Awareness of Secure Coding Guidelines in the Industry - A first data analysis

Tiago Espinha Gasiba
Siemens AG
Munich, Germany
tiago.gasiba@siemens.com

Ulrike Lechner
Universität der Bundeswehr München
Munich, Germany
ulrike.lechner@unibw.de

Maria Pinto-Albuquerque
Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR-IUL
Lisboa, Portugal
maria.albuquerque@iscte-iul.pt

Daniel Mendez Fernandez
Blekinge Institute of Technology
Karlskrona, Sweden
daniel.mendez@bth.se

***Abstract*—Software needs to be secure, in particular, when deployed to critical infrastructures. Secure coding guidelines capture practices in industrial software engineering to ensure the security of code. This study aims to assess the level of awareness of secure coding in industrial software engineering, the skills of software developers to spot weaknesses in software code, avoid them, and the organizational support to adhere to coding guidelines. The approach draws on well-established theories of policy compliance, neutralization theory, and security-related stress and the authors' many years of experience in industrial software engineering and on lessons identified from training secure coding in the industry. The paper presents the questionnaire design for the online survey and the first analysis of data from the pilot study.**

***Index Terms*—Security, Secure Coding, Software Development, Best Practices, Security Awareness, Industry**

I. Introduction

Software for critical infrastructures needs to be secure. The cyber-attacks to various industry sectors [1] are ever-increasing in their number, variety, and consequences. These consequences range from loss-of-life, loss-of-business (e.g. through service disruption), loss-of-confidential data (through sensitive information leakage), data-compromise (though unauthorized changes) and monetary losses [2]. This problem is made clear through the number of alerts and advisories issued yearly by the Industrial Control System - Computer Emergency Team (ICS-CERT) from the United States Department of Homeland Security [3]. Before 2014 less than 100 advisories per year have been issued, while from 2017 to 2019 more than 200 advisories per year have been issued. This increase underpins the need for secure coding practices for software deployed to industrial (critical) infrastructures. Another argument for more efforts to increase the security of code is the Common Vulnerabilities and Exposures (CVE) [1] online database, with the relationship between existing vulnerabilities and Common Weaknesses Enumeration (CWE) [4]. This online database shows that the number of known vulnerabilities has more than doubled from 2016 to 2017 (from 6447 to 14714). Also, on average, about 2000 new vulnerabilities per year have been added since then. In a recent (2019) GitLab survey with more than 4000 software developers, Patel et al. [5] found that less than 50% of software developers can identify secure coding vulnerabilities. Secure coding based on secure coding guidelines (SCG) is one way to create secure code in industrial software engineering.

The study at hand is part of a design research effort: designing a serious game to raise awareness for secure coding and increase knowledge of secure coding guidelines among software developers [6]. The plan is to collect data on awareness of vulnerabilities and training levels of industrial software developers and, in particular, on why do software developers comply with secure coding guidelines. In this study, we are interested in understanding how well secure software development practices are established in the industry, the factors that influence whether industrial software engineers comply with secure coding guidelines, and the knowledge of software developers of secure coding guidelines.

The survey design is based on theories about security policy compliance [7], [8], security-related stress [9], neutralization theory [10] and information on typical vulnerabilities. Furthermore, it is based on the first authors' experience in teaching secure coding in different programming languages and vulnerabilities in industrial software engineering.

The results of the survey will impact both theory and practice. By understanding the reasons that lead to a lack of awareness by software developers, it is possible to better tailor a serious game [11] and motive the overall research approach of designing a serious game for secure coding training for industrial software engineers. This paper presents and motivates our survey design and first data analysis of data collected in the extensive process to develop the questionnaire. We focus our results on the core questions that remained stable throughout the survey's development, and that made to the final survey.

This paper is organized as follows. In section II, we present related work. Section III gives details the research design. This section also describes how the survey was piloted in three different companies. Section IV shows the results of our research. In particular, it gives details on the selected theories for the survey, contributed questions, and offers an outline of the survey design. Furthermore, it presents preliminary results from the survey piloting. Lastly, it presents a comparison with previous work, threats to validity, and shortly discusses the work's impact. Section V concludes this paper with a summary of the present study and outlines further work.

II. Related work

Several previous studies have indicated that software developers lack secure programming skills. In 2020, Bruce Schneier, a well-known security researcher and evangelist has stated that less than 50% of software developers can spot security vulnerabilities in software [12] - this is, unfortunately, not a new trend. In 2011, Xie et al. [13] did several interviews with 15 senior professional software developers in the industry with an average of 12 years of experience. Their study has shown a disconnect between software security concepts and the role that the participants have in their jobs.

