International Journal of Business, Human and Social Sciences
ISSN: 2517-9411
Vol:10, No:8, 2016

A Review on Factors Influencing Implementation of Secure Software Development Practices

Sri Lakshmi Kanniah, Mohd Naz'ri Mahrin

***Abstract*—**More and more businesses and services are depending on software to run their daily operations and business services. At the same time, cyber-attacks are becoming more covert and sophisticated, posing threats to software. Vulnerabilities exist in the software due to the lack of security practices during the phases of software development. Implementation of secure software development practices can improve the resistance to attacks. Many methods, models and standards for secure software development have been developed. However, despite the efforts, they still come up against difficulties in their deployment and the processes are not institutionalized. There is a set of factors that influence the successful deployment of secure software development processes. In this study, the methodology and results from a systematic literature review of factors influencing the implementation of secure software development practices is described. A total of 44 primary studies were analysed as a result of the systematic review. As a result of the study, a list of twenty factors has been identified. Some of factors that affect implementation of secure software development practices are: Involvement of the security expert, integration between security and development team, developer's skill and expertise, development time and communication between stakeholders. The factors were further classified into four categories which are institutional context, people and action, project content and system development process. The results obtained show that it is important to take into account organizational, technical and people issues in order to implement secure software development initiatives.

***Keywords*—**Secure software development, software development, software security, systematic literature review.

I. Introduction

THE rapid growth of internet and e-commerce has made revolutionary change in peoples' lifestyle and their living standards. Most business organizations run their daily operations and marketing activities through online applications. In the recent years, web applications had become a target of the hackers since networks are closely monitored by firewalls and intrusion detection systems (IDS) [1]. Web application attacks appear as the greatest threat to the security of an organization [2]. Poorly constructed software systems can induce security weaknesses and defects, which, commonly, result in vulnerabilities that can be exploited by malicious users and violate one or more software security properties.

Software insecurity is commonly caused by negligence of security, flaws in the software engineering process, insufficient knowledge and understanding in relation with secure software development. Security should be incorporated during the whole engineering process, beginning from the pre-study phase and ending when the software is in use. However, software developers commonly apply "penetrate and patch" approach to detect and solve vulnerability issues but the root cause of the problem gets ignored. [3]. However, fixing or patching flaws and defects at the later phases of development or during deployment can be very costly and the patch itself can possibly create a new vulnerability in the system. Security is an emergent quality of software which requires advanced planning and careful design [3]. Hence, it is important to ensure that security is integrated into the software development throughout the life cycle.

This paper deals with the identification and listing of influencing factors which affects the implementation of Secure Software Development (SSD) practices. The data identification takes place from the literature by performing systematic literature review (SLR) [4].

The rest of the paper is structured as follows: Section II describes the related work, Section III illustrates the methodology of the study, Section IV comprises of results, Section V consists of findings and discussion, Section VI explains the limitation of the study and Section VII is the conclusion.

Sri Lakshmi Kanniah and Mohd Naz'ri Mahrin are with the Advanced Informatics School, Universiti Teknologi of Malaysia (UTM), Kuala Lumpur, Malaysia (e-mail: lksri2@live.utm.my, mdnazrim@utm.my).

II. Related Work

Research in security covers a varied range of approaches and processes that deal with security during software development. Several actions have been suggested in order to incorporate security in software development life cycle (SDLC) by using different software models. Several modifications have been made to traditional lifecycle by inserting security activities into traditional lifecycle for the purpose of creating security enhanced methodologies and processes [5].

Researchers at University College London have developed 'Appropriate and Effective Guidance in Information Security' (AEGIS), a research model that has integrated security and usability using a spiral model, based on UML. This model defines a UML meta-model of the definition and the rational over the system's assets [6]. AEGIS guides developers to deal with security and usability requirements in system design. The UML meta-model defined by authors identifies assets, the context of operation and supporting the modeling of security requirements. All security decisions in AEGIS are derived from knowledge of assets of the system. Core security activities for system design sessions in AEGIS are: Identification of assets and security requirements, analysis of risk and secure design, and identification of the risks,

vulnerabilities, and threats to the system. The output from these activities is documented in a design document which consist system architecture with all specified countermeasures. In AEGIS, security expertise is absent in development process. Moreover, decision making on the selection of security countermeasures is done by stakeholders. The author's rationale behind this is that decision makers are "better suited to deal with the enforcement of the social requirements of security" while developers are "necessary for the technical implementation of security.

