Secure Coding Intention via Protection Motivation Theory Based Survey

University of Oulu
Faculty of Information Technology and Electrical Engineering
Degree Programme in Information Processing Sciences
Master's Thesis
Tommi Sallinen
Date 27.4.2020

Abstract

According to studies, programming skills are obtained by a large number of persons but most of them lack the ability to produce secure software. This statement reflects the essence of this thesis and provides a direction to problem solving.

The focus of this study is a research into the possibility of using a questionnaire prepared with the use of a protection motivation theory (PMT) to provide a indication of intention for software developers towards secure programming techniques. This study answers the following research question: Can secure programming intention be aroused with a PMT questionnaire?

The questionnaire consists of three categories: background-, awareness-/knowledge- and PMT questions. Background questions are used to identify the focus group. Awareness and knowledge questions are used to provide secure coding information which is reflected by cognitive thinking via PMT questions. The questionnaire was built as web survey and distributed via professional social network.

The questionnaire uses focused subject group working in micro and small enterprises (<50 employees). The study results are analysed against PMT components to validate focus group selection as a correct choice. Survey findings analysed in qualitative manner (partly in quantitative), indicates that majority of subjects created intention towards studying or using secure coding techniques. The focus group PMT analysis results shows that in each PMT section, at least over half indicated positive response into it.

These results will provide a deeper research direction for how to promote secure coding.

Keywords
secure programming, secure coding, protection motivation theory, pmt, survey, questionnaire

Foreword

The main idea and motivation for this study resulted from my own observations and an increasing number of software vulnerability news. Also, my personal interest in lifelong learning added own characteristics to this study. Making this study was been long marathon, with up- and downhills. Complementing this thesis has been a time-consuming but at the same very rewarding process.

I would like to thank my supervisor Mari Karjalainen for starting this project with me and Ari Vesanen for guiding me. Special thanks also to my lovely wife and kids for supporting my “never-ending” project. Lastly, thanks to Joanna Seppänen for proofreading this thesis.

I truly hope that my study will highlight the importance of a programmer in the field of secure coding.

Tommi Sallinen

Oulu 27.4.2020

Abbreviations

PMT = Protection Motivation Theory

SDL = Secure Development Life Cycle

SDLC = Software Development Life Cycle

SMB = Small and Medium size Business

UML = Unified Modelling Language

Contents

1. Introduction

Long subcontracting chains and possibilities provided by the internet increase the value of software development and the importance of each company. On the other hand, global internet use and complicated information systems drastically increase an attack surface which is tackled with different information security tools. Regardless all development lifecycle models, standards, company rules, principles, practices, tools etc., the last and most important person in the organization is the software developer who writes the code (Mahadevan, Simon, & Meservy, 2011). Taking this into consideration we could think that the most important defence against a malicious hacker is a secure minded programmer. Small and medium business cannot afford to put money into planning and implementing information security throughout organisation. Expertise and knowledge of each employee is more important to the company than company rules or practices. Tackling information security risks purely via technology is destined to fail as human factor is the weakest link in the security (Mitnick & Simon, 2002).

Nowadays secure programming teaching is being tested and researched. In most cases secure programming learning is done via face-to-face teaching. This study is researching the possibility to increase motivation and attitude towards secure programming, so that a person would start to learn the subject and also would be more secure minded. Can we empower single individual to be as a critical part of secure programming? (Bishop & Frincke, 2005.)

Best practices, standards and company rules cannot hinder the fact that software developers have to be aware of secure issues related to programming (Futcher & Von Solms, 2008). Most critical problem is that software developers do not have security view of their product which can be also seen as awareness (Jones & Rastogi, 2004). Software developers should build easier-to-defend code (McGraw, 2004). Tackling secure coding challenge we should have advance in three areas: education, standards and metrics (Graff & Van Wyk, 2003).

Crossler et al. (2013) proposes that one of the future research directions would be improvement of information security compliance by using the Protection Motivation Theory (PMT) which is driven by fear appeals. This study uses PMT to explain and to reflect data acquisition results. The research method that is utilised is the qualitative questionnaire. (Crossler, et al., 2013.)

Using PMT make it possible to drive the subject towards a desired action or behaviour via cognitive thinking process (Rippetoe & Rogers, 1987; Rogers, 1983). Where fear and vulnerability are motivating factors in it. Combining this with authors own interest in secure coding resulted in forming the following research question.

RQ: Can secure programming intention be aroused with a PMT questionnaire?

This is a qualitative research that aims at studying whether a PMT based questionnaire can create intention towards secure coding. By using qualitative data analysis, it is possible to identify if any intention is aroused. The target group of the survey is micro and small business personnel who have extensive knowledge in the field of

programming. The focus group of the questionnaire includes programmers who work in small enterprises.

Author's own personal interests lie in promoting and studying secure and quality programming. The information security field has been extremely interesting to follow, study. Author's interest in educating regular programmers to be more secure minded in their work has led to completing this thesis.

Previous research focused on the way to enforce secure programming via development lifecycles, tools, training, testing, programming language and teaching techniques/curriculum which are either above (organizational rules, policies, training, top-to-down management etc.) or below (IDE tools, programming languages etc.) a programmer. All of these are discussed in this thesis but in relation to the challenge of applying secure programming mindset.

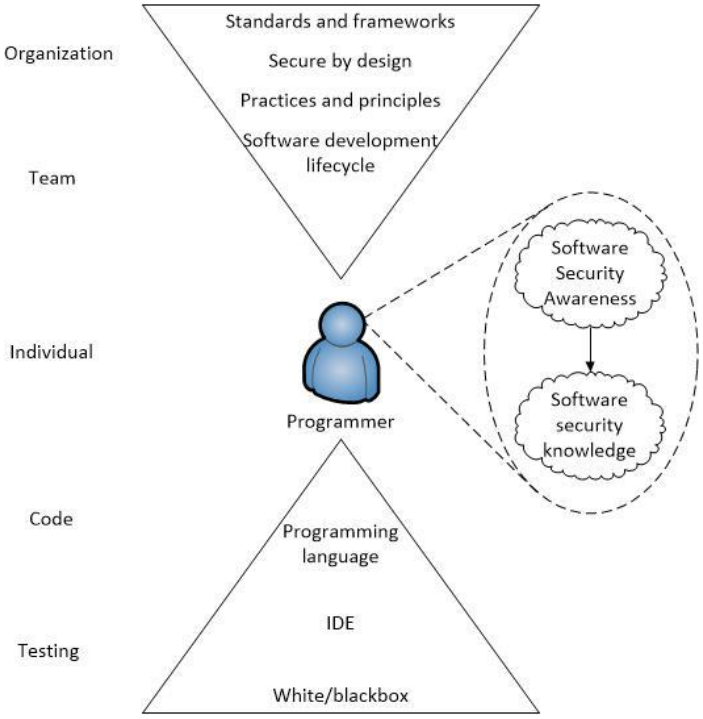

Figure 1. Different impact levels of secure programming.

Figure 1 represents the impact of programmer's practical work on different levels of software development. This figure highlights the importance of programmer's awareness and knowledge, as he or she is single most important element of the whole software development process. Regardless of a development stage (design, programming, testing/verification etc.), developer is always involved. (Figure 1)

Figure 1 clearly shows that the number of interfaces that affect a programmer is high. It can be seen that study fields below a programmer can produce empirically measured data. The study of different levels of organisation research presented above is more focused on handling secure programming as a homogeneous mass. Individual studying requires a combination of three components: practical programming, software development methods and behavioural theories.

A software security awareness research field has similar characteristics to information system security awareness. However, the approach is closer to practical work tasks and programming challenges. This study leans towards the software security awareness research field but it is different in nature as it aims at awaking developer's intention towards secure coding via questionnaire that includes PMT nuances. Contrary to typical awareness research, this research explores the potential of the subject group for secure programming awareness. It worth of noticing that the software security awareness research field is close to this study. The difference is that this study aims to create intention towards learning secure coding. It is possible to use results of this study to promote secure coding in its all perspectives to developers. According to author, secure coding awareness differs from secure coding knowledge because awareness does not provide a developer with solid knowledge that could be reflected to in each programming task in hand.

The second chapter of this thesis provides background information on the current situation. The concept of information security is discussed with the focus on secure software development in chapters 2.1, 2.2 and 2.3. Chapter 2.4 covers the topic of secure programming and its different perspectives. Moreover, it provides the reader with a broader understanding of the information security field. The need for this kind of research is discussed in Chapter 2.5. and the theory used in this research is described in Chapter 3. Chapter 4 presents the research method, theories and leads reader to core of this study. In addition to these, data gathering method and tactics are discussed in Chapter 5 together with a thorough connection between these theories and implemented surveys. Data analysis of collected results shown in Chapter 6 is followed by a presentation of these results in Chapter 7. Final conclusions are drawn, and research questions are answered in Chapter 8.

2. Research on Secure Coding

This chapter provides a cross-section of methods and tools used inside of an organization in order to make a more secure code. It should be noticed that the implementation of many organizational tools or methods is based on top-to-down management as secure programming methods and tools are in use of individual developer.

"Securing coding is the practice of developing computer software in a way that guards against the accidental introduction of security vulnerabilities" (Wikipedia, 2017).

Literature presents secure programming from multiple different perspectives. These perspectives vary from standards and development methods to tools and training. As in the case of information security, also secure coding awareness refers to a wide range of perspectives in the secure coding. Main perspectives of secure coding are represented in the following subchapters.

Directly relevant studies are represented in the Prior research -chapter.

2.1 Organisational tools, frameworks and standards

Previous research focused on secure programming for example creating guidelines, which drives software development on the organisational level. Others learning perspective and secure programming are viewed via curriculum by acknowledging the difference between different software security categories. The meaning of software security robustness is reflected in capability and process level maturity (CMM level). (Futcher & Von Solms, 2008; Yasinsac & McDonald, 2006)

There are many different secure programming standards available, which provide guidance. The ones that are mostly used are: ISO/IEC (ISO/IEC 27002, ISO/IEC TR 13335) standards, which define the framework for software life cycle processes. For example, ISO/IEC 27002 states that limited security is possible to achieve by technical means. ISO/IEC 27002 standard focuses more on achieving proper security level by using management controls and procedures. (Futcher & Von Solms, 2008.)

An organisation could use Software Security Assessment Instrument (SSAI) to improve software security. SSAI consists of Software Security Checklist (SSC), Vulnerability matrix, Flexible Modeling Framework (FMF), Property- Based Tester (PBT) and a collection of Security Assessment Tools (SATs). (Gilliam, Wolfe, Sherif, & Bishop, 2003; Gilliam, Kelly, Powell, & Bishop, 2001.)