There are various studies on the search processes and quality of information on secure coding topics online, e.g., on platforms such as stack-overflow. In the large scale study, Yang et al. [14] identify questions related to security that software developers ask on the platform stack-overflow.com. They conclude that software developers' questions could be categorized in several different topics, whereby they found more than 600 different items for every problem. Fisher et al [15] have shown that typical online platforms that software developers use to clarify development questions can be considered harmful, as the answers present in such platforms are not curated for correctness of security. Their work indicates that severe problems can arise if software developers use these references and are not aware of secure coding practices. Acar et al. [16] did an extensive analysis of existing online resources that software developers can access to get information on secure programming issues. They found out that the quality of information is not guaranteed due to, e.g., outdated information, no concrete examples or exercises. The analyses illustrate that software developers need secure coding skills, as software developers cannot depend on the well-known online sources for secure coding topics.

Gasiba et al. [6], [17] propose a method based on Capture-the-Flag events to train secure programming for software developers in the industry. Votipka et al. [18] also discuss Capture-the-Flag events as a means to improve secure software development. In [19], Davis et al. discuss the benefits of Capture-the-Flag (CTF) for software developers and Graziotin et al. [20] argue that *happy developers are better coders*, i.e., write higher quality code [21].

There are very few empirical results on the extent to which software developers comply with secure coding guidelines and practices. A notable recent study by Assal et al. [22] analyses how developers influence and are influenced by secure coding processes. They conclude that software developers are *not the weakest link* and are very motivated towards software security. However, they do not cover deeply the reasons that lead software developers to comply or not comply to secure coding guidelines.

In the present work, the concept of awareness or IT-security awareness is used to conceptualize the knowledge or skills in the IT-security domain. Benenson et al. provide a literature review on IT-security awareness [23]. Other conceptualizations that address compliance with IT-security policies or guidelines are given by Bulgurcu et al. [8]. The Unified Model by Moody et al. [7] synthesizes research on security policy compliance. Siponen et al. [10] address possible reasons why software developers might discard the usage of secure coding guidelines. Finally, D'Arcy et al. [9] use coping theory to explore the relationship between stress and deliberate policy violations.

III. Research Design

To assess the usage of secure coding guidelines in an industrial setting, we designed a survey aimed at software developers in the industry. The questionnaire is designed under the banner of three research questions:

RQ1: How well established are secure software development practices in the industry?

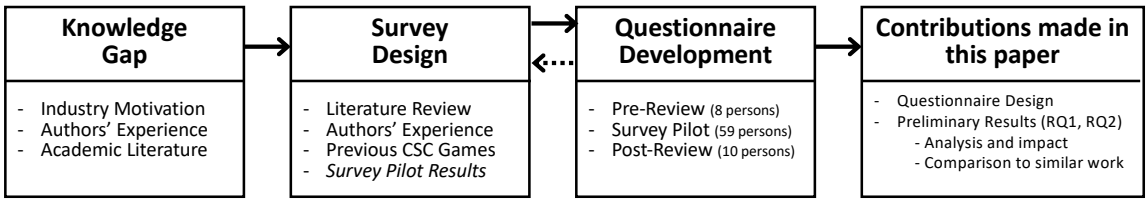
RQ2: Which factors lead software developers to use or ignore secure coding guidelines?

RQ3: How well are software developers aware of secure coding guidelines?

These questions are driven by the industry's emerging need to identify factors that lead software developers to ignore secure coding guidelines in daily work and to which no answers can currently be found in previous scientific work. Fig. 1 shows a summary of the systematic steps that we took in order to achieve this. These steps are the following: 1) definition of the research goals through the knowledge gap, 2) survey design, 3) questionnaire development, and 4) contributions made in this paper. In the following, we give details on each of these steps.

Our design of the questionnaire on secure coding guidelines draws from the first author's experience in industrial software engineering, from published experience reports in survey design [24], and data collected in the context of a serious game [11]. This information has been used for crafting the questions on knowledge on weaknesses in relevant programming languages.

We use the concept of IT security awareness with the three dimensions of knowledge following [23]: (1) recognize the threat, (2) know solutions to the threat, and (3) act right. For the factors that influence compliance with secure coding guidelines, we use from the literature on IT security compliance [7], [8], neutralization theory [10] and security-related stress [9]. In the development of the questionnaire, we took an iterative approach with first a pre-review to eliminate simple errors and review the questionnaire structure. In this phase, we have collected review comments from 8 security experts that covered, in particular, the relevance of the questions (in particular the ones on programming languages and typical vulnerabilities as well as on the knowledge of secure coding standards). In the subsequent pilot, we collected responses from 59 software developers from the industry. The survey pilot was administered to participants in the context of a CyberSecurity Challenges (CSC) workshop on secure coding guidelines. Each workshop had a duration of three days. Table I shows a summary of the different companies where these workshops took place and where the survey pilot results were