Secure Software Development Model (SSDM) which was developed at the Nigerian University of Agriculture [7] integrates security activities into engineering process, which are: Security training, threat modeling, security specification, review of security specification, and penetration testing. Furthermore, SSDM has separated security specification from functional specification.

The security process 'Comprehensive, Lightweight Application Security Process' — CLASP [8] introduces lightweight process for SSD. CLASP provides structured practices for deriving security requirements of software systems [9]. CLASP outlines seven key best practices, such as: Security awareness, application evaluations, derivation of security requirements, implementation of secure development practices, developing vulnerability remediation measures, defining and monitoring metrics, and publishing operational guidelines. CLASP also specifies a set of activities that should be incorporated in the development lifecycle. CLASP provides roles and security to structure and support the activities in the resources methodology.

'The Microsoft's Security Development Lifecycle (SDL) has incorporated security activities into each development phase of SDLC [10]. Its purpose is to reduce the number of vulnerabilities in software [10]. SDL consists of a set of activities that overcome security issues. The activities in SDL are grouped in phases, which can be mapped to general software development phases.

Seven 'touchpoints' exhibit how software developers can implement them in the development stages. The aim of 'touchpoints' is to increase effectiveness through: code review, architectural risk analysis, penetration testing, risk-based security tests, abuse cases, security requirements, and security operations [9].

Conclusively, the aforementioned models focus on what is needed to build secure software. However, there is lack of research on identifying the factors required for successful implementation of SSD process.

III. Methodology

The SLR was performed based on the guidelines provided by [4]. The SLR process consists of three main phases: Review Planning, Conducting Review and Review Reporting.

A. SLR Research Question

The objective of this review is to identify factors that affect the implementation of SSD practices. In considering this, research question was derived which supports the objective of the review:

- RQ1: What are the factors which may affect the implementation of the secure software development process?

B. Search Strategy

The SLR concentrates on searching in scientific databases. The following sources were selected for the SLR search process:

- Association for Computing Machinery Digital Library
- IEEE Xplore Digital Library
- Springer Link
- Science Direct
- Scopus
- Taylor and Francis Online
- Wiley Online Library

These sources were chosen because they provide a wide range of software engineering related journals and conference proceedings. The search consists of several stages. Fig. 1 shows the review process and the number of papers selected at each stage. In stage 1, we searched the databases using the search terms listed in Table I. Category 1 has more keywords and shows many variations of the same term "Secure Software Development" (SSD). These keywords were combined using the Boolean "AND" operator, to make sure that only article that focuses on both SSD and factors, will be retrieved. That is, we searched every possible combination of one item from Category 1 AND Category 2 in paper's title, keywords, and abstract.

TABLE I
KEYWORDS USED IN THIS REVIEW

Category	Keywords
1	"Secure software development" "Secure development" "Security Development Lifecycle" "Secure Application Development" "Secure Web Development"
2	"factors" "barriers" "motivation" "challenges"

Based on keyword list the following search string was formulated:

(("Secure software development" OR "Secure development" OR "Security development lifecycle" OR "Secure application development" OR "Secure web development") AND ("factors" OR "barriers" OR "challenges" OR "motivation"))

C. Conducting the Review and Inclusion Decision

Studies were selected based on inclusion and exclusion criteria. The inclusion criteria determines which study will be included for data extraction where else the exclusion criteria determines which studies are excluded for the review. Table II details out the inclusion and exclusion criteria that have been used for this SLR.