McGraw (2006) proposes seven touchpoints to be used inside of an organization to increase software security:

1. Code Review (tools)
2. Architectural Risk Analysis
3. Penetrating Testing
4. Risk-Based Security Testing
5. Abuse Cases
6. Security Requirements
7. Security Operations

An external analysis is not defined as a touch point but its importance is emphasized (McGraw, 2006).

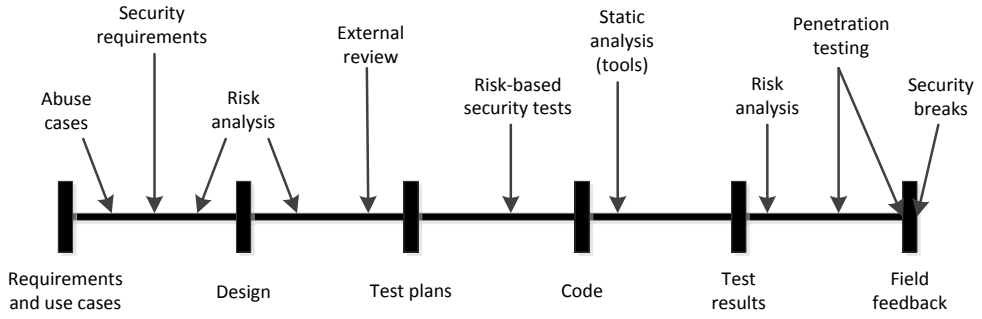

Figure 2. Best software security practices applied to different software phases (McGraw, 2004; McGraw, 2006; Van Wyk & McGraw, 2005)

The touch points or best practices (Figure 2) can be applied in different software development phases. Secure software development is used parallel with existing SDLC. In this way, secure software development does not need to have its own development life cycle model. (McGraw, 2004.)

2.2 Designing secure software

One approach is to ensure security of software via design. This is called secure by design and it can be achieved by securing software via SecureUML which is a variation of a regular UML design for modelling access control policies (with Role Based Access Control) utilised in model-driven software development (Lodderstedt, Basin, & Doser, 2002). Other modelling extension is UMLsec which can be used to express security relevant information (demands mandatory requirements to be filled) (Jürjens, 2002).

According to Viega and McGraw (2002) “It is always better to design security from scratch than to try to add security to an existing design.” (p. 14). This statement reflects the core idea of this study on how to improve secure programming among software developers in such a way that software development includes security perspective from the very beginning and software developers are secure minded in their work. Best practices proposal for secure programming relies on defining security requirement via abuse cases by recognizing overt functional security and emerging characteristics. This lies solid foundation via requirements for secure coding. (Viega & McGraw, 2002; McGraw, 2004.)

2.3 Securing software development lifecycle

Securing software development lifecycle has a major role in most frameworks and standards. The reason for this is that it provides a managerial tool to produce secure software. Securing development lifecycle could be a combination of existing methodology with secure coding nuances or it can be own software development methodology. Ensuring security of software can be also done parallel to existing software development cycle so that discovered security risks and issues are fixed in the software development phase where they were founded. (Byers & Shahmehri, 2007.)

Secure coding can be implemented into SDLC in many different ways from standardized frameworks, by using ready SDLC which emphasizes secure coding, building customized SDLC with secure coding in mind or by picking up best practices suitable for SDLC in hand.

Frameworks

Frameworks gives general guidelines for securing development lifecycle. NIST SP 800-64 document describes framework to be used in different parts of SDLC. Document helps to select and acquire right security controls but document cannot be used directly to implement SDLC. (Kissel, et al., 2008.)

One example of framework is: Comprehensive, Lightweight Application Security Process (CLASP), CLASP is part of Open Web Application Security Project (OWASP). Security companies related to OWASP consortium contributed and reviewed CLASP. (Gregoire, Buyens, Win, Scandariato, & Joosen, 2007.)

Secure development lifecycle

Microsoft Secure Development Lifecycle (MS-SDL) if founded and used widely by Microsoft. Purpose of MS-SDL is to have secure software development for all cloud-based software. MS-SDL goes hand-in-hand with business threat analysis as part of MS-SDL is to estimate business impact of certain threat. Microsoft -company shares publicly MS-SDL to everybody who is interested of it. (Lipner, 2010.)

Comparison of MS-SDL and CLASP indicates that CLASP is lightweight and it can be more customized for specific usage comparing to MS-SDL which do not provide same flexibility as CLASP. (Gregoire, Buyens, Win, Scandariato, & Joosen, 2007; Futcher & Von Solms, 2008.)

2.4 Secure coding

By using secure coding skills, a software developer minimises the amount of vulnerabilities in the source code. Arbaugh, Fithen and McHugh (2000) have defined the life cycle of vulnerability, which consists of birth, discovery, disclosure, correction, publicity, scripting and death. It should be noted that according to this study, even after discovering and patching, abuse of vulnerability increased until most people had upgraded to patched version. (Arbaugh et al., 2000.)

Secure programming can be divided into two different subfields: software security and application security. As software security focuses on designing and building secure software, application security is done after development in post facto way. According to some studies, an application security solution is not the right way to implement secure

programming because developers should have a secure programming mindset or awareness right from the beginning (Zenah & Aziz, 2011; Tøndel, Jaatun, & Meland, 2008). Implementing security after deployment could lead into conflicts with system requirements which can be seen also as a vulnerabilities (Anderson, 2008). It is important to inspect software in its real use environment so that all vulnerability aspects can be seen. (Hoglund & McGraw, 2004.)

Kumar, Pandey and Ahson (2007) lists the following secure coding practices: coding standards, code reviews, unit testing for security and defect management. On the other hand, study proposes use of tools to automate parts of code review which can be implemented in to practice regardless of the organization size. (Kumar, Pandey, & Ahson, 2007.)

2.4.1 Taxonomy

Bugs are existing software problems which can stay hidden as they are not executed but their number can be reduced by using code scanners. Bugs are simple implementation problems that can be tackled on an implementation-level. Flaws have deep roots in software and they spawn from an implementation level to design. Because of their subtle nature, flaws exist in the code without being exploited. Vulnerability is either individual or it is a combination of bugs and flaws. Attackers exploits vulnerabilities to achieve their goals. Complexity of flaws makes bugs more appealing targets of exploits. (Hoglund & McGraw, 2004.)

Exploits of vulnerabilities fall into following categories (Gilliam, Wolfe, Sherif, & Bishop, 2003):

1. Environment variables
2. Buffer Overflows
3. Data as Instructions or Script Injections
4. Numeric Overflows
5. Race Conditions
6. Network Exposures
7. Information Exposure
8. Operational Misuse
9. Default Settings
10. Programmer Backdoors

The list above reflects vulnerabilities of the implementation level and it should be well understood by a developer whom does not have secure coding knowledge. Most complex and discreet vulnerability is design-level vulnerability. Creating design vulnerability automation is difficult as it requires extremely good expertise. Vulnerability imposes major security risk in the code. (Hoglund & McGraw, 2004.)

Reports of vulnerabilities are steadily increasing mainly because of three factors: connectivity, extensibility and complexity. By connectivity, we mean a growing number of internet connectivity, which increases attack surface and enables remote attacks. As an attacker does not need to be in the proximity of his or her target. Extensibility of i.e., applications, operating systems and web browsers, creates a major challenge to prevent vulnerabilities from existing. This in turn, puts more pressure on a design phase of software. Most software created to day has high complexity which leads to an increasing number of vulnerabilities. (McGraw, 2002.)

2.4.2 Practices and principles

NIST SP 800-14 documentation provides a list of important principles and practices which should considered in secure programming. This documentation also connects practices and principles with each other. (Swanson & Guttman, 1996.)

Besides of the NIST SP 800-14 documentation, Viega and McGraw (2002) proposes 10 principles based on their experience:

1. Secure the weakest link
2. Practice defence in depth
3. Fail securely
4. Follow the principle of least privilege
5. Compartmentalize
6. Keep it simple
7. Promote privacy
8. Remember that hiding secrets is hard
9. Be reluctant to trust
10. Use your community resources

Most of these listed principles are self-explanatory, but as we start to discuss each of them, we can see paradox between the "practice defence in depth" and "keep it simple" –principles. As defence-in-depth embraces the importance of building redundancy, "keep it simple" principle embraces simplicity and understandability of your system. By using software risk management, the principles mentioned above can be applied successfully and efficiently. (Viega & McGraw, 2002.)

Graff and Van Wyk (2003) proposes a list of good practices to be used during implementation; inform yourself (self-education/learning), handle data with caution (sanitize inputs), reuse good code whenever practicable (minimise re-doing), insist on sound review processes (peer review, independent validation and verification, security tools), make generous use of checklists and be kind to maintainers (use standards, remove obsolete code, test all code changes). Inform yourself –practice underlines the importance of self-efficacy via self-learning of coding and especially secure coding. (Graff & Van Wyk, 2003.)

From the point of view of secure programming, it is possible to strategically select only safe programming languages as some programming languages are technically unsafe (C and C++). Safe languages like Java do not cause the problems which C/C++ causes. For example, C/C++ has flaws which enable simple attacks like buffer overflow. Secure coding can be achieved also by developing own typed language to enhance secure programming. (Viega & McGraw, 2002; Swamy, et al., 2011.)

Secure programming can be enforced via selecting integrated development environment (IDE) which has secure programming promoting features (Zhu, Lipford, & Chu, 2013; Microsoft, 2015). Different approach is to create teaching and learning tool to promote secure programming (Zenah & Aziz, 2011).

2.4.3 Awareness

Different studies research secure coding awareness and they bring forward a wide range of perspectives that vary from promoting policies to awareness tools.

Developers should have a secure programming mindset or awareness from the beginning of project (Zenah & Aziz, 2011; Tøndel, Jaatun, & Meland, 2008). Implementing security after deployment could lead to conflicts with system requirements which can be seen also as a vulnerability (Anderson, 2008). Secure coding awareness is fundamental for implementing secure coding skills.