CSC: CyberSecurity Challenges, **RQ**: Research Question

Fig. 1. Research Process

collected. This work presents an analysis of the data collected from this phase, i.e. our results are based on the survey design's first iteration, as shown in figure 1. After this phase, an additional post-review with ten security experts took place. The goal of this additional iteration was to refine and prepare a final version of the survey for a large-scale deployment based on the preliminary results hereby presented. In particular, the preliminary results analysis led to the additional formulation of RQ3 at this stage. However, in this paper, we do not present results concerning RQ3, since the data to tackle it is not available at the time of publication, but we present the survey's final design as an outcome of our work. After completing a large-scale deployment of the final survey, the authors intend to present an in-depth analysis of these results in a forthcoming publication. The process of questionnaire development was done Q4 in 2019 to April 2020. An essential inspiration for the refinement of the questionnaire was a recently published study on the interplay between developers and software security processes by Assal et al. [22]. Dillman et al. [25] was used as a guide to the questionnaire development process. The answers to the questionnaires' questions use the well-known Likert scale [26].

The pilot study participants were comprised of 50% software developers with more than five years of work experience and 50% with an average of 3 years of work experience. The software developers work for companies that develop software for critical infrastructures. The participants came from China (51%), Turkey (37%), and Germany (12%), and the ages ranged from 25 to 60. On average, the participants have attended 1.6 training events related to security over the last five years, whereby 6 participants have attended more than five secure coding related training events over the past five years. The majority of the developers (68%) are embedded developers, working in C/C++ (62%), and 38% work regularly with other programming languages such as e.g., Java, Python. Data analysis of the pilot study was done by standard statistical methods using RStudio 1.2.5019.

IV. Results

In this section, we present the selected base theories and also the final design of our survey. We also provide a preliminary analysis of the results obtained during the pilot phase.

A. The Questionnaire

Table II summarizes theories identified in the design of the questionnaire, through literature research, and also provides a brief description of the theory which motivates the inclusion in the questionnaire. Table III gives details on the survey questions with particular emphasis on the mapping towards our research questions. In this table, the type of question indicates the used established theory or a type of contributed question. The reference and construct give details on the origin of the question. Column "survey question" presents the final adapted question.

The questionnaire comprises the following four sections: 1) demographic data, 2) secure coding awareness, 3) secure coding compliance, and 4) deterrents to compliance. The first part of the questionnaire includes general demographic questions on work experience, previous training on secure coding, the primary programming language used at work, used secure coding processes in the company, and software developed method.

The second section of the questionnaire deals with awareness for secure coding. This part is individualized according to the primary programming language selected in the first section. For each example, three questions of high-impact vulnerabilities (according to [4]) related to secure coding guidelines are presented, corresponding to Per1, Be1, and Prot1 in Table III. The theoretical background of these questions is the concept of IT security awareness, as defined by Hänsch et al. [23]. The questions deal with individual abilities to detect weaknesses in code and skills to deal with these weaknesses. This part of the questionnaire draws from the first author's experience in industrial software engineering and various serious games designed and played to train software developers in secure coding.

The third section in the questionnaire presents questions to measure the intent to comply to secure coding guidelines. The theoretical background of the questions is given by Bulgurcu et al. [8] and Moody et al. [7]. The following six constructs were selected: *self efficacy to comply* (SE-C), *intention to comply with the policy* (ITC), *general information security awareness* (GISA), *awareness* (ISPA), *response cost* (RespCost4) and *facilitating conditions* (FacCond5).

The fourth section of the questionnaire is about the factors that influence compliance with secure coding guidelines. Here we have used two different theories: neutralization theory

TABLE I
SURVEY PILOTING ACCORDING TO DELIVERED SECURE CODING WORKSHOPS

Pilot	Company	Region	Nr. of Developers	When
#1	Company A	China	30: 15 C/C++, 15 Web	Aug. 2019
#3	Company B	Germany	7: 7 Web	Sep. 2019
#3	Company C	Turkey	22: 22 C/C++	Oct. 2019

TABLE II
SELECTED THEORIES

Type	Name	Ref.	Description
PC	Policy Compliance Theory	[7], [8]	Assess to which extent do the participants intend to comply with secure coding guidelines in their organization
NT	Neutralization Theory	[10]	Determine which reasons do software developers mostly use in order not to comply with secure coding guidelines
SRS	Security-Related Stress	[9]	Determine to which extend does complying to secure coding guidelines lead to an increase of stress at work
AW	Dimensions of Awareness	[23]	Measure awareness of secure coding guidelines by the software developers on perception, behavior and protection

