FSN3	**0**	**Spring**	**120**	**7 Days**	**25min**	**10h**	**P**	**MD5**	**128**	**1**		**0/1**	**0/1**	
FSN5	0	Spring	120	1 Day		6h	G	sym. Encryption				0	0	
FSN6	**0**	**Spring**	**120**	**13 Days**	**19 Days**	**30h**	**P**	**bcrypt**	**184**	2^{12}	**SR**	**0/1**	**0/6**	
FSN7	**0**	**Spring**	**120**	**1 Day**	**1 Day**	**10h**	**P + G**	**SHA-1**	**160**	**1**		**0/1**	**0/2**	
FSN9	**0**	**Spring**	**220**	**1 Day**	**2h**	**6h**	**P + G**	**SHA-256**	**256**	**1**		**0/1**	**0/2**	
FSN10	**0**	**Spring**	**120**	**1 Day**	**1h 10min**	**6h**	**P + G**	**sym. Encryption**				**0/0**	**0/0**	
FSN11	**0**	**Spring**	**220**	**9h**	**20min**	**2h**	**P**	**bcrypt**	**184**	2^{10}	**SR**	**0/1**	**0/6**	
FSN12	**0**	**Spring**	**220**	**1 Day**	**3 Days**	**9h**	**P**	**bcrypt**	**184**	2^{10}	**SR**	**0/1**	**0/6**	
FSP1	1	Spring	220	29 Days		72h		bcrypt	184	2^{10}	SR	1	6	
FSP2	1	Spring	120	4 Days		20h	G	SHA-256	256	1		1	2	
FSP3	1	Spring	120	4h 30min		2h		bcrypt	184	2^{10}	SR	1	6	
FSP4	1	Spring	120	5 Days		10h	G	MD5	128	1 000	SR	1	4	
FSP5	1	Spring	220	2 Days		16h		bcrypt	184	2^{10}	SR	1	6	
FSP6	1	Spring	120	1 Day		12h	G	Base64				0	0	
FSP7	1	Spring	220	7 Days		20h		bcrypt	184	2^{10}	SR	1	6	
FSP9	1	Spring	220	1 Day		11h		bcrypt	184	2^{10}	SR	1	6	
FSP10	**1**	**Spring**	**220**	**2 Days**	**1 Day**	**35h**	**P**	**bcrypt**	**184**	2^{10}	**SR**	**0/1**	**0/6**	
FSP12	1	Spring	220	2 Days		7h		bcrypt	184	2^{10}	SR	1	6	
FSP13	1	Spring	220	1 Day		8h		bcrypt	184	2^{10}	SR	1	6	

Table 3: Evaluation of participants' initial and SecRequest-P submissions
Bold: Participants who at first delivered a plain text solution and thus, received the first security request (SecRequest-P). **Working Time** participants took to submit their initial solution. **Include SecRequest-P:** Time participants needed to add security after SecRequest-P (1 Day = 24 hours). **Active Working:** Self-reported active working time reported by participants for their final submissions. **Salt:** SR = SecureRandom, St = Static, pgC = pgCrypto, MR = Math.Random. **Copied:** Security code was most likely copied and pasted from the Internet.

6.3 Java and Password Storage Experience (H-G1, H-G2)

Similar to the previous freelancer study, we did not find any effect of Java experience on the security score of the submissions (Kruskal-Wallis: $\chi^2(12) = 8.46$, $p = 0.75$, $cor - p = 0.75$). Further, we only examined submissions which did include some security, but did not find any effect of Java experience on the security score either (group: secure = 1; Kruskal-Wallis: $\chi^2(7) = 3.83$ $p = 0.80$).

In addition to that, we asked our participants whether they had stored user passwords in a database before. Only 3 of 43 participants said that they had never stored passwords before. As in the previous studies [28, 30], we did not find any significant effect on their decision to store the passwords securely in our study (FET: $p = 0.59$, $cor - p = 0.59$, OR = 0.42, CI = [0.01, 8.63]).

6.4 Framework (H-F1)

Similar to the previous student study [30], we did not find that the framework had a significant effect on the security score when participants stored the passwords securely (group: secure = 1; Wilcoxon Rank sum: W = 32.5, $p = 0.16$, family = 2, $cor - p = 0.32$). The mean score for the JSF group was 4.18 (group: secure = 1; min: 1, max: 6 , median: 5, SD: 2.23). The Spring group received a higher mean of 5.33 (group: secure = 1, min: 2, max: 6, median: 6, SD: 1.41). In both studies the spring group achieved slightly higher scores, but since neither effects were statistically significant, future studies will have to decide whether the small improvement the spring framework might bring is worth conducting a study with more power.

Further, we investigated the reported usability of both APIs. We calculated the API usability scores suggested by Acar et al. [2] for the Spring group (min: 62.5, max: 92.5, mean: 74.17, median: 72.5, SD: 9.36) and the JSF group (min: 50, max: 100, mean: 68.75, median: 70, SD: 13.11). The

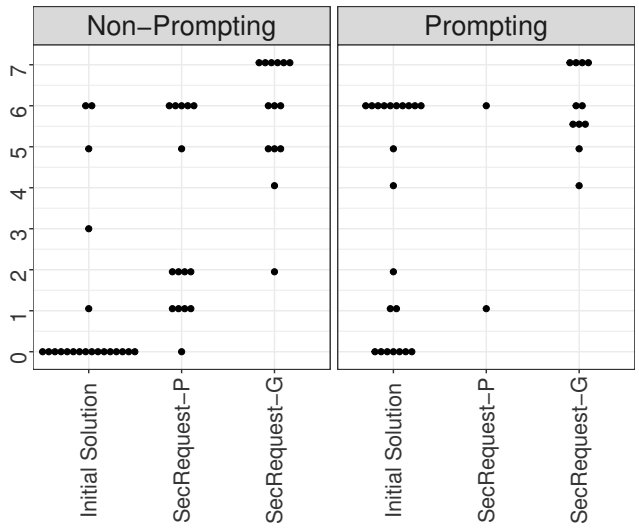

Figure 1: Distribution of scores before and after the security requests for the prompted and non-prompted group.
Each point stands for one participant.

score could range from 0-100 with 100 being the highest usability score that can be achieved. Thus, the Spring group achieved higher usability scores, however the difference was not statistically significant. (Wilcoxon Rank Sum: W = 160, $p = 0.08$).