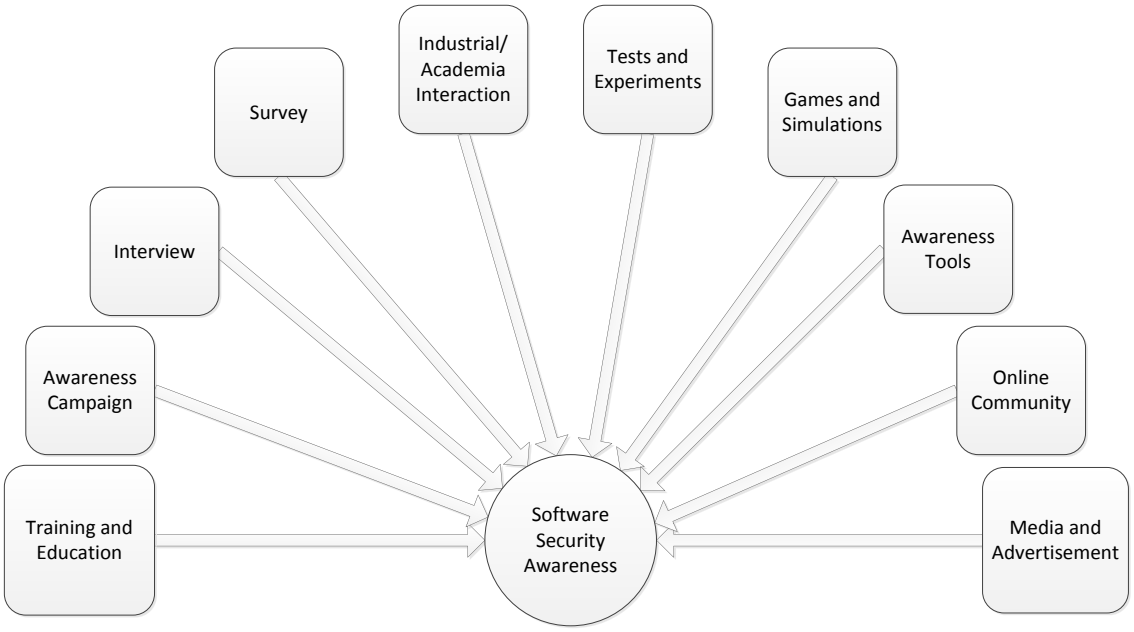

Figure 3. Techniques of Software Security Awareness (Banerjee & Pandey, 2010).

Figure 3 represents Banerjee and Pandley (2010), the pinpointed areas which would affect creation of software security awareness. The study revises existing literature related to security awareness and it concludes that the attack from inside performed, for example, by an employee is the biggest threat to the system. (Banerjee & Pandey, 2010.)

One approach states that awareness and attitude of a software developer can be changed via IDE tools (Whitney, Lipford, Chu, & Zhu, 2015). The other approach proves the point that security awareness can be changed through educating a development, security and operational team (Steven & Peterson, 2006).

An awareness increasing tool, Palantír, used in Configuration Management (CM) systems, decreases unresolved conflicts in the code (Sarma, Hoek, & Redmiles, 2007). Other similar tool, YooHoo awareness system, has the same goal but it is developer specific (Holmes & Walker, 2008). The same awareness tools could be used to promote secure programming principles and practices.

2.5 Prior research on secure coding via Protection Motivation Theory

This study differs from typical secure coding awareness because it examines closer to programmer's work tasks as the ability to implement secure coding as a result of cognitive reasoning. Therefore, finding prior research was difficult. This problem was tried to be tackled with a research strategy.

Relevant research is represented in its own subchapter.

2.5.1 Search strategy

Search query is built in three phases. Firstly, keywords are identified, secondly query parts are constructed and lastly, parts are combined into each database command syntax.

Search keywords

A list of keywords was collected into Table 9 (Appendix B. Search Queries) to support building search queries. These were collected from book literature and from references collected in previous chapters. These keywords were categorised based on what they represented.

Main part of the search query is a "secure coding" concept. This concept is a combination of secure –an adjective and programming -a verb with recognition of different variations and words used in literature. The subject of this research is identified as a programmer with its different variations and mostly used words that are used most. From qualitative data collection methods, a survey/questionnaire and an interview represent a data collection category. As PMT is selected theory, it represents the theory category in the search word table.

Parts of queries

Parts of search query are represented in Table 10 (Appendix B. Search Queries). The search clauses were created by combining keywords in Table 9 (Appendix B. Search Queries) with their own category.

Each search clause was prioritized based on the importance of target information. Prioritization was used to indicate the importance of each query part in the whole search query.

Database queries

Search for prior research was accomplished by using the following databases: IEEE, Scopus, Web Science, ProQuest. Each of them uses different search command syntax.

Individual database search queries are represented in search result Tables 11-13 (Appendix B. Search Queries). Each of them is a combination of query parts based on their prioritization. IEE electric library results are extremely low as PMT is used only in few studies.

2.5.2 Relevant prior research

Finding relevant prior research by using Chapter 2.5.1 search strategy was not successful. By performing different combinations of search query parts in Google Scholar, the research using PMT and questionnaire was found in the study of Woon Tan and Low (2005).

Woon et al. (2005) used in their research protection motivation to promote information security in home wireless network security features. This study used a survey structured with hypothesis for each section of PMT that were then used to create questions for each PMT section. Three different sections are: demographic, main research questions and knowledge quiz sections. The knowledge quiz was used to measure respondent's level of knowledge of network security. The results from this quiz were used to validate measured results in relation to self-efficacy. The knowledge domain quiz measures respondent's awareness in the given field. (Woon, Tan, & Low, 2005.)

Woon et al. (2005) study context was different from this study but the idea of using a questionnaire as a way of delivering information and possibly creating intention towards desired goal was same.

There was not any research where secure coding was promoted with PMT. All other research related to secure coding was always missing the most important element, software developer. It is clear that that a great amount of research was performed from tools to code analysis and from standards to development life cycle. However, the main source of software development, the software developer, has not been yet studied equally well.

3. Protection Motivation Theory

Selecting PMT was based on previous usage of theory and especially fear arousal aspect of it. PMT is used to predict users' intentions to protect themselves after communicator recommendations (Floyd, Prentice-Dunn, & Rogers, 2000). PMT was used in this study to create questionnaire so that different sections of PMT can be recognized in the area of secure coding.

The PMT theory is widely used in the computer science field to measure effectiveness of intervention. The Protection Motivation Theory (PMT), created by Ronald Rogers inspects subjects individually. Later on, Rogers refined and connected fear arousal to PMT. (Rogers, 1983.)

During the years PMT has evolved and it has been used only partially, not with all the PMT components. This study uses the PMT model including fear arousal which is defined by Boss, Galletta, Lowry, Moody and Polak (2015) as full nomology of PMT. (Rogers, 1983; Floyd, Prentice-Dunn, & Rogers, 2000; Boss, Galletta, Lowry, Moody, & Polak, 2015.)

In the study, software developers are perceived as individuals. The Protection motivation theory (PMT) views each person as an individual and it excludes environmental factors, such as organization, co-workers etc. (Rogers, 1983.)

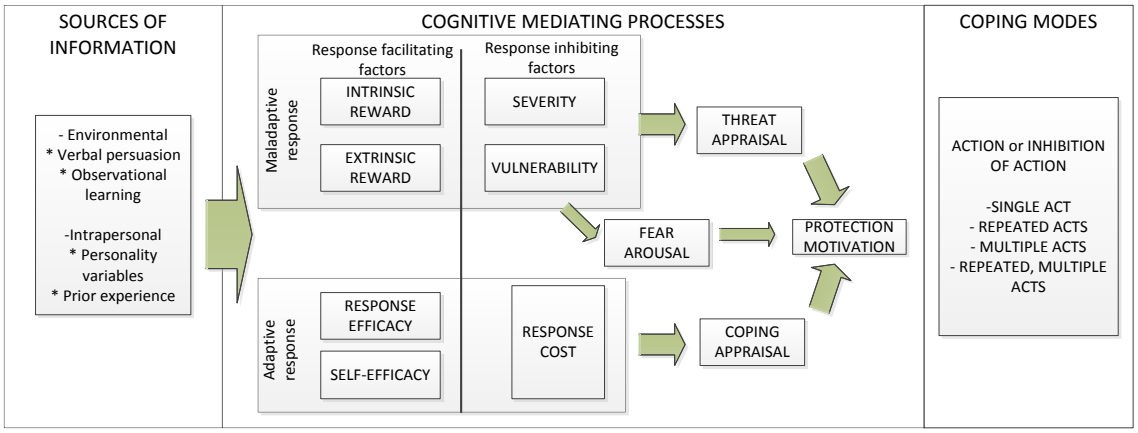

Figure 4. Protection Motivation Theory (Rogers, 1983; Floyd, Prentice-Dunn, & Rogers, 2000)

The schema of PMT is shown in Figure 4. For a survey to be successful, it needs to have focused questions in each response facilitating factors of PMT schema.

The core of a cognitive mediating process core is made from 2x2 table (Figure 4) that contains both factors of increasing and decreasing probability to respond to a given recommendation in coping and threat appraisal. The emphasis of PMT is on cognitive processes rather than fear (Rogers, 1983)

In PMT (Figure 4), the source of information begins cognitive mediating process towards threat and coping appraisal. Both of them generate motivation to action (or inhibition of action). The cognitive mediating process produces threat (maladaptive) and

coping (adaptive) appraisal. As a result, both lead to protection motivation or intention to act. (Floyd, Prentice-Dunn, & Rogers, 2000.)

In order to obtain action, adaptive and maladaptive responses facilitating factors must overweight their decreasing counter parts. Otherwise no motivation is aroused. Threat or fear appealing source of information initiates cognitive mediating process. (Rogers, 1983). Fear arousal generates vulnerability in subject's perception, which leads to protection motivation. (Floyd, Prentice-Dunn, & Rogers, 2000.)

The main principle of PMT is that one tries to protect himself from danger based on four beliefs, which Rogers (1983) lists as the following: "(1) the threat is severe, (2) one is personally vulnerable to the threat, (3) one has the ability to perform the coping response, and (4) the coping response is effective in averting the threat." (pp. 170).

Both, adaptive and maladaptive responses feature response facilitating and inhibiting factors. Facilitating factor components stimulate intention towards protection motivation easier as they provide positive motivation for the subject, regardless of path response. On the other hand, inhibiting factors are components which prevent subject's intention towards desired action(s). (Figure 4)

Adaptive response coping, is based on subjects' high response efficacy (i.e., perception of own responsibilities) and self-efficacy (i.e., sensitivity to learn new things), which are negatively affected by response cost (i.e., additional work required by intended action). Adaptive response could be understood as action which is done by adapting new information via learning process into repairing action. (Floyd, Prentice-Dunn, & Rogers, 2000.)

Contrary to the adaptive response, maladaptive response coping is engaged by intrinsic (i.e., personal satisfaction of job well done) and extrinsic (i.e., thanks given by co-workers) rewards, which are supported by high severity (i.e., own perception of threat) and vulnerability (i.e., how threat is involved in a personal level). Fear arousal coping comes from personal vulnerability towards recognized threat. (Floyd, Prentice-Dunn, & Rogers, 2000.)

PMT has been part of fundamentals of the fear appeals theory. Fear appeals have three different components: magnitude of noxiousness, probability of occurrence and efficacy of recommended response. If one of the previously mentioned components equals zero, no motivation is aroused. (Rogers, 1975.)

Rogers (1983) states that "We learned that fear arousal (which includes a physiological component) can affect attitude change only by first altering the cognitive appraisal of the severity of the threatening event." (pp. 173). Reflecting on Rogers' statement, we should focus on representing threats or noxious events from facts to subjects which then connects facts and threats in their own mind. (Rogers, 1983.)