Ref: Literature reference

and security-related stress theory. For Neutralization Theory (NT), six constructs were selected: defense of necessity (N-DON3), appeal to higher loyalties (N-ATHL1), denial of injury (N-DOI1, N-DOI2, N-DOI3), denial of responsibility (N-DOR3), condemnation of the condemners (N-COC1, N-COC2) and metaphor of the ledger (N-MOTL1). For Security-Related Stress (SRS), the following three constructs were selected: complexity (CX2, CX4), overload (OL1, OL4), and uncertainty (UC1, UC4)

These constructs were selected based on the industry experience of the first author and practical limitations on the number of questions (to increase response rate). Additionally, based on lessons learned from teaching secure coding in the industry, we have included additional questions. These questions are of the following two types: Company Background (CBG) and Background Knowledge (BGK). The CBG questions intend to assess the established processes in the company related to secure coding guidelines. The BGK questions intend to assess background knowledge related to secure coding guidelines by the software developer.

The questions corresponding to RQ1 were included in the first section, and the ones corresponding to RQ3 were included in the second section of the questionnaire. The questions corresponding to RQ2 were split into the third and fourth sections of the survey, as detailed above. Company background questions related to RQ2 were included in the third section of the questionnaire.

B. Survey Results

Table IV shows the results of the preliminary analysis of the survey pilot. This analysis focuses on the questions that were not substantially changed in the last survey design iteration and made to the final version of the survey. Since RQ3 was formulated due to the refinements made in the post-review phase, this analysis focuses solely on RQ1 and RQ2.

1) Preliminary results for RQ1: The results in Table IV show that, since the majority of the software developers have a neutral opinion and 19.2% disagree on CBg2 (software developers know about the secure software development lifecycle used in the company they work for), we can conclude that they generally lack knowledge about the company's secure software development lifecycle. Additionally, there is an indicator that compliance with SCG is being checked (36.5%); however, the large number of neutral results also indicates that this might be an issue. Also, the vast majority of software developers recognize the importance of SCG (85.5%). Finally, although 48.2% agree on knowing the secure coding guideline policies (policy compliance, ISPA construct), a large number does not have an opinion (33.9%) or disagrees on this fact (17.9%). Therefore we conclude that many software developers lack awareness about secure coding policies since the average agreement is low. We observe that, although software developers lack awareness of secure software development practices, 85.5% claim that they are aware of their importance (background knowledge, BgK2 construct).

2) Preliminary results for RQ2: In terms of the factors that lead software developers to use secure coding guidelines (RQ2), the results in Table IV can be summarized as follows. Software developers express having enough freedom (74.9%), skills (60.3%), time (57.4%), resources (54.7%), knowledge (53.7%), and competencies (51.9%) to write secure code according to SCG. This high agreement values for freedom, time, and resources indicate that executing company processes should not be an issue. However, when considering