Moreover, we tested for a correlation between API usability and the achieved security scores but did not find any significant correlation (Pearson: r = 0.08, $p = 0.57$).

In this study, we asked participants to evaluate the API usability after possible security requests, so we can be certain that all the participants used security mechanisms within the frameworks before giving us their assessment. In the student study [30] however, there were no follow up security requests and thus thus, there were participants who did not use any security mechanisms. Since their experience with a framework was reported based on functionality aspects only, we do not draw a direct comparison to our study.

6.5 Security Guidelines (NIST and OWASP)

Figure 1 shows the distribution of scores for the prompted and non-prompted groups for initial submissions and after participants received SecRequest-P and/or SecRequest-G. The figure visualizes how the distribution evolved after each request. The achieved scores rose after each request. Out of all 43 participants, 25 received SecRequest-G with web links to NIST and OWASP. The evaluation of these submissions is available in the Appendix (Table 6). The mean security score achieved was 5.86 (min: 2, max: 7, median: 6). Ten of the 25 participants used Argon2 to store the passwords and thus, achieved 7 points on the security score. Five of the 25 participants used bcrypt and achieved 6 points. The remaining 10 of 25 participants submitted solutions below 6 points, like PBKDF2 with insufficient parameters, SHA-1, or MD5.

In the follow-up survey—independently whether participants received SecRequest-G—we asked participants whether they know about and have experience with NIST or OWASP sources for user password storage. 58% (25 of 43) of all participants reported that they had heard of NIST or OWASP before. 47% (20 of 43) participants stated to have followed one (or both) of the guidelines in their submissions (including participants without SecRequest-G). Of the 25 participants who received SecRequest-G, 15 reported to have heard of NIST or OWASP. Consequently, 10 participants received instructions from us with NIST and OWASP sources, but indicated to not know the sources.

When comparing both guidelines, NIST offers a more theoretical and complex recommendation, while OWASP offers Argon2 example code in Java with clearer recommendations. We investigated whether our participants had copied and pasted code from the OWASP guideline and found that out of 10 participants who implemented Argon2 after our second security request, 8 copied and pasted the code from the OWASP guideline. We found the same comments in the programming code as on the website.

In the following we present an analysis of participants' experience with the guidelines based on their open-question answers in the survey and the chat communication.

Guideline Experience with SecRequest-G. Our participants mentioned that reading the guidelines has helped them to increase their knowledge. Participants found the guidelines useful, as they provided them with details, they were not aware of:

> "*I was unaware of Argon2 and the weakness in PBKDF2-with-HMAC and hadn't thought of a few things that were mentioned in the guidelines (max password length for DoS protection, Unicode normalization)*" (FJP3).

FSN5 reported to like OWASP because it instructed him to use existing hashing and salting algorithms instead of implementing them himself:

> "*One of the OSWAP design principles is to keep security simple. In the registration process I avoided implementing own salt and hashing algorithms [...]. This reduces chances of making security mistakes.*"

Guideline Experience without SecRequest-G. Ten of 43 participants (21%), who did not receive SecRequest-G, reported to have heard of NIST or OWASP before. Five of them (FJN4, FJP4, FSN2, FSN12, FSP5) stated that they did not follow the guidelines of the organizations in our task. As reasons FJP4 noted that he heard of the organizations, but never implemented their guidelines before and FSP5 stated:

		Company Freelancer [28]				Study Freelancer (only JSF)				Study Freelancer (JSF and Spring)			
		Non-secure	Secure	Score	Total	Non-secure	Secure	Score	Total	Non-secure	Secure	Score	Total
Initial submission	Prompting	8	13	$\mu = 2.19$ ($\sigma = 2.52$) min = 0, max = 6	21	5	6	$\mu = 2.27$ ($\sigma = 2.8$) min = 0, max = 6	11	7	15	$\mu = 3.32$ ($\sigma = 2.8$) min = 0, max = 6	22
	Non-Prompting	17	4	$\mu = 0.86$ ($\sigma = 1.96$) min = 0, max = 6	21	6	5	$\mu = 1.91$ ($\sigma = 2.59$) min = 0, max = 6	11	16	5	$\mu = 1$ ($\sigma = 2.07$) min = 0, max = 6	21
	Total	25	17	$\mu = 1.52$ ($\sigma = 2.33$)	42	11	11	$\mu = 2.09$ ($\sigma = 2.64$)	22	23	20	$\mu = 2.19$ ($\sigma = 2.68$)	43
After SecRequest-P	Prompting	2	1	$\mu = 2$ ($\sigma = 3.46$) min = 0, max = 6	3	0	1	$\mu = 1$ ($\sigma = 0$) min = 1, max = 1	1	0	2	$\mu = 3.5$ ($\sigma = 3.54$) min = 1, max = 6	2
	Non-Prompting	4	10	$\mu = 2$ ($\sigma = 2.29$) min = 0, max = 6	14	0	6	$\mu = 2.17$ ($\sigma = 1.94$) min = 1, max = 6	6	1	14	$\mu = 3.13$ ($\sigma = 2.36$) min = 0, max = 6	15
	Total	6	11	$\mu = 2$ ($\sigma = 2.33$)	17	0	7	$\mu = 2$ ($\sigma = 1.83$)	7	1	16	$\mu = 3.18$ ($\sigma = 2.31$)	17