Rippetoe and Rogers (1987) found in their study that regardless subject's measured coping, threatening communication was pushing the subject more towards adaptive or maladaptive coping appraisal. Based on previous findings, this study focuses on using fear appraisal to motivate engaged subjects to desired intention. PMT is used to create different questions that aim to promote subject's fear arousal in form of intention in order to find out more about the secure coding. The questions should be created in a way that they touch subject's particular PMT component. (Rippetoe & Rogers, 1987.)

4. Research Method

This study uses qualitative research as a research method. This research is based on questionnaire that includes the questions created with the use of using PMT.

Data collection is completed by a web questionnaire, but this study is uses questions in a qualitative manner to drive subject's intention towards secure coding.

Qualitative research can be seen as flexible, subjective and grounding, just to name a few. The purpose of qualitative research is to study phenomenon in their natural settings and it approaches the world in a naturistic and interpretive way. Research method intends to understand and describe social phenomena from inside to outside. (Silverman, 2005; Flick, 2008; Flick, 2018).

A questionnaire in qualitative research is done for selected population (Marshall & Rossman, 2006). Smaller population increases resolution of details in cost of scope (Silverman, 2005). Thus, this study has a narrow and focused group of subjects to improve gathered data quality and details.

Table 1. Qualitative research perspectives (Flick, 2018).

	Approaches to subjective viewpoints	**Description of the making of social situations**	**Hermeneutic analysis of underlying structures**
Theoretical positions	Symbolic Interactionism Phenomenology	Ethnomethodology Constructionism	Psychoanalysis Genetic structuralism
Methods of data collection	Semi-structured interviews Narrative interviews	Focus groups Ethnography Participant observation Recording interactions Collecting documents	Recording Interactions Photography Film
Methods of interpretation	Theoretical coding Content analysis Narrative analysis Hermeneutic methods	Conversation analysis Discourse analysis Analysis of documents	Objective hermeneutics Deep hermeneutics

Research perspectives are summarized in Table 1. This study's theoretical position is the phenomenology study of questionnaire respondent which is called from this on the subject. An interpretation method is hermeneutic as subject's intention towards secure coding is aroused through cognitive understanding of information given by means of the questionnaire.

5. Data Collection

Data collection is accomplished by a web survey which focuses on discovering background information and subject's current knowledge of secure coding. Also, the questionnaire focuses on discovering response inhibiting factors in both adaptive and maladaptive responses. It should also discover preferred media of subject's source of information. Most importantly, the questionnaire may also give a hint of aroused intention towards secure coding.

Preliminary questions are used to narrow the focus group in such a way that subjects do not need to self-reflect on themselves. Concentrating on work and task relating questions provides better answers. Background questions have a major role in selecting correct focus group for the research. This is also major risk of this research if minimum number of answerers is not achieved.

The survey (Appendix C. Online Survey) was constructed with the use of the Webropol –tool (Webropol, 2019) and distributed as a link to a professional networking platform with narrow subject group.

Focus group

In this research it is important to focus only certain focus group. A desired target subject should have the following characteristics: responsible towards own work, focused on programming work, self-learning and curiosity to discover new. These elements are included in the questionnaire to filter the subjects.

The focus group of this study is defined by the author as programmers working in small (<50 persons) enterprise (European Commission, 2014) where software development done by only a few persons. However, employees in a small enterprise have wide knowledge of domain. The author assumes that small enterprise does not have the structure typical for a large organization. Hence, it does not have organizational policies, rules, software development lifecycle models etc. Focus on subjects in small companies helps to direct a questionnaire to implementation-level problems. Because small enterprises have limited resources to focus on fixing vulnerabilities resulting from bugs.

The author understands that PMT is more effective in the case of subjects who believe that they have wider responsibility of their own work.

Questionnaire structure

The focus on the questions needs to be directed in such a way that driving factors to self-learning, in both maladaptive and adaptive response cases are achieved and both threat and coping appraisal paths can be used in future research.

The questionnaire consists of three sections: background information, awareness and knowledge, PMT based questions. Each section forms a group of questions that are used to give information on a process of subject's cognitive thinking.

Questions in each of the section are listed in the full appendix (Appendix A. Questions) and they are identified with a question identification number.

5.1 Background questions

Subject's suitability for the focus group is filtered with the use of a background questionnaire with the following categories:

Table 2. Background question categories.

Category	Target	Reason	Question
Maturity	Responsibility	Life experience should affect taking responsibility for own actions	1, 2, 3, 4
Education	Knowledge	Basic knowledge on the subject	2, 6
Company size	Work environment	Different -sized companies requires different ways of doing practical programming (free vs. strict)	3
Work experience	Knowledge	Longer experience in the programming field could give better perspective of software vulnerabilities	4
Work role	Practical or Architectural	What is subject's input to software development	5
Practical coding	Involvement level	How often does the subject use secure coding?	6

List of background questions:

Q1: What is your age?
Q2: What is your education level?
Q3: How many employees are there in your company (or in a typical client company)?
Q4: How long (years) have you been working in the field of programming?
Q5: What is your work role(s)? (rank roles if you have many)
Q6: How often do you do practical programming in your work?

The purpose of each background question is to identify the subjects according to previous specifications. Most of the time, workers have multiple work roles, thus roles should be ranked. A rank order allows the subjects to establish primary and secondary work roles, e.g., even though a manager does mainly management work, he or she could also do some programming.

Involvement level supports work role questions answer. For example, a subject feels that his primary work role is being a programmer and secondary a designer. However, his level of involvement is low because most of his time is spent on designing.

5.2 Awareness and knowledge questions

Awareness and knowledge questions are based on secure coding implementations in different forms. These questions are not formed with a specific programming language in mind. They contain information useful for all software developers.

Table 3. Awareness and knowledge questions.

Category	Target	Reason	Question
Secure coding principles and practices	Awareness	Is the topic familiar to the subject	7, 8, 18
Organizational policies	General rules	Is secure programming enforced through organizational policies and rules?	9, 10, 11, 12
Practical work	Practical knowledge	Is subject aware of practical secure coding methods?	12, 13
Organizational standards	Practical work framework	Is software development driven by certain standards?	14, 15
Software development lifecycle	Practical programming	How is software development organized?	16, 17, 19

List of awareness and knowledge questions:

Q7: Do you know what secure programming / coding is?
Q8: Have you used secure programming principles and practices in software development?
Q9: Does your company use information security policies?
Q10: Do you use vulnerability lists in your work?
Q11: Do you use code analysis and/or secure programming tools in your work?
Q12: Do you use secure development lifecycle (SDL) in your work?
Q13: What kind of software development lifecycle method is used in your projects?
Q14: Do you use secure software development standards in your work? (i.e., ISO/IEC 27002)
Q15: Do you use "secure by design" –design method in your work?
Q16: Do you use integrated development environment (IDE) tools with software security promoting features?
Q17: Do you use code reviews in your work?
Q18: Do you know what an "attack pattern" is?
Q19: How do you select a programming language for a project?

These questions measures awareness and pre-existing knowledge, but also should indicate lack of subject's knowledge in secure coding. Subject's knowledge of secure programming is measured from different perspectives (organization, practical methods, standards and development methods). Hence, the questions work as a PMT's source of information and prepares the subject for PMT based questions. Questions are based on general knowledge of secure coding (Chapter 2).

To be able to do coping response and being personally vulnerable to threat are linear functions of PMT (Rogers, 1983). Hence, intention towards studying secure coding should be stimulated among the subjects who are motivated but not aware of secure coding. Collecting motivation source with the use of open questions will be used in future research.

As previously defined, the focus of this study is on finding out whether it is possible to create for a software developer an intention towards secure coding. It is important that the questions allow to define motivation of the subject (importance of motivation was discussed in Chapter 3). As discussed in the Chapter 2.4, learning about secure programming should come first on a general level. Next it can be intensified once knowledge domain has been deepened.

It is also crucial to identify the best influence channel for the future use in research. This can be done by using open questions.

5.3 PMT -based questions

Software security awareness and motivation are enforced through PMT -based questions. Questions categories are based on PMT sections and they are directed to specially to address software developers who work in SME's.

Table 4. PMT questions categories.

PMT section	Object of question	Subject of question	Question
Source of information	Subject	Tools, media	24
Instrinct reward^{+M}	Subject		22
Extrinsic reward^{+M}	Subject	Positive feedback	26, 27, 28
Severity^{-M}	Subject	Software	25
Vulnerability^{-M}	Subject	Responsibility	31
Response efficacy^{+A}	Self-education	Learning, Homing	23, 29
Self-efficacy^{+A}	Subject	Learning	20, 21
Response cost^{-A}	Employer	Time, Money	30
Protection motivation	Subject	Intention	32

^{A} Adaptive response
^{M} Maladaptive response
+ Response facilitating factor
$^{-}$ Response inhibiting factor

List of PMT based questions:

Q20: Do you improve your knowledge of programming by i.e., reading books or taking courses in your working hours?
Q21: Do you improve your knowledge of programming by i.e., reading books or taking courses in your spare time?
Q22: Do you find satisfying to be able to implement something you have learnt?
Q23: What motivates you to learn about new programming techniques or languages?
Q24: What media/tools or sources do you prefer to use while learning new things related to programming?
Q25: How critical do you see your software's impact?
Q26: How often do you receive positive feedback (for example spoken, online chat, email etc.) from your colleague(s) on your work performance
Q27: How often do you receive positive feedback (for example spoken, online chat, email etc.) from your manager(s) on your work performance
Q28: How important to you is the feedback you receive?
Q29: Do you have hobby projects related to programming? (open source etc.)
Q30: Would you be ready to produce a secure code even if it required more effort?
Q31: Are you concerned that the software you produce may lack some critical security point of view?
Q32: I am likely to use or study secure programming techniques in the future.

Questions should provide answers to each section of PMT schema. Questions types are categorized according to Table 4. Each PMT question contains knowledge of secure coding, which should affect subject's motivation to answer the final question about secure coding intention.

Self-efficacy is important quality of the focus group. Subjects with high self-efficacy, express higher willingness to comply recommendations (Rogers, 1983). Because of this, subjects that have self-efficacy can be analysed. Self-efficacy also reflects subjects' motivation to learn (Zimmerman, 2000). In this study self-efficacy is measured in terms of free time learning and programming. Self-efficacy is not referred to directly as subjects' perspective can be different from the truth. This shows how much each individual is motivated to improve themselves voluntarily. Questions in the severity and vulnerability sections are based on own understanding of produced software usage and its importance. It should be possible to identify correctly uneasiness of the subjects by realising and assessing potential risks and damage of developed software. Severity questions are in one matrix questions; thus, subject can easily compare severity levels and their importance. In extrinsic reward questions, colleague and manager feedbacks are asked in the same, matrix question. In this way, subject can reflect own importance of acceptance, between colleagues and managers.