TABLE III
FINAL QUESTIONNAIRE CONSTRUCTS AND ADAPTED QUESTIONS

RQ.	Type	Ref.	Construct	Survey Question
RQ1	CBG	—	*CBg1*	In my company compliance to secure code guidelines is being checked in projects I work in
			CBg2	**I know the secure software development lifecycle in my company**
			CBg3	To which extent do you work with the _____ secure coding standard?
			CBg7	**How is the compliance to secure coding guidelines checked in my current project?**
			CBg8	In my company we use a well established secure software development life-cycle
	BGK		***BgK1***	**Compliance to secure coding guidelines is an important part of the development of our products**
			BgK2	Which of the following secure coding standards and best practices do you know?
	PC	[8]	**ISPA**	**I know that my company has a policy that mandates the usage of secure coding guidelines in software development**
		[7]	FacCond5	Support is available if I experience difficulties in complying with secure coding guidelines
RQ2	CBG	—	*CBg4*	Could you explain why you use secure coding guidelines when writing code for the product you currently develop?
			CBg5	Could you tell us why you do not use secure coding guidelines?
			CBg6	Why is compliance to secure coding guidelines not actively being checked in the projects I work in?
	PC		*PC-Conf*	Complying to SCG makes me feel more confident about the security of the code that I write
			PC-NT	**In my opinion, to write secure code, I have the necessary time**
			PC-NR	**In my opinion, to write secure code, I have the necessary resources**
			PC-NF	**In my opinion, to write secure code, I have the necessary freedom**
		[8]	**SE-C1**	**In my opinion, to write secure code, I have the necessary skills**
			SE-C2	**In my opinion, to write secure code, I have the necessary knowledge**
			SE-C3	**In my opinion, to write secure code, I have the necessary competency**
			ITC	**I intend to always comply with secure coding guidelines**
			GISA	**I am aware of the existing security threats to the products of my company**
		[7]	**RespCost4**	**Secure coding guidelines make the task of writing software more difficult**
	NT	[10]	**N-DON3**	**It's OK to disregard secure coding guidelines when this means that I deliver my work-packages faster**
			N-ATHL1	**It's OK to disregard secure coding guidelines when I would otherwise not get my job done**
			N-DOI1	**It's OK to disregard secure coding guidelines when this would result in no harm to the customer**
			N-DOR3	It's OK to disregard secure coding guidelines if you do not understand them
			N-DOI2	It's OK to disregard secure coding guidelines if no damage is done to the company
			N-COC1	It's not as wrong to ignore secure coding guidelines that are not reasonable
			N-COC2	It's not as wrong to ignore secure coding guidelines that require too much time to comply with
			N-MOTL1	I feel my general adherence to secure coding guidelines compensates for occasionally them
		—	***NT-MArc***	**It's OK to disregard secure coding practices when this would lead to major architectural changes**
			NT-CH	It's OK to disregard secure coding guidelines when this means that it makes my customers happy
			NT-SC	It's OK to disregard secure coding guidelines if the software is not safety critical
	SRS	[9]	CX2	I find that new employees often know more about secure coding than I do
			CX4	I often find it difficult to understand my organization's security coding guidelines
			OL1	Complying to secure coding guidelines force me to do more work than I can handle
			OL4	I am forced to change my work habits to adapt to my organization's secure coding guidelines
			UC1	There are constant changes in secure coding guidelines my organization
			UC4	There are constant changes in security-related technologies in my organization
RQ3	BGK	—	*BgK4*	What other weaknesses do you pay attention to in developing software for the product you currently work for?
			*BgK5**	I know about this vulnerability
			BgK3	I am aware of negative consequences resulting from exploiting vulnerabilities on the products I deliver software for
	AW	[23]	*Per1**	I can recognize code that contains this weakness
			*Be1**	I know how to write code that does not contain this weakness
			*Prot1**	I understand the possible consequences that can result from exploiting this weakness

RQ.: Research Question, **CBG**: Company Background, **BGK**: Participant Background Knowledge, **PC**: Policy Compliance Theory, **NT**: Neutralization Theory, **SRS**: Security-Related-Stress Theory, **AW**: Awareness, Note: constructs marked with * are relative to specific software weaknesses
Note: results for the highlighted constructs and questions, which were obtained during the preliminary survey, are presented in the results section

TABLE IV
RESULTS OF PRELIMINARY SURVEY BASED ON A SUBSET OF SURVEY QUESTIONS

RQ.	Type	Construct	Summary of Question	SD	D	N	A	SA	SD+D	N	A+SA
RQ1	CBG	*CBg2*	known S-SDLC in the company	3.8	15.4	50.0	28.8	1.9	19.2	50.0	30.7
		CBg7	SCG being actively checked	3.8	21.2	38.5	32.7	3.8	25.0	38.5	36.5
	BGK	*BgK1*	importance of SCG	0.0	3.6	10.9	69.1	16.4	3.6	10.9	85.5
	PC	ISPA	know policies	3.6	14.3	33.9	39.3	8.9	17.9	33.9	48.2
RQ2	PC	*PC-NT*	I have necessary time	0.0	18.5	24.1	50.0	7.4	18.5	24.1	57.4
		PC-NR	I have necessary resources	0.0	13.2	32.1	52.8	1.9	13.2	32.1	54.7
		PC-NF	I have necessary freedom	1.9	11.1	11.1	57.4	18.5	13.0	11.1	75.9
		SE-C1	I have necessary skills	0.0	7.5	32.1	50.9	9.4	7.5	32.1	60.3
		SE-C2	I have necessary knowledge	0.0	13.0	33.3	46.3	7.4	13.0	33.3	53.7
		SE-C3	I have necessary competencies	0.0	7.7	40.4	44.2	7.7	7.7	40.4	51.9
		ITC	intent to comply	0.0	1.8	21.4	57.1	19.6	1.8	21.4	76.7
		GISA	know security threats	0.0	14.3	25.0	46.4	14.3	14.3	25.0	60.7
		RespCost4	job more difficult	0.0	30.9	30.9	36.4	1.8	30.9	30.9	38.2
	NT	N-DON3	deliver faster	24.1	51.9	14.8	9.2	0.0	76.0	14.8	9.2
		N-ATHL1	not get job done	18.5	38.9	33.3	9.3	0.0	57.4	33.3	9.3
		N-DOI1	no harm to customer	14.5	25.5	27.3	21.8	10.9	40.0	27.3	32.7
		NT-MArc	major architectural changes	18.5	33.3	29.6	13.0	5.6	51.8	29.6	18.6

RQ: Research Question, **CBG**: Company Background, **PC**: Background Knowledge, **PC**: Policy Compliance Theory, **NT**: Neutralization Theory, **SD**: Strongly Disagree, **D**: Disagree, **N**: Neutral, **A**: Agree, **SA**: Strongly Agree
Results show the average percentage of agreement to each individual likert scale point

the items skills, knowledge, and competencies, these have lower values. Additionally, software developers express the intention to comply (ITC) with SCG (76.6%) and also express knowledge about possible threats to the products from the company (60.9%). However, software developers are not sure if complying with SCG makes the daily job more difficult.