Table 4: Comparison of (non-)secure solutions and the security score

"*I could develop it without these practices.*" Both used bcrypt in their initial solution and received 6 points in the security score. FSN11, FSP1, FSP3, FSP10, and FSP13 reported to have followed one of the guidelines. Four of them used bcrypt in their first submission and FSP10 used bcrypt in his second submission after SecRequest-P. FSP1 reported about NIST:

> "*NIST guidelines were easy to follow as they are in line with security best practices.*"

6.6 Sample Comparison

In this section we present a direct comparison between the previous freelancer study [28] and our replication study. In the following we denote this study as *Study Freelancer* and the original study as *Company Freelancer*, since a company deception was used as study design. Unlike in the Company Freelancer study, where only JSF was used as framework, this study also looked at Spring as a between groups condition. However, for the examination of the deception effect, we restrict our comparison to the JSF participants of our study, since only there a direct comparison can be made. Figure 2 displays a distribution of participants' initial submission security scores from the both studies Study Freelancer and Company Freelancer. For our study, Spring and JSF results are reported separately. Considering the prompted and non-prompted groups, the distributions of the obtained scores are fairly similar.

Table 4 summarizes participants' initial and SecRequest-P related security results of both the Study Freelancer and Company Freelancer. Since in the original study SecRequest-G was not introduced to participants, we do not consider it for the comparison of the both studies. Table 4 shows the count of participants from the two groups (prompted and non-prompted) and how many of the solutions were secure or non-secure. The proportions of participants in each group in both freelancer samples appear to be similar. Further, the mean score values from the company freelancers and study freelancers (only the JSF group) are similar as well.

We did not find any significant difference between the JSF security scores of the Study Scenario ($\mu = 2.09$, $\sigma = 2.64$) and Company Freelancer Scenario ($\mu = 1.52$, $\sigma = 2.33$) in the scores of the initial submissions (group: secure = 1 & group = JSF; Wilcoxon Rank sum: W = 86, $p = 0.73$, $r = 0.09$). However, a lack of a statistically significant finding does not mean there is none. With a large enough sample even small differences will result in a statistically significant finding. Power analysis is necessary to avoid Type II static errors (i.e., false negatives). In this study, power analysis would have established whether the lack of a statistically significant difference between the replicated and original results was meaningful, or if it was just an artifact of too few participants. Future studies will have to decide whether the differences shown above are worth re-examining with a larger sample size.

6.6.1 Implementation Time

We compared the implementation time to submit the initial solution of our study with the prior freelancer study [28] (min: 1 day, max: 8 days, mean: 3 days, median: 3 days, SD: 1.88). We did not find any significant difference between the implementation time of both JSF groups (Wilcoxon Rank sum: W = 457, $p = 0.83$). Further, we compared the time spent on the first security request. In [28], the participants needed on average 6.4 hours (sd 7.3 h, median 3.17 h). We did not find a significant effect in both freelancer studies either (W = 74, $p = 0.52$). The same caveats about the lack of power apply as above.

6.6.2 Study Announcement: Self-reflection

In the follow-up survey our participants were asked whether they would have stored the password securely if it had not been a study. Interestingly the answers split in half, 22 reported to would have stored the passwords securely, and 21 reported that they did not think that they would have stored them securely. Six of the 22 who reported that they would have performed differently if it was not a study, stored the

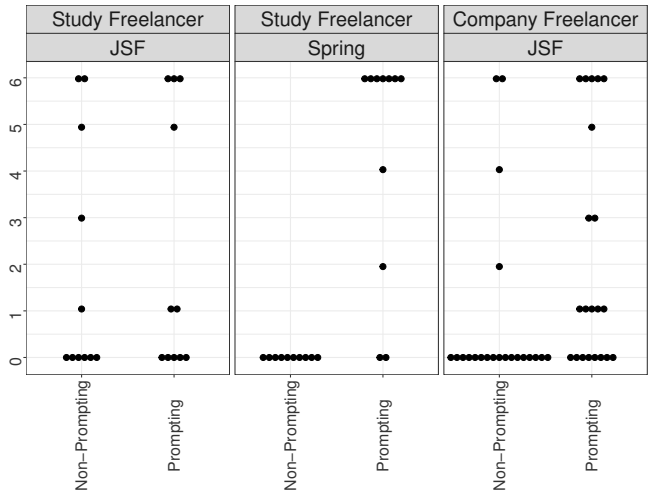

Figure 2: Initial submission score comparison of the previous freelancer study [28] (Company Freelancer) and our replication study (Study Freelancer).
Each point stands for one participant. In the Company Freelancer JSF group, Naiakshina et al. had almost twice as much participants as in the JSF group of this study, because we also investigated Spring as another value for the framework variable.

passwords in a sufficient way in the initial submissions. The results of the freelancer study of Naiakshina et al. [28] where the participants were not aware that they were participating in a study until they finished programming, shows that participants in developer studies can overestimate their own security awareness. We found that the reported details do not always fit the actual code submission.

7 Discussion

Methodological implications for developer studies: Our results suggest that freelancers are a useful sample for usable security developer studies. Response rates are higher than on GitHub. The studies can be conducted online and it is possible to reach developers from all over the world. Freelancers could provide more experience with real world projects and a wide range of age and experience. Since freelancers on the platform most likely do not know each other, the probability of participants communicating with each other about the study task and solutions is lower than for example, in a CS student sample. Additionally, freelancers are a convenient sample as they are looking for jobs and work with their own devices.