Fear arousal is done by combining previous information from the questionnaire with subject's present situation. Intention is measured with last question that indicates subject's intention to study or use secure programming techniques.

5.4 Constructing questions

Most of the questionnaire's questions are closed questions, which makes answering them easier, faster and more substantive. The primary goal is to make the subject read the questions carefully and think well about their content. In this way closed questions works as information distribution objects.

With certain questions, additional information is needed. For example, with ranking and matrix question types (Appendix C. Online Survey) as more specific guidance is required.

Open questions are used when answers differ and depend on each individual. Constructing answer options for these questions would narrow results drastically.

Matrix questions are used where answer could be neutral. For example, between two opposites (Appendix C. Online Survey). These questions are also used when there is a main question posed but with a variating subject.

6. Data Analysis

Graff and Van Wyk (2003) define three factors which have negative impact on producing secure coding among software developers: technical factor, physiological factor and real-world factor. The technical factor means technical complexity of software which makes it difficult to produce really a secure code. In terms of the physiological factor, a certain type of mental model or mindset is needed to produce a secure code. This can be really hard to adapt by software developers. In every software development project, there are real-world constrains or real-world factors, such as time pressure or low secure coding requirements from customers. (Graff & Van Wyk, 2003.)

Author's personal hypothesis was that subjects whom feel more responsibility in their work and who self-educate will indicate intention towards secure coding.

Firstly, personnel from different-sized enterprises answered the questionnaire. The focus group of this study includes subjects who work in SMEs. The answers were narrowed to the subjects who work in a micro enterprise (<10) or in small enterprise (<50).

Secondly, data analysing is focused on the subjects who answered the question of regularity of their programming work "*often (i.e. few times in a week)*" or "*Very often (everyday)*". This narrows data to the subjects who perform practical programming work.

The survey request was sent to 17 persons and nine of them provided their answers. Five out of nine matched the focus group criteria.

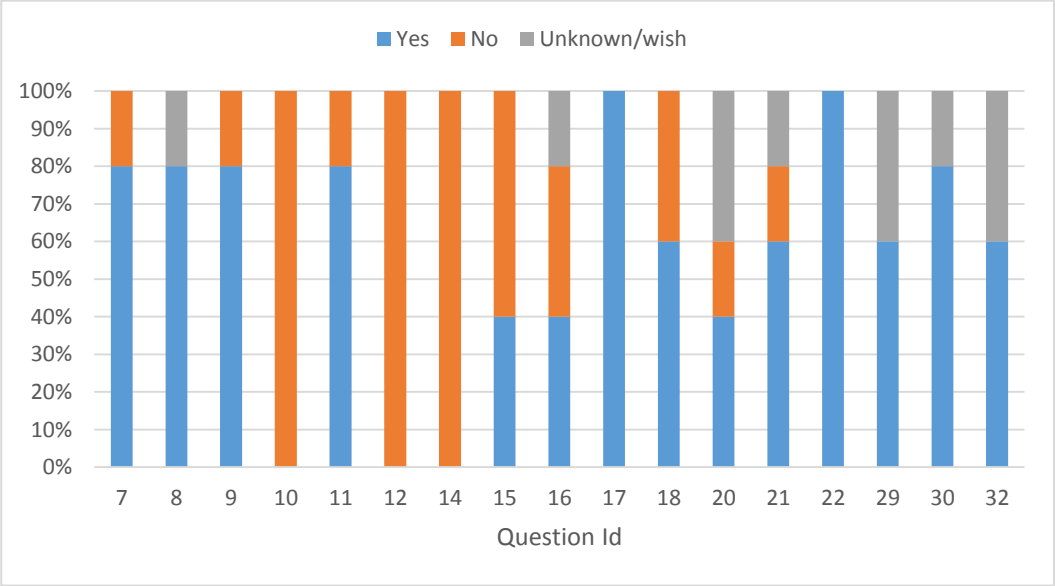

Figure 5. Answers of 3 option questions.

Figure 5 shows questions with 3 answer options: "*Yes*", "*No*" and "*Unknown/wish*". The "*Unknown/wish*" answer reflects subject's lack of knowledge of the particular information, except with Q20 (*"Do you improve your knowledge of programming by i.e., reading books or taking courses in your working hours?"*), Q21 (*"Do you improve*

your knowledge of programming by i.e., reading books or taking courses in your spare time?"), Q29 (*"Do you have hobby projects related to programming? (open source etc.)"*) and Q32 (*"I am likely to use or study secure programming techniques in the future."*). The subjects who responded to these questions wish to know more or they express certain intention to learn more about the questions' information. Figure 5 contains all other categories (awareness Q7-Q19 and PMT Q20-Q32) except for background questions (Q1-Q6) and open questions (Q23 and Q24).

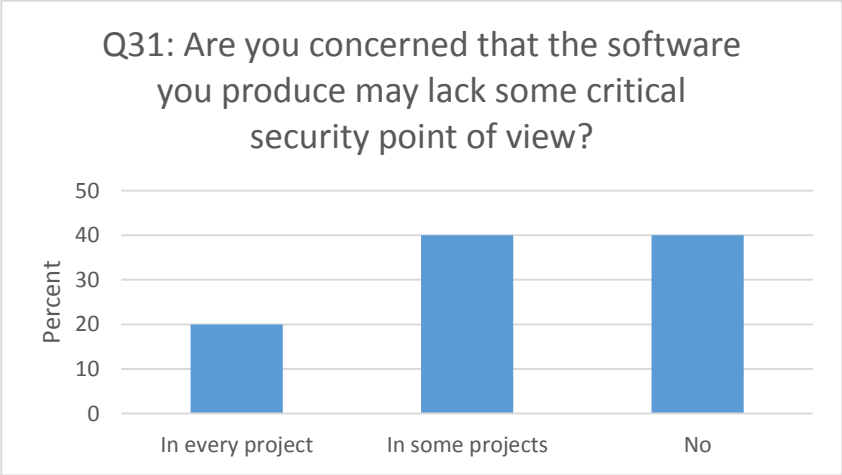

Figure 6. Answers of vulnerability.

Subject's personal vulnerability via produced code delivered different answers (Figure 6). The majority of subjects still recognized own vulnerability ("*In every project*" and "*In some projects*").

On the one hand, all the subjects answered getting intrinsic reward by implement something that they have learned (Figure 5, Q22: *"Do you find satisfying to be able to implement something you have learnt?").* This leads to a conclusion that, an employer should find a way to appreciate subjects' efforts to learn new information because it could encourage them to try to learn even more. On the other hand, this could also influence subjects' motivation in a negative way if the new knowledge they have acquired is not taken into consideration while discussing development improvements.

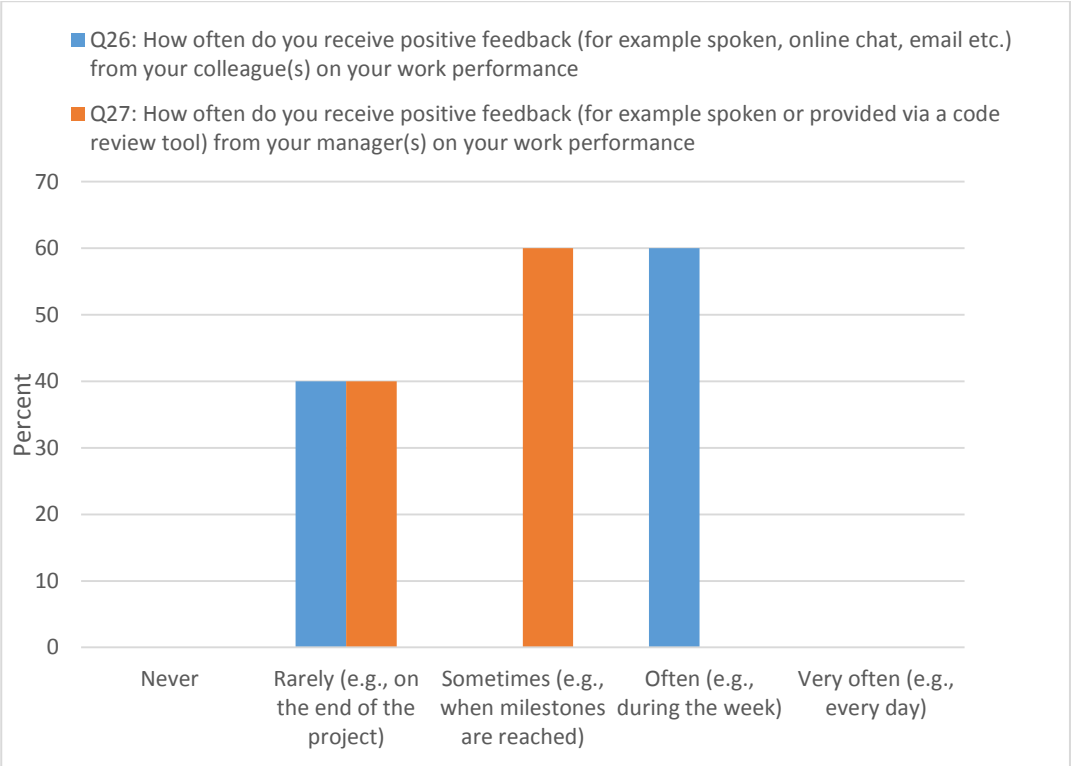

Figure 7. Answers to the feedback questions.

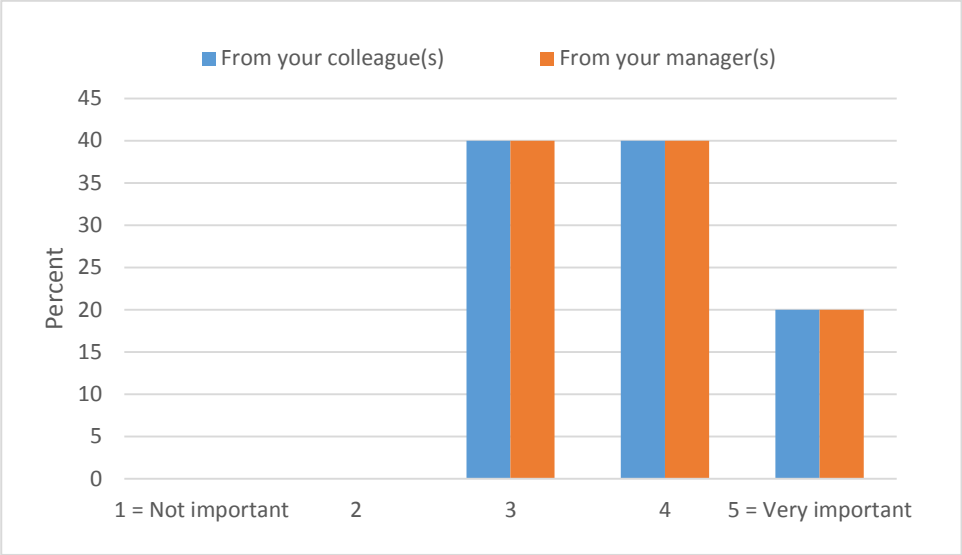

Figure 8. Importance of the feedback (Q28: "*How important to you is the feedback you receive?*").