In terms of SE-C2 (knowledge), the results show that 53.7% of the software developers do not know secure coding guidelines. This value correlates very well with the results by Patel et al. [5], where they found out that more than 50% of software developers cannot spot security vulnerabilities in code.

These facts indicate a need for awareness training in secure coding and the application of secure coding guidelines. Furthermore, the results show that 76.6% express the intention to comply with secure coding guidelines and their policies and express knowledge about possible threats to the company (60.7%). The combined results indicate that secure coding is not a matter of governance but awareness. Additionally, the results in Table IV show that software developers are not sure (30.9%) if complying with SCG makes the daily job more difficult (38.2%) or not (30.9%). These results can be interpreted as a lack of awareness and, therefore, lack of good sense to evaluate the task's difficulty.

In terms of the factors that lead software developers not to comply with secure coding guidelines, the neutralization theory results are shown in the following. The majority of software developers (76%) disagree that SCG should be overlooked to deliver software faster. Although there is a higher uncertainty than in the previous case, software developers also disagree that secure coding guidelines should not be ignored to get a job done. However, 33% have an ambiguous opinion on this matter, which is surprising since most software developers agree on the importance of SCG. Although still with 51.8% agreement by the developers, the numbers express a higher uncertainty on disregarding SCG if these mean significant architectural changes. Finally, software developers do not have a definite opinion (27.3%) if they may ignore secure coding guidelines in case the customer of the software would not be harmed. Furthermore, the agreement level (32.7%) and disagreement level (40.0%) show only a slight tendency towards disagreement. These observations are surprising given that software developers should not ignore secure coding guidelines based on their judgment, e.g., on how the end-customer will use software.

In general, we conclude that software developers agree on not ignoring SCG (i.e., following them). However, considering the agreement level on competencies and skills, software developers might lack the skills to judge whether they comply with the secure coding guidelines.

In a similar study conducted by Assal et al. [22], called "Think Secure, " 123 software developers from different industry sectors, mostly located in Canada and the United States, were surveyed. The average age of the software developers was 41.3 years, which is comparable with the age range of the participants from our study. We have looked at the questions

and results by Assal et al. To compare their results to our own, the results presented in table IV were further processed. First, we use the following standard Likert mapping: *strongly disagree* $\rightarrow$ 1, ... *strongly agree* $\rightarrow$ 5. Also, two of the Likert scales are inverted: the construct PC-NT (*have necessary time*) is inverted to match the question in Assal et al. and similarly for the construct SE-C2 (*have knowledge*) as compared to D24 (*have no knowledge*). We also considered a mapping between a compound construct based on the average agreement between CBg2 and CBg7 to the construct *we have security procedures* in the survey by Assal et al.

Table V shows a summary of the comparison of the average agreement between the current work and the survey from Assal el al. Note that the errors in the "think survey," except for the construct (CBg2+CBg7), are smaller than in our survey - this fact is not a surprise, given that the "think survey" includes 140 participants compared with 59 in our pilot study survey. Table V shows that results concerning the existence of security procedures, the knowledge of the importance of security, and software developers' facilitating conditions are comparable between the two surveys. Furthermore, the alignment of the results validates our questionnaire, the results of the data analysis and approach.

C. Threats to validity

A threat to our results' validity steams from the fact that, although a reasonable number of answers were collected (59), data comes from a pilot study. To improve the results' quality, we did not consider the full collection of questions and answers in this study from the pilot survey - only those that also survived the post-review process unchanged. Also, the data collected in this survey comes mainly from companies based in Asia; regional results might vary and may lead to different conclusions. However, we notice that the results that we have obtained in our survey agree with the work from Assal et al. [22], which was performed in the United States and Canada and was also deployed over several different companies.

D. Impact of this work

More effort and different measures are needed to increase code security, particularly for software used in critical infrastructures. Our preliminary findings indicate that software developers have the resources for writing secure code, but not necessarily the knowledge, capabilities, and skills. It needs to be discussed to what extent software developers can judge whether they comply with secure coding guidelines. More research and a more in-depth analysis of data are necessary to resolve the dichotomy between the current status-quo and the number of vulnerabilities vs. software developer self-assessment on the secure coding guideline topic. A better understanding of the topic may contribute to more secure code and more effective and efficient organizational structures for secure coding. The first data analysis of preliminary data and the publication of our questionnaire contribute herein.