In comparison to the field study of Naiakshina et al. [28], where the study itself was concealed, our study was conducted by openly communicating the study context to the freelancers. Using the same sample size and same study protocol (minus the study deception), we got similar results as Naiakshina et al. We got similar effect sizes, directions, and statistically significant results for the same tests as they did and did not get any that they did not. In Table 5 a comparison of both studies and tests can be found. In both studies prompting lead to more security. The impact was significant in both samples, so we observed the same treatment effect (Company Freelancer: OR = 6.55 and Study Freelancer: OR = 6.51).

Neither study found a significant effect of previous password storage experience. However, in both cases only very few participants reported to have no password storage experience at all (Company Freelancer: n = 2 and Study Freelancer: n = 3). Therefore, the results should not be over-interpreted. Furthermore, Naiakshina et al. [28] did not find an effect of Java experience on security. In this study, we did not find an effect either. However, the direction of the correlation is the same in both studies. Future studies will have to decide whether to examine this effect with larger sample sizes.

In contrast to the previous freelancer study, we also tested the two frameworks Spring and JSF as done in the student study [30]. In [30], Naiakshina et al. did not find any significant difference between Spring and JSF. We did not find a significant difference between the two frameworks either. In both studies the mean security score was higher for the Spring framework but future studies with more participants would be needed to examine this effect further.

We conclude that for this study the removal of the deception element does not seem to have changed any outcomes and all relevant results gained from the original study with deception have also been gathered by this one without. Since deception should only be used when necessary, for this study we would not recommend to use it again in this specific context. While these results certainly do not generalize to all developer security studies, it is an important first indication that freelance developers recruited as part of a study behave similarly to when they are hired for a regular job. Therefore, our findings also offer an early indication that platforms such as Freelancer.com may be promising platforms for developer recruitment to supplement other channels such as CS students and GitHub developers.

Security guidelines: Acar et al. [3] showed in their programming experiment that using standard documentation without access to other sources such as the Internet lead to more secure code. However, the set-up of the experiment was rather artificial. In the real world developers use the Internet to find solutions while programming [30]. The standards are available but only a few developers use them or are even aware of them.

We found that concrete security policies with web links to the guidelines did increase the score of the password storage code. Ten participants were able to achieve full 7 points although no participants in the past studies and in the initial submissions achieved such a level for security. Even though some participants were able to use secure industry standards

IV	DV	Statistical Test	Company Freelancer [28]	Study Freelancer
Prompting	Secure	FET	p = **0.01*** OR = 6.55, CI = [1.44, 37.04]	p = **0.006*** OR = 6.51, CI = [1.51, 33.18]
Java Experience	Score	Kruskal-Wallis	p = 0.21, r = 0.12	p = 0.75, r = 0.04
Stored Passwords Before	Secure	FET	p = 0.17 OR = 0, CI = [0,3.87]	p = 0.59 OR = 0.41, CI = [0.04, 2.69]
Framework	Score	Wilcoxon Rank sum	- -	p = 0.16 group: secure = 1, r = 0.3

Table 5: Summary of all tests across the different samples
IV: Independent variable, **DV:** Dependent variable, **Company Freelancer:** Freelancers with study deception [28], **Study Freelancer:** Freelancers without study deception, **OR:** Odds ratio, **CI:** Confidence interval. The IV framework was not examined for freelancers in [28]. Significant tests are marked with *.

without being requested with the specific policies, a number of participants reported to know the guidelines. However, these participants were only able to score at most 6 points. This finding emphasizes the need for explicit encouragement of using security policies. The more support the policies offer the more secure the code can be.

Performance-based payment using milestones: Milestone payment is the recommended payment method on Freelancer.com. We split the payment into three milestones and chose the milestones for the initial submission and the (security) code review in a 1:1 ratio. It has to be investigated whether different ratios might result in different security results. Different performance rewards could help to increase security and the security awareness as well. We chose this ratio as we did not know how many security requests a participant would need as this depends on the password storage security in the initial and follow-up submission. In this study we chose the security to weight as much as the initial functional solution. We note that rewarding security performance as a study variable might lead to better initial solutions. In future research, this has to be evaluated with other scenarios and developer studies.

8 Conclusion

One major issue of usable security developer studies is their ecological validity. In the password-storage study of Naiakshina et al. [29, 30], CS students claimed that they would have behaved differently if they would have worked for a company. In a follow-up study, Naiakshina et al. [28] concealed the study context and hired freelancers for the same project. This form of deception requires additional study design work to maintain the deception. Frequent use of deception can cause problems as well. Both these issues make the use of deception for this kind of developer study something one would want to avoid if possible. Therefore, we replicated the deception study of Naiakshina et al. [28] without deception and compared the results. Overall the results were very similar leading us to propose the following recommendations:

- **When trying to get ecologically valid results for freelance developers, deception is not always necessary.** In our example running the study openly produced very similar outcomes compared to hiring freelancers for real. We got similar effect sizes and directions for the same tests as Naiakshina et al. in [28]. However, our study presents only one data point and further research is needed on the use of deception in other studies covering other security issues, tasks and scenarios, as well as with other types of developers.

- **Instruct developers to use security-guidelines.** We found that providing participants with specific guidelines for password storage can increase the security of their solutions drastically. Almost all our participants were able to implement secure password storage after being provided with specific security guidelines. If guidelines offer code examples, they are more likely to be implemented and included into the developers' code. Thus, we recommend designers of security guidelines to give specific code examples of secure code. We further recommend organizations to provide developers with specific security guidelines to receive software with state-of-the-art security standards. If guidelines might not be known to the employer, we recommend to include at least security prompting in the task to raise security awareness.