Answers to PMT extrinsic reward questions are represented in Figure 7 and in Figure 8. They show that the subjects receive more feedback from their colleagues than from their managers, even though both sources of feedback are equally important to them.

Response efficacy was asked with open question (question 23: *"What motivates you to learn about new programming techniques or languages?")* and selection (question 29: *"Do you have hobby projects related to programming? (open source etc.)"*). Answers to open question vary from self-improvement into career improvement, but most of them indicate self-improvement (directly and non-directly). The non-direct answers refer to the need to improve developed code. The selection answer (Figure 5) supports

this finding, as majority of the subjects are involved in an ongoing open source project or are thinking about starting one.

The self-efficacy answers vary depending on the subjects (Figure 5, Q20: *"Do you improve your knowledge of programming by i.e., reading books or taking courses in your working hours?"* and Q21: *"Do you improve your knowledge of programming by i.e., reading books or taking courses in your spare time?"*). The answer to Q21 has the most value when subject's self-efficacy is measured. This is because using personal free time for learn new can be perceived as greater barrier than using work time for the same purpose. Over half of the subjects admitted that they use their free time to learning new.

The open question 24 (*"What media/tools or sources do you prefer to use while learning new things related to programming?"*), was meant to find out about the source of information, which are used to learn new. The most common answer turned out to be "*internet*". Some answered a website which works as a discussion forum for programmers. Most likely, the answers, such as "*internet*" or a website prove the point that a subject perceives problem solving as a way to learn new.

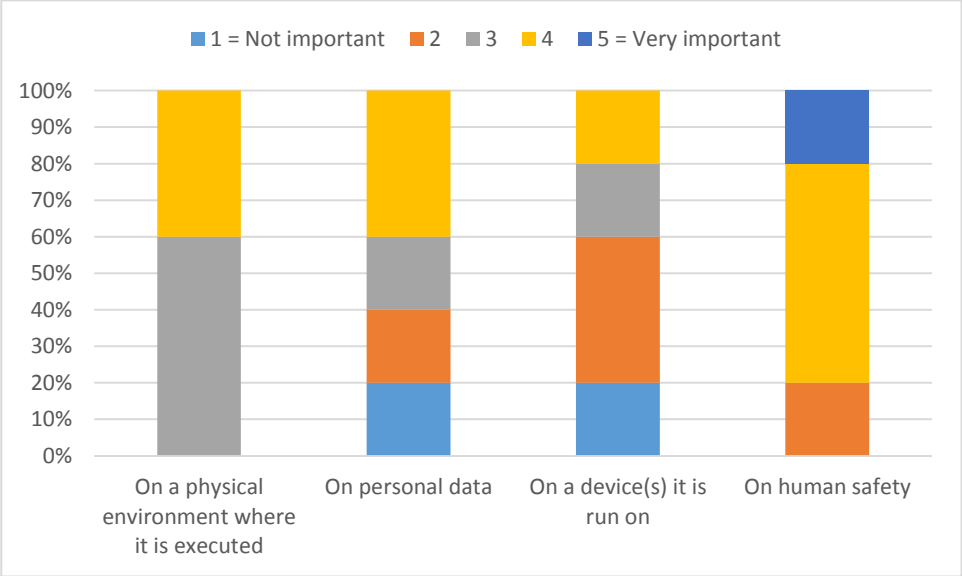

Figure 9. Answers on software severity (Q25: *"How critical do you see your software's impact?"*).

The way a subject feels about the severity of own software can be seen from Figure 9. The subjects do not feel that their software is threating physical world but again they attach high value to the severity of human safety. The severity answers that refer to personal data differ from "*Not important*" into rate 4, which could result from the difference in understanding personal data.

The response cost question (Figure 5, Q30: *"Would you be ready to produce a secure code even if it required more effort?"*) can be interpreted in a way that 80% would like to put more effort to produce a secure code and the rest would like to try.

Most of the focus group subjects answered (Figure 5, Q7: *"Do you know what secure programming / coding is?"*) that they know what secure coding means. Which indicates that subjects consider having existing knowledge of it. Only a small percentage was not aware of it. Even if most of the subjects have pre-existing knowledge of the topic, over half of them would still like to deepen their expertise (Figure 5, Q32: *"I am likely to use or study secure programming techniques in the future."*). The rest of them were unsure as they responded with "*maybe*" to the same question. This indicates that most of the subjects have intention towards studying secure coding.

7. Discussion

It is interesting to find out that the subjects see problem solving as a way of learning new things. It could be that these subjects do not systematically self-study broader topics but improve own existing programming skills. A downside of using i.e., discussion forums as a source of information is that they provide quick and precise answers but lack teaching the holistic view of the problem and solution.

This study aims at narrowing the focus group based on authors own experience and observations of the programming field. Especially observation of programmers working in different sized enterprises. The validity of the focus group is studied by analyzing the answers in the PMT categories.

Table 5. Analysis of the answers of the PMT categories.

PMT section	**Question Id**	**Desired answer from the PMT point of view**	**Answer analysis**
Source of information	24	Available for micro- and small enterprise personnel	Discussion and blog sites
Instrinct reward^{+M}	22	Get satisfaction by using new knowledge	All subjects
Extrinsic reward^{+M}	26, 27, 28	Feedback is perceived as important (at least 4)	60% provide the rate of at least 4
Severity^{-M}	25	One of the question topics should be at least 4	80% provide the rate of at least 4 on "On human safety"
Vulnerability^{-M}	31	Most of the project should be vulnerable	60% provide the rate of at least "In some projects"
Response efficacy^{+A}	23, 29	Positive thinking towards using leisure time for self-improvement	All subjects
Self-efficacy^{+A}	20, 21	Existing characteristics for self-learning during leisure time	60% provide the answer "Yes"
Response cost^{-A}	30	Willingness to work more to increase the quality	80% provide the answer "Yes"
Protection motivation	32	Intention towards secure coding	60% have the intention (40% is unsure)

^{A} Adaptive response
^{M} Maladaptive response
+ Response facilitating factor
$^{-}$ Response inhibiting factor

Answer analysis of each of the PMT category is represented in Table 5, which enables the conclusion that subjects in the selected focus group in all categories express at least 50% likelihood to fulfil the PMT goals in each category. This will most likely will lead to protection motivation. The question that refers to measuring the intention towards secure coding supports this conclusion, as over half of the subjects express the intention towards it.

The subjects that indicated the intention towards secure coding have high correlation (R= 0,92 and P= 0,03) with self-education (Q20: *"Do you improve your knowledge of programming by i.e., reading books or taking courses in your working hours?"* and Q21: *"Do you improve your knowledge of programming by i.e., reading books or taking courses in your spare time?"*). This indicates that those subjects have high self-efficacy (see importance from Chapter 5.3). When the same analysis is done for all the respondents, high correlation occurs (R= -0,78 and P=0,01) in positive feedback (Q28: *"How important to you is the feedback you receive?"*). The focus group has R= -0,87/P= 0,06 correlation with the same question. This indicates that the subjects who express the intention towards secure coding (the *"Yes"* answer has value 1) value positive feedback more. However, it should be noted that the amount of analysed data is small.

The research question of this study is as follows: Can secure programming intention be aroused with a PMT questionnaire? This study has succeeded in producing the intention towards secure programming in the case of half of the subjects. The results show that using this questionnaire as a tool to increase secure code intention is effective. However, this hypothesis should still be verified with the use of a higher number of respondents.

8. Conclusions

The most effective way to train software developers to be more secure minded is to describe the problem and then demonstrate the importance of this problem as well as its impact (McGraw, 2004). Secure coding should not be separated from normal software development. (Kumar, Pandey, & Ahson, 2007), which is a corner stone of this study

PMT towards the suggested focus group (micro and small enterprises) is highly effective as most of the PMT categories are answered in a theory expected manner. Most subjects also indicate the intention towards secure coding in the end of the survey. This intention is supported with strong self-efficacy among them. Another interesting finding is that many subjects (also outside of the focus group) see great importance of positive feedback (Chapter 7).

8.1 Research limitations

Overall answer rate was low. It could be that the subjects perceived the questions as too sensitive in their company/project and that is why they did not continue with answering the questions. It is also possible that the questionnaire was too long for them. The combination of a low answer rate and a qualitative research method makes it impossible to generalize the results of this study. Generalisation of results is also limited because of the fact that the material for the analysis was provided by a selected focus group in a certain kind of software development organization or company.

It also worth of noticing that PMT's response inhibiting factors have not been studied as they would require projected manipulation of the subjects, who would then make the estimation themselves.

The difference between software security awareness and software security knowledge should also be recognized. Many studies imply that measuring awareness reflects developers secure coding capability which is based on practical knowledge.

It should be noticed, that within this study, it is difficult to know if subjects had existing intention towards secure coding. This could be measured with an intervention research.

8.2 Future research

Future studies could implement a quantitative survey which can be analysed in statistic methods. The focus group should still be personnel in micro and small enterprise, because answers analysis of this study indicates that PMT is effective within a particular focus group.

The intention towards secure coding among subjects could be increased by increasing vulnerability. It could be done by using more detailed questions. For example, by asking programming language specific questions to increase subject's reflection on own practical work. I.e., answer to programming questions would also contain secure coding facts.

Protection motivation could be achieved by an active questionnaire with customized content based on a subject's technical background. This would require more focused and more technical questions, which would increase the effectiveness of response inhibiting factors.

Intention change towards secure coding could be ensured with a follow-up study. Data gathered with the use of the follow-up study would help improve the effectiveness of PMT components and recognise the intention of potential subjects towards secure coding.

References

Anderson, R. J. (2008). *Security Engineering: A Guide to Building Dependable Distributed Systems.* John Wiley Sons.

Arbaugh, W., Fithen, W., & McHugh, J. (2000). Windows of vulnerability: a case study analysis. *Computer*, 52-59.

Banerjee, C., & Pandey, S. K. (2010). Research on software security awareness. *ACM SIGSOFT Software Engineering Notes, 35*(5), 1-5. doi:10.1145/1838687.1838701

Bishop, M., & Frincke, D. (2005). Teaching secure programming. *IEEE security & privacy*, 54-56.

Boss, S., Galletta, D., Lowry, P., Moody, G., & Polak, P. (2015). What Do Systems Users Have to Fear? Using Fear Appeals to Engender Threats and Fear that Motivate Protective Security Behaviors. *MIS Quarterly, 39*(4), 837–864.