V. Conclusions

The work presented in this paper is motivated by industry needs. Secure coding is a fundamental competence that every software developer working in the industry should have. Being competent in secure coding and secure coding guidelines can significantly impact the industry in terms of product quality and security. This impact becomes especially significant for software that is deployed in critical infrastructures. However, software vulnerabilities are still increasing, which raises the question of - why is it so?

The survey developed in this work tries to answer this question by focusing on the human factor - the software developer - and its compliance to secure coding guidelines. In particular, we explore the following three aspects: which factors lead software developers to use or ignore secure coding guidelines, how well established are secure programming practices in the industry, and how well do software developers know secure coding guidelines. Our survey design is based on a mixture of previously well-established theories, the authors' industry experience, and lessons learned from previous serious games on secure software development and secure coding guidelines.

The survey was piloted in three different companies during several CyberSecurity Challenges workshops, which are secure coding workshops to train software developers in the industry on secure software development. The preliminary results, which are partially confirmed by a similar previous work, show that, while software developers express intention to comply with secure coding guidelines, real-world knowledge on the guidelines is lacking. Our results also indicate the need for running secure coding awareness campaigns directed towards software developers in the industry. According to the authors' many years of experience in the industry, software developers tend to overestimate their secure coding capabilities. Together with the results hereby presented, we hypothesize that software developers need to be challenged to grow in terms of knowledge and awareness on secure coding.

We believe that CTF-like serious games on secure coding, which are designed for software developers in the industry, are an adequate method to raise secure coding awareness. Awareness training based on these kinds of games poses secure coding challenges to players, which can be used to measure the self-knowledge on secure coding and raise awareness on secure coding guidelines.

Based on a large scale deployment of the designed survey, further work will extract practical advice for CTF-like awareness campaigns and analyze the relationship between the different variables to gain further insight. One such additional insight might be, e.g., the relationship between the number of years of experience of participants and their knowledge level of secure coding guidelines or willingness to use secure coding guidelines.

Supporting Data

The raw data collected in the CyberSecurity Challenge workshops, which is the basis for this work, is openly available

TABLE V
COMPARISON BETWEEN THIS STUDY AND THE STUDY FROM ASSAL ET AL.

This study		"Think Secure" Study [22]	
Construct	Average	Construct	Average
CBg2 + CBg7 †	3.1 ± 0.6	we have security procedures	3.8 ± 0.9
BkG1	4,0 ± 0,3	security is important	4.3 ± 0.2
PC-NT*	2.5 ± 0.8	D23: no time	2.3 ± 0.3
SE-C2*	2.5 ± 0.7	D24: no knowledge	2.6 ± 0.3

Note†: Construct corresponds to the average of CBg2 and CBg7 from table IV
Note*: Construct has an inverted Likert scale in comparison to table IV

in the Zenodo [27] platform. The raw survey data is provided in Comma Separated Values (CSV) format.

ACKNOWLEDGEMENTS

The authors would like to thank the survey participants for completing the survey, their time, and their comments. The authors would also like to thank the three anonymized companies which have allowed the survey pilot to be carried and the manuscript reviewers. The authors would also like to thank Kristian Beckers and Thomas Diefenbach for their helpful, insightful, and constructive comments and discussions.

This work is co-financed by portuguese national funds through FCT - Fundação para a Ciência e Tecnologia, I.P., under the project FCT UIDB/04466/2020. Furthermore, the third author thanks the Instituto Universitário de Lisboa and ISTAR-IUL, for their support.