9 Acknowledgments

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

References

[1] Open web application security project (owasp). https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Password_Storage_Cheat_Sheet.md. Last Accessed: November 14, 2019.

[2] Yasemin Acar, Michael Backes, Sascha Fahl, Simson Garfinkel, Doowon Kim, Michelle L Mazurek, and Christian Stransky. Comparing the usability of cryptographic apis. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 154–171, San Jose, CA, USA, 2017. IEEE, IEEE.

[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 *2016 IEEE Symposium on Security and Privacy (SP)*, pages 289–305, Piscataway, NJ, USA, May 2016. IEEE Press.

[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, Santa Clara, CA, 2017. USENIX Association.

[6] Malik Aleem Ahmed and Jeroen van den Hoven. Agents of responsibility—freelance web developers in web applications development. *Information Systems Frontiers*, 12(4):415–424, Sep 2010.

[7] Rebecca Balebako, Abigail Marsh, Jialiu Lin, Jason I Hong, and Lorrie Faith Cranor. The privacy and security behaviors of smartphone app developers. 2014.

[8] Titus Barik, Justin Smith, Kevin Lubick, Elisabeth Holmes, Jing Feng, Emerson Murphy-Hill, and Chris Parnin. Do developers read compiler error messages? In *Proceedings of the 39th International Conference on Software Engineering*, ICSE '17, pages 575–585, Piscataway, NJ, USA, 2017. IEEE Press.

[9] Jason Bau, Frank Wang, Elie Bursztein, Patrick Mutchler, and John C Mitchell. Vulnerability factors in new web applications: Audit tools, developer selection & languages. *Stanford, Tech. Rep*, 2012.

[10] Douglas G Bonett. Replication-extension studies. *Current Directions in Psychological Science*, 21(6):409–412, 2012.

[11] J. Bonneau. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *2012 IEEE Symposium on Security and Privacy*, pages 538–552, San Francisco, CA, USA, May 2012. IEEE.

[12] Joseph Bonneau and Sören Preibusch. The password thicket: Technical and market failures in human authentication on the web. In *WEIS*, 2010.

[13] 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.

[14] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[15] 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.

[16] Serge Egelman, Andreas Sotirakopoulos, Ildar Muslukhov, Konstantin Beznosov, and Cormac Herley. Does my password go up to eleven?: The impact of password meters on password selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2379–2388, New York, NY, USA, 2013. ACM, ACM.

[17] Sascha Fahl, Marian Harbach, Yasemin Acar, and Matthew Smith. On the ecological validity of a password study. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, page 13. ACM, 2013.

[18] Matthew Finifter and David Wagner. Exploring the relationship between web application development tools and security. In *Proceedings of the 2Nd USENIX Conference on Web Application Development*, WebApps'11, pages 9–9, Berkeley, CA, USA, 2011. USENIX Association.

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

[20] 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, Baltimore, MD, 2018. USENIX Association.

[21] Paul A Grassi, James L Fenton, EM Newton, RA Perlner, AR Regenscheid, WE Burr, JP Richer, NB Lefkovitz, JM Danker, Yee-Yin Choong, et al. Nist special publication 800-63b: Digital identity guidelines. *Enrollment and Identity Proofing Requirements*. https://pages.nist.gov/800-63-3/sp800-63b.html, 2017.

[22] Ameya Hanamsagar, Simon S Woo, Chris Kanich, and Jelena Mirkovic. Leveraging semantic transformation to investigate password habits and their causes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 570, New York, NY, USA, 2018. ACM, ACM.

[23] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. Of passwords and people: Measuring the effect of password-composition policies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2595–2604, New York, NY, USA, 2011. ACM, ACM.

[24] 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.

[25] 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.

[26] Peter Mayer, Jan Kirchner, and Melanie Volkamer. A second look at password composition policies in the wild: Comparing samples from 2010 and 2016. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, pages 13–28, Santa Clara, CA, 2017. USENIX Association.

[27] Michelle L. Mazurek, Saranga Komanduri, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley, Richard Shay, and Blase Ur. Measuring password guessability for an entire university. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 173–186, New York, NY, USA, 2013. ACM.

[28] 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.

[29] 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.

[30] 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, USA, 2018. USENIX Association.

[31] Duc Cuong Nguyen, Dominik Wermke, Yasemin Acar, Michael Backes, Charles Weir, and Sascha Fahl. A stitch in time: Supporting android developers in writingsecure code. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1065–1077, New York, NY, USA, 2017. ACM.

[32] Lutz Prechelt. Plat_forms: A web development platform comparison by an exploratory experiment searching for emergent platform properties. *IEEE Transactions on Software Engineering*, 37(1):95–108, Jan 2011.

[33] Elissa M Redmiles, Ziyun Zhu, Sean Kross, Dhruv Kuchhal, Tudor Dumitras, and Michelle L Mazurek. Asking for a friend: Evaluating response biases in security user studies. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1238–1255. ACM, 2018.

[34] Sean M Segreti, William Melicher, Saranga Komanduri, Darya Melicher, Richard Shay, Blase Ur, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Michelle L Mazurek. Diversify to survive: Making passwords stronger with adaptive policies. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS)*, pages 1–12, Santa Clara, CA, USA, 2017. USENIX Association.

[35] Awanthika Senarath and Nalin Asanka Gamagedara Arachchilage. Understanding software developers' approach towards implementing data minimization. *arXiv preprint arXiv:1808.01479*, 2018.

[36] Richard Shay, Saranga Komanduri, Adam L. Durity, Phillip (Seyoung) Huh, Michelle L. Mazurek, Sean M.

Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Can long passwords be secure and usable? In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 2927–2936, New York, NY, USA, 2014. ACM.

[37] Richard Shay, Saranga Komanduri, Adam L. Durity, Phillip (Seyoung) Huh, Michelle L. Mazurek, Sean M. Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Designing password policies for strength and usability. *ACM Transactions on Information and System Security (TISSEC)*, 18(4):13:1–13:34, May 2016.

[38] 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.

[39] Elizabeth Stobert and Robert Biddle. The password life cycle: User behaviour in managing passwords. In *10th Symposium On Usable Privacy and Security (SOUPS 2014)*, pages 243–255, Menlo Park, CA, USA, 2014. USENIX Association.

[40] Christian Stransky, Yasemin Acar, Duc Cuong Nguyen, Dominik Wermke, Doowon Kim, Elissa M Redmiles, Michael Backes, Simson Garfinkel, Michelle L Mazurek, and Sascha Fahl. Lessons learned from using an online platform to conduct large-scale, online controlled security experiments with software developers. In *10th USENIX Workshop on Cyber Security Experimentation and Test (CSET'17)*, 2017.

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

[42] Blase Ur, Jonathan Bees, Sean M. Segreti, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Do users' perceptions of password security match reality? In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 3748–3760, New York, NY, USA, 2016. ACM.

[43] Blase Ur, Patrick Gage Kelley, Saranga Komanduri, Joel Lee, Michael Maass, Michelle L. Mazurek, Timothy Passaro, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Serge Egelman, and Julio López. Helping users create better passwords. *USENIX*, 37(6):51–57, 2012.

[44] Blase Ur, Fumiko Noma, Jonathan Bees, Sean M. Segreti, Richard Shay, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. I added '!' at the end to make it secure: Observing password creation in the lab. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 123–140, Ottawa, 2015. USENIX Association.

[45] Rick Wash, Emilee Rader, Ruthie Berman, and Zac Wellmer. Understanding password choices: How frequently entered passwords are re-used across websites. In *Twelfth Symposium on Usable Privacy and Security (SOUPS)*, pages 175–188, Denver, CO, 2016. USENIX Association.

[46] Rick Wash, Emilee Rader, and Chris Fennell. Can people self-report security accurately?: Agreement between self-report and behavioral measures. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 2228–2232, New York, NY, USA, 2017. ACM.

[47] Chamila Wijayarathna and Nalin A. G. 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*, EASE'18, pages 205–210, New York, NY, USA, 2018. ACM.

[48] Chamila Wijayarathna and Nalin AG Arachchilage. Am i responsible for end-user's security? Baltimore, MD. USENIX Association.

[49] 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.

[50] Aiko Yamashita and Leon Moonen. Do developers care about code smells? an exploratory survey. In *Reverse Engineering (WCRE), 2013 20th Working Conference on*, pages 242–251. IEEE, IEEE, 2013.

[51] Aiko Yamashita and Leon Moonen. Surveying developer knowledge and interest in code smells through online freelance marketplaces. In *User Evaluations for Software Engineering Researchers (USER), 2013 2nd International Workshop on*, pages 5–8, San Francisco, CA, USA, 2013. IEEE, IEEE.

APPENDIX

A Security Requests

Figure 3 visualizes the security request procedure. When we were sent a solution where the password was stored in plain text or where the participant received less than 6 points in the *security score*, we sent the following messages:

- **SecRequest-P:** Participant handed in plain text code:
  R: *I saw that the password is stored in clear text. Could you also store it securely?*
- **SecRequest-G:** Security score < 6:
  R: *Thank you for submitting your solution. Now I have one further request. I noticed, that you did not follow industry best practices, e.g., NIST (National Institute of Standards and Technology) or OWASP (Open Web Application Security Project), to securely store the end-user password. Could you please revise your submission and ensure that you follow industry best practices? You can find some information on OWASP on this website:* `https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Password_Storage_Cheat_Sheet.md` *and information on NIST on this website:* `https://pages.nist.gov/800-63-3/sp800-63b.html` *in section 5.1.1.2.*

B Security Scale

We based the evaluation of participants' submissions on the security score of Naiakshina et al. [29]:

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).

C Hypotheses

Due to the adjusted study design, we were able to test only four of the seven main hypotheses from [30]. While it was possible to track security attempts in a lab setting, in an online study this information was not accessible. We did, though, consider the subset *secure = 1 (achieving security)* for our analysis. Hypotheses from [30]:

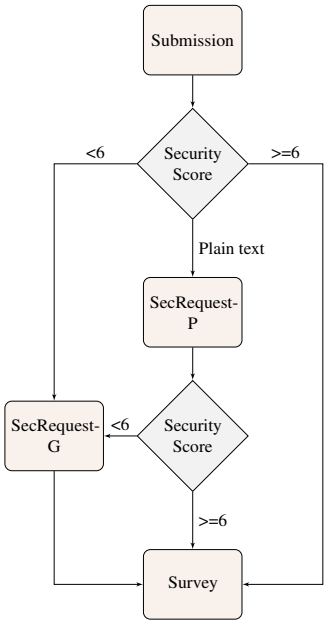

Figure 3: Security request procedure
Participants received SecRequest-P, if they submitted a plain text solution and SecRequest-G if the solution scored less than 6 points. Participants with plain text solutions thus, could receive both requests.

- H-P1 - Priming has an effect on the likelihood of participants attempting security.
- H-F1 - Framework has an effect on the security score of participants attempting security.
- H-G1 - Years of Java experience have an effect on the security scores.
- H-G2 - If participants state that they have previously stored passwords, it affects the likelihood that they store them securely.