Byers, D., & Shahmehri, N. (2007). Design of a Process for Software Security. *The Second International Conference on Availability, Reliability and Security (ARES'07)* (pp. 301-309). Vienna: IEEE.

Crossler, R., Johnston, A., Lowry, P., Hu, Q., Warkentin, M., & Baskerville, R. (2013). Future directions for behavioral information security research. *Computers & Security, Volume 32*, 90-101.

European Commission. (2014, October 5). *Recommendation 2003/361/EC*. Retrieved from EUR - Lex, Access to European Union law: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32003H0361&from=EN

Flick, U. (2008). *Designing Qualitative Research.* London: Sage Publications Ltd. doi:10.4135

Flick, U. (2018). *The SAGE Handbook of Qualitative Data Collection.* London: SAGE Publications Ltd. doi:10.4135/9781526416070

Floyd, D. L., Prentice-Dunn, S., & Rogers, R. W. (2000). A Meta-Analysis of Research on Protection Motivation Theory. *Journal of Applied Social Psychology, 30*(2), 407-429. doi:10.1111/j.1559-1816.2000.tb02323.x

Futcher, L., & Von Solms, R. (2008). Guidelines for secure software development. *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries riding the wave of technology - SAICSIT '08* (pp. 56-65). New York: ACM Press.

Gilliam, D., Kelly, J., Powell, J., & Bishop, M. (2001). Development of a software security assessment instrument to reduce software security risk. *Proceedings*

Tenth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. WET ICE 2001 (pp. 144-149). IEEE.

Gilliam, D., Wolfe, T., Sherif, J., & Bishop, M. (2003). Software security checklist for the software life cycle. *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003* (pp. 243-248). IEEE.

Graff, M. G., & Van Wyk, K. R. (2003). *Secure coding: principles and practices.* Sebastopol: O'Reilly & Associates.

Gregoire, J., Buyens, K., Win, B. D., Scandariato, R., & Joosen, W. (2007). On the Secure Software Development Process: CLASP and SDL Compared. *Third International Workshop on Software Engineering for Secure Systems (SESS'07: ICSE Workshops 2007)*, 1-7.

Hoglund, G., & McGraw, G. (2004). *Exploiting Software: How to Break Code.* Addison-Wesley.

Holmes, R., & Walker, R. J. (2008). Promoting developer-specific awareness. *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering - CHASE '08* (pp. 61-64). New York, USA: ACM Press. doi:10.1145/1370114.1370130

Jones, R. L., & Rastogi, A. (2004). Secure Coding: Building Security into the Software Development Life Cycle. *Information Systems Security*, 29-39.

Jürjens, J. (2002). UMLsec : Extending UML for Secure Systems Development. In *≪UML≫ 2002 — The Unified Modeling Language* (pp. 412-425). Springer Berlin Heidelberg.

Kissel, R., Stine, K., Scholl, M., Rossman, H., Fahlsing, J., & Gulick, J. (2008, October). *Security Considerations in the Information System Development Lifecycle.* Retrieved from National Institute of Standards and Technology: http://csrc.nist.gov/publications/nistpubs/800-64-Rev2/SP800-64-Revision2.pdf

Kumar, R., Pandey, S. K., & Ahson, S. I. (2007). Security in Coding Phase of SDLC. *2007 Third International Conference on Wireless Communication and Sensor Networks* (pp. 118-120). IEEE. doi:10.1109/WCSN.2007.4475760

Lipner, S. (2010). Security Development Lifecycle - Security Considerations for Client and Cloud Applications. *Datenschutz und Datensicherheit - DuD*, 135-137.

Lodderstedt, T., Basin, D., & Doser, J. (2002). SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *≪UML≫ 2002 — The Unified Modeling Language* (pp. 426-441). Springer Berlin Heidelberg.

Mahadevan, L., Simon, J., & Meservy, T. (2011). Effects of developer cognitive style and motivations on information security policy compliance. *17th Americas Conference on Information Systems 2011, AMCIS 2011*, 1-8.

Marshall, C., & Rossman, G. B. (2006). *Designing Qualitative Research* (4th ed.). London: Sage Publications Ltd.

McGraw, G. (2002). Managing software security risks. *Computer, Volume 35, Issue 4*, 99-101.

McGraw, G. (2004). Software security. *IEEE Security & Privacy Magazine*, 80-83.

McGraw, G. (2006). *Software Security: building security in.* Boston: Pearson Education Inc.

Microsoft. (2015). *Security Development Lifecycle*. Retrieved 7 29, 2015, from Microsoft: https://www.microsoft.com/en-us/sdl/default.aspx

Mitnick, K. D., & Simon, W. L. (2002). *The Art of Deception: Controlling the Human Element of Security.* Indianapolis, Indiana: John Wiley & Sons.

Rippetoe, P. A., & Rogers, R. W. (1987). Effects of components of protection-motivation theory on adaptive and maladaptive coping with a health threat. *Journal of personality and social psychology, 52*(3), 596-604. doi:10.1037/0022-3514.52.3.596

Rogers, R. (1983). Cognitive and physiological processes in attitude change: A revised theory of protection motivation. *Social Psychophysiology*(July), 153-176.

Rogers, R. W. (1975). A Protection Motivation Theory of Fear Appeals and Attitude Change1. *The Journal of Psychology, 91*(1), 93-114.

Sarma, A., Hoek, A., & Redmiles, D. F. (2007). A Comprehensive Evaluation of Workspace Awareness in Software Configuration Management Systems. *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007)* (pp. 23-26). Coeur d'Alene, ID: IEEE. doi:10.1109/VLHCC.2007.7

Silverman, D. (2005). *Doing Qualitative Research* (2nd ed.). London: SAGE Publications Ltd.

Steven, J., & Peterson, G. (2006). Essential Factors for Successful Software Security Awareness Training. *Security & Privacy, IEEE, 4*(5), 80 -83. doi:10.1109/MSP.2006.119

Swamy, N., Chen, J., Fournet, C., Strub, P.-Y., Bhargavan, K., & Yang, J. (2011). Secure distributed programming with value-dependent types. *Proceeding of the 16th ACM SIGPLAN international conference on Functional programming - ICFP '11* (pp. 266-278). New York: ACM.

Swanson, M., & Guttman, B. (1996, September). *Generally Accepted Principles and Practices for Securing Information Technology Systems.* Retrieved from National Institute of Standards and Technology: http://csrc.nist.gov/publications/nistpubs/800-14/800-14.pdf

Tøndel, I. A., Jaatun, M. G., & Meland, P. H. (2008). Security Requirements for the Rest of Us: A Survey. *Software, IEEE, 25*(1), 20 - 27. doi:10.1093/rsq/11.2.55

Van Wyk, K., & McGraw, G. (2005). Bridging the Gap between Software Development and Information Security. *IEEE Security and Privacy Magazine, 3*(5), 75-79. doi:10.1109/MSP.2005.118

Webropol. (2019, 11 25). *Webropol*. Retrieved from Webropol: https://webropol.fi/

Whitney, M., Lipford, H. R., Chu, B., & Zhu, J. (2015). Embedding Secure Coding Instruction into the IDE. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education - SIGCSE '15* (pp. 60-65). New York, USA: ACM Press. doi:10.1145/2676723.2677280

Viega, J., & McGraw, G. (2002). *Building Secure Software.* Boston: Addison-Wesley.

Wikipedia. (2017, 1 9). *Secure coding*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Secure_coding

Woon, I., Tan, G.-W., & Low, R. (2005). A protection motivation theory approach to home wireless security. *Proceedings of the Twenty-Sixth International Conference on Information Systems* (pp. 367-380). Las Vegas: International Conference on Information Systems.

Yasinsac, A., & McDonald, J. (2006). Foundations for security aware software development education. *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)* (p. 219c). IEEE.

Zenah, N., & Aziz, N. (2011). Secure coding in software development. *2011 Malaysian Conference in Software Engineering* (pp. 458-464). Johor Bahru: IEEE. doi:10.1109/MySEC.2011.6140716

Zhu, J., Lipford, H. R., & Chu, B. (2013). Interactive support for secure programming education. *Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE '13* (pp. 687-692). New York, USA: ACM Press. doi:10.1145/2445196.2445396

Zimmerman, B. J. (2000). Self-Efficacy: An Essential Motive to Learn. *Contemporary educational psychology, 25*(1), 82-91. doi:10.1006/ceps.1999.1016

Appendix A. Questions

Background questions:

Table 6. Background questions.

Id	Question	Type [1]	Answer options
1	What is your age?	S	<25 26-30 31-40 41-50 50<
2	What is your education level?	S	Doctoral or higher degree Master's degree Bachelor's degree (University) Bachelor's degree (Applied sciences) Lower than bachelor's degree
3	How many employees are there in your company (or in a typical client company)?	S	<10 <50 <250 >250
4	How long (years) have you been working in the field of programming?	S	None/Student <1 2-5 6-10 11-20 20<
5	What is your work role(s)? (rank roles if you have many) 1 = primary, 2 = secondary etc.	R2	Programmer Team Leader Manager Designer/Architect
6	How often do you do practical programming in your work?	S	Never Rarely (i.e., once a month) Sometimes (i.e., a few times in a month) Often (i.e., a few times in a week) Very often (everyday)

[1] S=Selection, O=Open, M1=Multiple choice, M2=Matrix, R1=Rating, R2=Ranking

Awareness questions

Table 7. Awareness and knowledge questions.

Id	Question	Type [1]	Answer options
7	Do you know what secure programming / coding is?	S	Yes No I have heard about it
8	Have you used secure programming principles and practices in software development?	S	Yes No I do not know
9	Does your company use information security policies?	S	Yes No I do not know
10	Do you use vulnerability lists in your work?	S	Yes No I do not know
11	Do you use code analysis and/or secure programming tools in your work?	S	Yes No I do not know
12	Do you use secure development lifecycle (SDL) in your work?	S	Yes No I do not know
13	What kind of software development lifecycle method is used in your projects?	S	One standard method (i.e., scrum, kanban) Several standard methods (i.e., depends on a project/customer) Customized method (i.e., contains parts of different methods) None
14	Do you use secure software development standards in your work? (i.e., ISO/IEC 27002)	S	Yes No I do not know
15	Do you use "secure by design" –design method in your work?	S	Yes No I do not know
16	Do you use integrated development environment (IDE) tools with software security promoting features?	S	Yes No I do not know
17	Do you use code reviews in your work?	S	Yes No I do not know
18	Do you know what an "attack pattern" is?	S	Yes No I do not know
19	How do you select a programming language for a project?	M1	Based on an execution platform Based on own programming skills Based on the security of language Based on customer requirements/existing codebase

[1] S=Selection, O=Open, M1=Multiple choice, M2=Matrix, R1=Rating, R2=Ranking

PMT –based questions

Table 8. PMT questions.