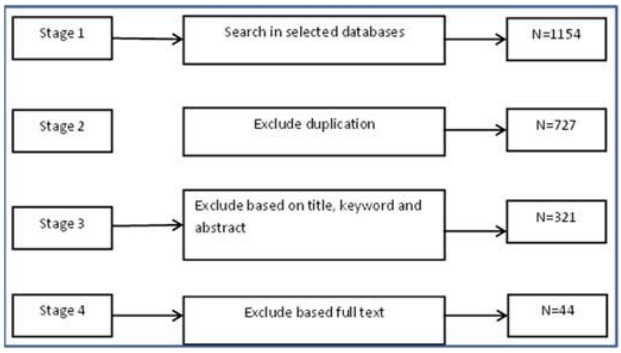

Fig. 1 Study Selection process

Initially, 1154 studies were identified in the Stage 1. Studies identified were managed and stored using EndNote, a citation management tool. In stage 2, studies that met the exclusion criteria were excluded, leaving the selected numbers of studies at 727. During stage 3, the title, abstract and keywords were screened through to determine the relevancy of the study for the review. At the end of stage 3, 321 studies were identified for the next stage. In the stage 4, each study was reviewed and 44 studies were identified as primary studies for the final review.

TABLE II
INCLUSION AND EXCLUSION CRITERIA

Inclusion Criteria	• Studies that are related SSD implementation • Studies that discusses on factors affecting SSD implementation
Exclusion Criteria	• Studies which not written in English • Studies which does not provide full text • Duplicated studies

Once the studies were selected based on inclusion and exclusion criteria, quality assessment was conducted on the primary studies to quantitatively assess the quality of the study. In this SLR, the following four (4) Quality Assessment (QA) questions were defined to ensure the studies address the research topic. The scoring procedure used was Yes (Y) = 1, Partly (P) = 0.5 or No (N) = 0.

1. Does the study address the use of any secure software development practice?
2. Does a paper discuss any real life experience of using secure software development practices?

As there is a lack of existing empirical research, "lesson learned" report based on expert opinion that address the Secure Software Development factors were also considered.

3. Does the study discuss on the research process?
4. Does the paper discuss secure software development implementation factors adequately?

These 4 points provided a measure of the extent to which we are confident that a selected study could make a valuable contribution to understand the factors affecting SSD implementation. Out of 321 studies, 44 studies were finally selected after conducting QA based on the four (4) QA questions.

D.Data Extraction

From the final selected studies, data were extracted using a predefined Data Extraction Form as shown in Table III. Data extraction form is used to collectively record comprehensive information in order to answer the research question. Upon completion of data extraction process, in-depth analysis is done on the gathered information.

TABLE III
KEYWORDS USED IN THIS REVIEW

Data Extracted	Description
Data ID	Unique ID for each study selected
Title	Tile of the study selected
Author	Name of authors who have written the article
Year	Publication year
Type	Publication type (Journal, Conference Proceedings, Book Section)
Publisher	Name of the publisher
Domain	The domain the research was conducted (e.g Health, Public Sector, Telecommunication)
Methodology	Included technique for the design of the study, e.g. case study, survey, experiment, interview to obtain data, observation
Factors	Description of factors that affect the implementation of SSD practices
Practices	Description of practices that are used in SSD
Models	SSDMs adopted/adapted in the study
Phases	Software Development Lifecycle phases in which the SSD implementation occurs

IV. RESULTS

In the final review, 44 primary studies have been selected. Table V presents the distribution of papers selected according to sources in each stage. From the table it can be concluded that IEEE provides the highest percentage (37.2%) of papers relevant to this SLR, followed by Springer Link (23.3%) and Scopus (13%) respectively.

Most papers identified from IEEE and Scopus at initial stage were redundant with papers selected under ACM, Science Direct and Springer Link. In Stage 2 of the review, besides excluding duplications, studies which do not provide full access to the studies were also removed from the list.

V.FINDINGS AND DISCUSSION

SSD implementation consists of a set of predefined activities embedded into SDLC to develop and maintain secured software. Most organizations find it challenging to implement all the activities and put the responsibility on the developers and programmers. Organizations that successfully implement the full SSD lifecycle normally are big organizations that provide commercial software such as Microsoft or highly critical agencies such as banks [11]. Most of the empirical research in this review focuses on selected practices from SSD lifecycle. The most common practices identified from the literature are static analysis, threat modelling, and security requirement engineering.