D Summary of Statistical Analysis

Table 7 summarizes all hypotheses from our analysis of the study.

E Playbook

We used the same playbook Naiakshina et al. used in [28]. We extended and adapted it by several relevant aspects. **P** indicates the participant and **R** indicates the researcher. Because of space limitation, we mention only playbook extensions here.

Participant	Prompting	Framework	Payment	Include SecRequest-G	Function	Length in bits	Iteration	Salt	Secure	Score	Copied	NIST	OWASP
FJN1	0	JSF	120	1 Day	SHA-1	160	1		1	2			✓
FJN3	0	JSF	120	1 Day	bcrypt	184	2^{10}	SR	1	6			
FJN5	0	JSF	220	50 min	PBKDF2 (SHA-1)	256	10 000	St	1	5	✓	✓	
FJN7	0	JSF	120	1 Day	MD5	128	1	SR	1	4			
FJN8	0	JSF	120	2 Days	PBKDF2 (SHA-512)	512	65 536	St	1	5			✓
FJN9	0	JSF	220	19 Days	Argon2i	256	2	SR	1	7	✓	✓	
FJN10	0	JSF	220	1h 30min	PBKDF2 (SHA-1)	128	65 536	SR	1	5			✓
FJN12	0	JSF	220	15min	Argon2i	256	10	SR	1	7			✓
FJP2	1	JSF	120	2 Days	PBKDF2 (SHA-1)	512	1 000	SR	1	5.5			
FJP3	1	JSF	120	5h 30min	Argon2i	256	20	SR	1	7			✓
FJP6	1	JSF	120	2 Days	Argon2i	256	40	SR	1	7	✓		
FJP7	1	JSF	120	35min	PBKDF2 (SHA-1)	128	4	SR	1	4	✓	✓	✓
FJP8	1	JSF	120	3h 30min	PBKDF2 (SHA-1)	512	1 000	SR	1	5.5	✓	✓	
FJP9	1	JSF	120	2 Days	PBKDF2 (SHA-1)	512	1 000	SR	1	5.5	✓	✓	
FJP10	1	JSF	120	1 Day	Argon2i	256	40	SR	1	7	✓		
FJP11	1	JSF	220	1 Day	bcrypt	184	2^{12}	SR	1	6			
FSN1	0	Spring	120	2 Days	Argon2i	256	20	SR	1	7	✓	✓	✓
FSN3	0	Spring	120	6 Days	Argon2i	256	4	SR	1	7	✓		✓
FSN5	0	Spring	120	4 Days	bcrypt	184	2^{10}	SR	1	6			✓
FSN7	0	Spring	120	1 Day	Argon2i	256	40	SR	1	7	✓		
FSN9	0	Spring	220	1 Day	bcrypt	184	2^{10}	SR	1	6		✓	✓
FSN10	0	Spring	120	1h	Argon2i	256	40	SR	1	7	✓	✓	
FSP2	1	Spring	120	10h	PBKDF2 (SHA-1)	128	65 536	SR	1	5			
FSP4	1	Spring	120	3 Days	bcrypt	184	2^{10}	SR	1	6			
FSP6	1	Spring	120	14h	Argon2d	128	3	SR	1	7	✓		

Table 6: Evaluation of participants' submissions after SecRequest-G

Include SecRequest-G: Time participants needed to add security after SecRequest-G. **Salt:** SR = SecureRandom, R = Random, St = Static. **Copied:** Security code was probably copied and pasted from the Internet. **NIST/OWASP:** Participant stated that he/she followed the security guidelines of NIST/OWASP programming the task.

H	Sub-sample	IV	DV	Test	O.R.	CI	p-value	$cor-p$-value
H-P1#	-	Prompting	Secure	FET	6.51	[1.51, 33.18]	0.006*	0.01*
H-G1#	-	Java experience	Score	Kruskal-Wallis	-	-	0.75	0.75
H-G2#	-	Stored passwords before	Secure	FET	0.42	[0.01, 8.63]	0.59	0.59
H-F1#	secure = 1	Framework	Score	Wilcoxon rank sum	-	-	0.16	0.32
E-A1	secure = 1	Java experience	Score	Kruskal-Wallis	-	-	0.80	-
E-A2	-	Framework	API usability	Wilcoxon rank sum	-	-	0.08	-
E-A3	-	Score	API usability	Pearson Cor.	-	[-0.21, 0.38]	0.57	-
S-C1	secure = 1& group = JSF	Study sample	Score	Wilcoxon rank sum	-	-	0.73	-
S-C2	group = JSF	Study sample	Implementation time initial submission	Wilcoxon rank sum	-	-	0.83	-
S-C3	group = JSF	Study sample	Implementation time time for SecRequest-P	Wilcoxon rank sum	-	-	0.52	-

Table 7: Summary of statistical analysis

IV: Independent variable, **DV**: Dependent variable, **O.R.**.: Odds ratio, **CI**: Confidence interval, **E-A**: Exploratory analysis, **S-C**: Sample Comparison. All tests were conducted on security values of the *initial solutions* before participants received any security requests. H-P1 and H-G2 as well as H-G1 and H-F1 are corrected with Bonferroni-Holm correction (cor-p value). # = Hypothesis of Naiakshina et al. [28, 30], * = Significant Tests.

E.1 Study Announcement and Study Offer

We are researchers from the University of Bonn working in the field of software usability. We are always looking for software developers and system administrators who are interested in taking part in one of our studies (programming and surveys). All data will be processed pseudonymously and stored anonymized after the study; there will be no identifying information published in any form. If you are interested in participating in studies - Please contact us!