REFERENCES

[1] MITRE, "CVE Details," Online, 2019, https://www.cvedetails.com/.
[2] Apextechservices, "Notpetya: World's first $10 billion malware," 10 2017. [Online]. Available: https://tinyurl.com/y6mkok57
[3] Department of Homeland Security, "ICS-CERT: Industrial Control Systems - Computer Emergency Response Team," 09 2020. [Online]. Available: https://us-cert.cisa.gov/ics
[4] MITRE-Corporation, "Common Weaknesses Enumeration," 2019. [Online]. Available: https://cwe.mitre.org/
[5] S. Patel, "2019 Global Developer Report: DevSecOps finds security roadblocks divide teams," July 2020, [Online; posted on July 15, 2019]. [Online]. Available: https://about.gitlab.com/blog/2019/07/15/global-developer-report/
[6] T. Gasiba, K. Beckers, S. Suppan, and F. Rezabek, "On the Requirements for Serious Games geared towards Software Developers in the Industry," in *Conference on Requirements Engineering Conference*, D. E. Damian, A. Perini, and S. Lee, Eds. Jeju, South Korea: IEEE, 09 2019, pp. 286–296. [Online]. Available: https://ieeexplore.ieee.org/xpl/conhome/8910334/proceeding
[7] G. D. Moody, M. Siponen, and S. Pahnila, "Toward a Unified Model of Information Security Policy Compliance," *MIS quarterly*, vol. 42, no. 1, pp. 1–50, 2018.
[8] B. Bulgurcu, H. Cavusoglu, and I. Benbasat, "Information Security Policy Compliance: An Empirical Study of Rationality-Based Beliefs and Information Security Awareness," *MIS quarterly*, vol. 34, no. 3, pp. 523–548, 2010.
[9] J. D'Arcy, T. Herath, and M. K. Shoss, "Understanding Employee Responses to Stressful Information Security Requirements: A Coping Perspective," *Journal of management information systems*, vol. 31, no. 2, pp. 285–318, 2014.
[10] M. Siponen and A. Vance, "Neutralization: New Insights into the Problem of Employee Information Systems Security Policy Violations," *MIS quarterly*, vol. 34, no. 3, pp. 487–502, 2010.
[11] R. Dörner, S. Göbel, W. Effelsberg, and J. Wiemeyer, *Serious Games: Foundations, Concepts and Practice*. Switzerland: Springer International Publishing, 1. Ed, 2016.
[12] B. Schneier, "Software Developers and Security," Online, July 2020, https://www.schneier.com/blog/archives/2019/07/software_develo.html.
[13] J. Xie, H. R. Lipford, and B. Chu, "Why do Programmers Make Security Errors?" *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 161–164, 09 2011.
[14] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What Security Questions Do Developers Ask? A Large-Scale Study of Stack Overflow Posts," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 910–924, 09 2016. [Online]. Available: https://doi.org/10.1007/s11390-016-1672-0
[15] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, "Stack overflow considered harmful? the impact of copy&paste on android application security," in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE. an Jose, CA: IEEE, 2017, pp. 121–136.
[16] Y. Acar, C. Stransky, D. Wermke, C. Weir, M. L. Mazurek, and S. Fahl, "Developers Need Support, Too: A Survey of Security Advice for Software Developers," in *2017 IEEE Cybersecurity Development (SecDev)*. Cambridge, MA, USA: IEEE, 09 2017, pp. 22–26.
[17] T. Gasiba and U. Lechner, "Raising Secure Coding Awareness for Software Developers in the Industry," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. Jeju, South Korea: IEEE, 09 2019, pp. 141–143.
[18] D. Votipka, M. L. Mazurek, H. Hu, and B. Eastes, "Toward a Field Study on the Impact of Hacking Competitions on Secure Development," in *The 4th Workshop on Security Information Workers Baltimore Marriott Waterfront-Baltimore*, Baltimore, MD, USA, 2018, pp. 1–6.
[19] A. Davis, T. Leek, M. Zhivich, K. Gwinnup, and W. Leonard, "The fun and future of CTF," *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, pp. 1–9, 2014. [Online]. Available: https://www.usenix.org/conference/3gse14/summit-program/presentation/davis
[20] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, "What happens when software developers are (un)happy," *Journal of Systems and Software*, vol. 140, pp. 32–47, 2017.
[21] ISO, "ISO 250xx Series," International Organization for Standardization, Geneva, CH, Standard, 2005. [Online]. Available: http://iso25000.com/index.php/en/iso-25000-standards
[22] H. Assal and S. Chiasson, "'Think secure from the beginning' A Survey with Software Developers," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1–13.
[23] N. Haensch and Z. Benenson, "Specifying IT security awareness," in *25th International Workshop on Database and Expert Systems Applications, Munich, Germany*. Munich, Germany: IEEE, Sep 2014, pp. 326–330.
[24] S. Wagner, D. Mendez, M. Felderer, D. Graziotin, and M. Kalinowski, "Challenges in Survey Research," in *Contemporary Empirical Methods in Software Engineering*, G. H. T. Michael Felderer, Ed. ArXiv: Springer, 2020, pp. 1–34.
[25] D. Dillman, *Mail and Internet surveys: The tailored design method–2007 Update with new Internet, visual, and mixed-mode guide*. Indianapolis: John Wiley & Sons, 2011.
[26] R. Likert, "A Technique for the Measurement of Attitudes," *Archives of psychology*, vol. 22, no. 140, pp. 1–55, June 1932.
[27] *Raw Results for the Preliminary Survey on Awareness of Secure Coding Guidelines in the Industry*. Zenodo, Oct. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4075282