Id	Question	Type[1]	Answer option
20	Do you improve your knowledge of programming by i.e., reading books or taking courses in your working hours?	S	Yes No I would like to
21	Do you improve your knowledge of programming by i.e., reading books or taking courses in your spare time?	S	Yes No I would like to
22	Do you find satisfying to be able to implement something you have learnt?	S	Yes No
23	What motivates you to learn about new programming techniques or languages?	O	
24	What media/tools or sources do you prefer to use while learning new things related to programming?	O	
25	How critical do you see your software's impact? - On a physical environment where it is executed - On personal data - On a device(s) it is run on - On human safety	M2	1 = Not critical 2 = 3 = 4 = 5 = Very critical
26	How often do you receive positive feedback (for example spoken, online chat, email etc.) from your colleague(s) on your work performance	S	Never Rarely (e.g., on the end of the project) Sometimes (e.g., when milestones are reached) Often (e.g., during the week) Very often (e.g., every day)
27	How often do you receive positive feedback (for example spoken, online chat, email etc.) from your manager(s) on your work performance	S	Never Rarely (i.e., on the end of the project) Sometimes (i.e., when milestones are reached) Often (i.e., during the week) Very often (i.e., every day)
28	How important to you is the feedback you receive? From your colleague(s) From your manager(s)	M2	1 = Not important 2 = 3 = 4 = 5 = Very important
29	Do you have hobby projects related to programming? (open source etc.)	S	Yes No I would like to start one
30	Would you be ready to produce a secure code even if it required more effort?	S	Yes No I would like to try
31	Are you concerned that the software you produce may lack some critical security point of view?	S	In every project In some projects No
32	I am likely to use or study secure programming techniques in the future.	S	Yes No Maybe

[1] S=Selection, O=Open, M1= Multiple choice, M2=Matrix, R1=Rating, R2=Ranking

Appendix B. Search Queries

Table 9. Keywords used in search strategy.

Keyword	Variants	Category	Search word
secure	security	Concept	secur*
defensive		Concept	defensive
coding	code	Concept	cod*
programming	program	Concept	program*
development	develop	Concept	develop*
developer	developers	Subject	developer*
programmer	programmers	Subject	programmer*
designer	designers	Subject	designer*
coder	coders	Subject	coder*
survey		Data collection	survey
questionnaire	question questions	Data collection	question*
interview		Data collection	interview
Protection Motivation Theory	Protection motivation PMT	Theory	“protection motivation” PMT

Table 10. Parts of search query.

Query part	**Priority**
(secur* OR defensive) AND (cod* OR program* OR develop*)	1
(“protection motivation” OR pmt)	2
(survey OR question* OR interview)	3
(coder* OR programmer* OR developer* OR designer*)	4

Table 11. Search priority 4

Database	Search query	Results	Noted publications
Scopus	TITLE-ABS-KEY (secur* OR defensive) AND TITLE-ABS-KEY (cod* OR program* OR develop*) AND TITLE-ABS-KEY("protection motivation" OR PMT) AND TITLE-ABS-KEY(survey OR question* OR interview) AND TITLE-ABS-KEY (coder* OR programmer* OR developer* OR designer*)	0	-
Web of Science	TS= ((secur* OR defensive) AND (cod* OR program* OR develop*) AND ("protection motivation" OR PMT) AND (survey OR question* OR interview) AND (coder* OR programmer* OR developer* OR designer*))	0	-
ProQuest	all(secur* OR defensive) AND all(cod* OR program* OR develop*) AND all("protection motivation" OR PMT) AND all(survey OR question* OR interview) AND all(coder* OR programmer* OR developer* OR designer*)	0	-
IEE Electric Library	("Abstract": (secur* OR defensive) AND (cod* OR program* OR develop*) AND ("protection motivation" OR PMT) AND (survey OR question* OR interview) AND (coder* OR programmer* OR developer* OR designer*))	0	-

Table 12. Search priority 3.

Database	Search query	Results	Noted publications
Scopus	TITLE-ABS-KEY (secur* OR defensive) AND TITLE-ABS-KEY (cod* OR program* OR develop*) AND TITLE-ABS-KEY("protection motivation" OR PMT) AND TITLE-ABS-KEY(survey OR question* OR interview)	34	-
Web of Science	TS= ((secur* OR defensive) AND (cod* OR program* OR develop*) AND ("protection motivation" OR PMT) AND (survey OR question* OR interview))	46	-
ProQuest	all(secur* OR defensive) AND all(cod* OR program* OR develop*) AND all("protection motivation" OR PMT) AND all(survey OR question* OR interview)	32	-
IEE Electric Library	("Abstract": (secur* OR defensive) AND (cod* OR program* OR develop*) AND ("protection motivation" OR PMT) AND (survey OR question* OR interview))	0	-

Table 13. Search priority 2.

Database	Search query	Results	Noted publications
Scopus	TITLE-ABS-KEY (secur* OR defensive) AND TITLE-ABS-KEY (cod* OR program* OR develop*) AND TITLE-ABS-KEY("protection motivation" OR PMT)	111	-
Web of Science	TS= ((secur* OR defensive) AND (cod* OR program* OR develop*) AND (“protection motivation” OR PMT))	112	-
ProQuest	all(secur* OR defensive) AND all(cod* OR program* OR develop*) AND all("protection motivation" OR PMT)	167	-
IEE Electric Library	("Abstract": (secur* OR defensive) AND (cod* OR program* OR develop*) AND ("protection motivation" OR PMT))	0	-

Appendix C. Online Survey

Use of secure coding

1. What is your age?

- ◯ <25
- ◯ 26-30
- ◯ 31-40
- ◯ 41-50
- ◯ 50<

2. What is your education level?

- ◯ Doctoral or higher degree
- ◯ Master's degree
- ◯ Bachelor's degree (University)
- ◯ Bachelor's degree (Applied sciences)
- ◯ Lower than bachelor's degree

3. How many employees are there in your company (or in a typical client company)?

- ◯ <10
- ◯ <50
- ◯ <250
- ◯ >250

4. How long (years) have you been working in the field of programming?

- ◯ None/Student
- ◯ <1
- ◯ 2-5
- ◯ 6-10
- ◯ 11-20
- ◯ 20<

5. What is your work role(s)? (rank roles if you have many)

1 = primary, 2 = secondary etc.

Programmer	◯ 1 ◯ 2 ◯ 3 ◯ 4
Team Leader	◯ 1 ◯ 2 ◯ 3 ◯ 4
Manager	◯ 1 ◯ 2 ◯ 3 ◯ 4
Designer/Architect	◯ 1 ◯ 2 ◯ 3 ◯ 4

6. How often do you do practical programming in your work?

- ◯ Never
- ◯ Rarely (i.e., once a month)
- ◯ Sometimes (i.e., a few times in a month)
- ◯ Often (i.e., a few times in a week)
- ◯ Very often (everyday)

7. Do you know what secure programming / coding is?

- ◯ Yes
- ◯ No
- ◯ I have heard about it

8. Have you used secure programming principles and practices in software development?

- ◯ Yes
- ◯ No
- ◯ I do not know

9. Does your company use information security policies?

- ◯ Yes
- ◯ No
- ◯ I do not know

10. Do you use vulnerability lists in your work?

- ◯ Yes
- ◯ No
- ◯ I do not know

11. Do you use code analysis and/or secure programming tools in your work?

- ◯ Yes
- ◯ No
- ◯ I do not know

12. Do you use secure development lifecycle (SDL) in your work?

- ◯ Yes
- ◯ No

- ◯ I do not know

13. What kind of software development lifecycle method is used in your projects?

- ◯ One standard method (i.e., scrum, kanban)
- ◯ Several standard methods (i.e., depends on a project/customer)
- ◯ Customized method (i.e., contains parts of different methods)
- ◯ None

14. Do you use secure software development standards in your work? (i.e., ISO/IEC 27002)

- ◯ Yes
- ◯ No
- ◯ I do not know

15. Do you use "secure by design" –design method in your work?

- ◯ Yes
- ◯ No
- ◯ I do not know

16. Do you use integrated development environment (IDE) tools with software security promoting features?

- ◯ Yes
- ◯ No
- ◯ I do not know

17. Do you use code reviews in your work?

- ◯ Yes
- ◯ No
- ◯ I do not know

18. Do you know what an "attack pattern" is?

- ◯ Yes
- ◯ No
- ◯ I do not know

19. How do you select a programming language for a project?

- ☐ Based on an execution platform
- ☐ Based on own programming skills
- ☐ Based on the security of language
- ☐ Based on customer requirements

20. Do you improve your knowledge of programming by i.e., reading books or taking courses in your working hours?

- ◯ Yes
- ◯ No
- ◯ I would like to

21. Do you improve your knowledge of programming by i.e., reading books or taking courses in your spare time?

- ◯ Yes
- ◯ No
- ◯ I would like to

22. Do you find satisfying to be able to implement something you have learnt?

- ◯ Yes
- ◯ No

23. What motivates you to learn about new programming techniques or languages?

24. What media/tools or sources do you prefer to use while learning new things related to programming?

25. How critical do you see your software's impact?

1 = Not critical, 5 = Very critical

	1	2	3	4	5
On a physical environment where it is executed	◯	◯	◯	◯	◯
On personal data	◯	◯	◯	◯	◯
On a device(s) it is run on	◯	◯	◯	◯	◯
On human safety	◯	◯	◯	◯	◯

26. How often do you receive positive feedback (for example spoken, online chat, email etc.) from your colleague(s) on your work performance

- ◯ Never
- ◯ Rarely (e.g., on the end of the project)
- ◯ Sometimes (e.g., when milestones are reached)
- ◯ Often (e.g., during the week)
- ◯ Very often (e.g., every day)

27. How often do you receive positive feedback (for example spoken or provided via a code review tool) from your manager(s) on your work performance

- ◯ Never
- ◯ Rarely (e.g., on the end of the project)

- ◯ Sometimes (e.g., when milestones are reached)
- ◯ Often (e.g., during the week)
- ◯ Very often (e.g., every day)

28. How important to you is the feedback you receive?

1 = Not important, 5 = Very important

	1	2	3	4	5
From your colleague(s)	◯	◯	◯	◯	◯
From your manager(s)	◯	◯	◯	◯	◯

29. Do you have hobby projects related to programming? (open source etc.)

- ◯ Yes
- ◯ No
- ◯ I would like to start one

30. Would you be ready to produce a secure code even if it required more effort?

- ◯ Yes
- ◯ No
- ◯ I would like to try

31. Are you concerned that the software you produce may lack some critical security point of view?

- ◯ In every project
- ◯ In some projects
- ◯ No

32. I am likely to use or study secure programming techniques in the future.

- ◯ Yes
- ◯ No
- ◯ Maybe