After the participants placed a bid on the project, we contacted them via the private chat at Freelancer.com with the following message:

R: *Hello XYZ, we are happy that you want to take part in our study. The payment will be divided into 3 milestones: 50 euros (100 euros) for your initial code release, additional 50 euros (100 euros) for the final code release after our review and further additional 20 euros for completing our survey about your programming experience and your experiences with this task. If this is fine for you and you want to take part in the study, we need you to sign a consent form. Please go to the following website to do so: LINK Your study-ID is: XXX Thanks in advance. Kind regards, ...*

When the freelancer signed the consent form, we sent the second message and a ZIP file with task and code:

R: *Hello XYZ, you agreed to take part in our study. Thank you for signing the consent form! Now I will send you the code as a ZIP file. You will find the task in there too. Please have a look at it and tell me if you want to do it. Then I will award you with the project and create the milestones. Kind regards, ...*

After that message, we got mostly three kinds of reactions:

- The freelancer agreed to take part, we awarded him/her

and wished him/her *Happy coding!*

- The freelancer did not react anymore:
  Hello XYZ, what do you think? Are you still interested to take part in the study?
- **P**: *May I check the code and get back to you tomorrow/later/. . . ?*
  R: *Yes, sure! Take your time.*

E.2 Deadline

Some of the participants asked us for a final deadline. Since we did not want to rush them, we did not set one, but asked them to tell us how much time they needed to finish it. When a freelancer did not ask for a deadline, we decided to contact him/her after 10 days to ask for an update.
We had two cases where the freelancers did not answer several questions for updates. In that case we set a deadline and ended the study, when they exceeded it.

- **P**: *What is the final delivery?*
  R: *What do you think how much time you need to solve the task?*
  P: DATE
  R: *Ok, that's fine. Thank you.*
- Deadline exceeded:
  R: *Hey XYZ, could you give us a status update? Kind regards, . . .*
- 10 days after task was sent and in case no deadline was set:
  R: *Hey XYZ, could you give us a status update? What do you think how much time you need to solve the task?*
- If no reaction after 3 weeks:
  R: *Hello, please send your solution till* date in one week. *If you decided to no longer participate in the study, I would be very glad if you could let me know. Thank you and kind regards, . . .*

E.3 Password-related Questions

The following questions concerned password storage:

- Before submitting initial solution:
  P: *Should I implement security/secure password storage?*
  R: *Whatever you would recommend!*
- **P**: *Is *** fine?*
  R: *Whatever you would recommend/use!*
- **P**: *I cannot find password encryption in the requirements, can you tell me where it is written? I might have missed it.*
  R: *It is not in there, that is true. But could you add it?*
- After SecRequest-G:
  P: *Should I implement all the rules mentioned in NIST document? There are so many rules in NIST.*
  R: *You don't have to implement all rules, but please concentrate on secure password storage in a database on the back-end site. For us it's most important that the password is saved securely.*

E.4 General Questions

Also in the general communication we often received similar questions.

- **P**: *Can I build it from scratch?*
  R: *You can solve the task as you prefer.*
- **P**: *If I have some questions for your project, can I ask you?*
  R: *Sure!*
- **P**: *Do you have a server where we can upload this code for you to test?*
  R: *None that I have access to. Would it be possible for you to send me a video or screenshots so I can see that it is working on your computer?*
- Participant did not work on our database:
  R: *Could you also make it work on our database?*
- **P**: *Once the user registers, do we need to send verification email also and once he clicks on that, we will make user status as active?*
  R: *No, we only need the data to be stored in our data base for now!*
- **P**: *I need to see the ER diagram.*
  R: *We do not have that yet. Is it necessary?*
  P: . . .
  R: *Could you please create a single table for now and I will talk to my mentor about the rest?*
- **P**: *Do you require the login functionality as well? Should I implement is as a further task? What else except registration will be needed?*
  R: *No, thank you. Please only program the registration functionalities.*
- **P**: *The password is in the database, so users won't be able to access it.*
  R: *And what if someone gets access to the database?*
- **P**: *Could you tell me what (. . .) is for? / Could you help me with (. . .)?*
  R: *Since we are conducting a study and all participants should have the same requirements, I cannot help you with specific questions about the code. I'm sorry!*

- Participant is confused (after SecRequest-G):
  R: *You will not receive any further request. You can choose, which industry standard you would like to follow; OWASP or NIST. So that you do not have to read the whole NIST guideline, you can read all necessary information in section 5.1.1.2. This section approximately complies with the length of the provided OWASP source. It is up to you to choose one standard. Afterwards you only have to fill out a subsequent survey, which concludes the study.*

E.5 Receiving the Solution and Survey Request

After receiving the final solution, we wrote:
Thank you, for sending your result! I will look into it.
We checked the remote database for code examples. If the freelancer had not worked on it, we wrote: *Could you also make it work on our database?*
If the freelancer could not make it work on our database, we asked for pictures and a video that showed that the code was working. We also checked it for functionality and if everything worked, we asked the freelancers to take part in our survey.
Message: *Hello XYZ, thank you for sending us your results. Like announced in the study description we would like to invite you to a concluding survey. You can find it here: LINK Kind regards, . . .*

E.6 Exit Communication

After the freelancers finished the survey, we released the last milestone and sent them the following message: *Thank you for your participation! We are happy about your feedback, but we would like to kindly ask you to not mention our study content in order to ensure the validity of our study. Thank you again!* And they all replied that they would not mention it.
Many of the freelancers asked for a good or a five-star review, which we gave them: *Yes, we did. It was nice working with you.*
Also many of them asked, if we had further projects in which they could take part.
R: *At the moment we unfortunately have only one project.*

E.7 Review

We gave all participants the same review:
Very good communication, delivered on time. It was nice working with him/her!