TABLE IV
DISTRIBUTION OF PAPERS BASED ON SOURCES

Source	Stage 1	Stage 2	Stage 3	Stage 4	Percentage
ACM	102	100	57	5	11.6%
IEEE	524	261	127	16	37.2%
Science Direct	121	121	43	2	4.7%
Scopus	227	105	35	6	14.0%
Springer	111	111	50	10	23.3%
Taylor & Francis	36	15	6	3	7.0%
Wiley Online	33	14	3	1	2.2%
Total	1154	727	321	43	100.0%

TABLE V
LIST OF IDENTIFIED FACTORS ACCORDING TO LITERATURE

No	Factors	References	Freq. (N=44)
1	Developer's skill, experience and knowledge	[12]-[25]	14
2	Security Training and Awareness	[11], [13], [15], [23], [26]-[30]	9
3	Automated tool support	[23], [31]-[36],[56]	8
4	Adequate development time	[13], [14], [18], [22], [23], [31], [35], [37]-[40],	11
5	Developer's Attitude and Motivation	[13], [23], [26], [33]	4
6	Top Management Support	[11], [13], [14], [25], [26], [39], [41],	7
7	Building and retaining a security team	[11], [18], [27], [34], [36], [39], [42]-[44],	9
8	Adequate budget/money/cost	[11], [18], [22]-[24], [27], [35], [37], [38],	9
9	Provide incentives for project team	[26]	1
10	Involvement of security expert in the project	[30], [38], [45]-[47]	5
11	Collaboration between security experts, developers and other stakeholders	[36], [48]-[51]	5
12	Organization's objectives, culture and awareness	[13], [36], [41]	3
13	Security policies, standards and reference guidelines	[13], [25]-[27], [52]	5
14	Change management team	[13], [34]	2
15	Specified internal metrics and KPI	[13], [11]	2
16	Software development methodology	[39], [48]	2
17	Enforcement of policy	[52], [11], [25]	3
18	Project Manager's experience, competence and skill	[15], [44], [24]	3
19	Sufficient numbers of security experts	[18], [46], [40]	3
20	Clear, comprehensive, consistent, unambiguous security requirements	[16], [53], [37], [27], [25]	5

Based on the comprehensive review on the literature, a list of 20 potential factors that affect implementation of SSD practices had been identified. "Factors" in this context of study had been defined as any contributing element that affects the implementation of SSD practices at organizations. Some of the factors can be applicable to software development projects generally. However, as SSD implementation is not technological holistically, some factors identified are specific for implementation of SSD practices. Table V summarizes the review results that listed factors identified in this SLR. Among the factors, 'developer's skill, experience and knowledge', 'adequate development time', 'adequate budget/cost' and 'security training and awareness' were the most frequently cited factors that affects SSD implementation.

The listed factors were further classified into 4 (four) key factors adapted from [54] as shown in Table VI. According to [1], **institutional context, people and action, system development processes and project content** strongly influence system development project outcome which includes development or implementation process. Since SSD implementation is a process to develop and produce secure software, the factors identified from this SLR can be categorized under the given key factors. Each factor had been discussed in detail in the following section.

1) Developer's Skill, Experience and Knowledge

Successful implementation of SSD practices is influenced by developer's knowledge, experience and skill. According to [27], developer's education is critical because lack of understanding on security threats causes the developer to unknowingly introduce more vulnerability during development. Developers require experience and skill to implement secure development practices outlined by organization [17], [19], [51]. Some organizations acquire expensive security tools such as static analysis tools; however, these efforts go in vain as the developers neither have skills nor experience to effectively use them. Reference [22] indicated that it is important that the people involved in the whole of the SDLC understand what needs to be done in order to produce secure applications and software.

2) Security Training and Awareness

Organizations must clearly draw an effective security training and awareness program for their employees [11]. The purpose of this program is to arm the project team with skills and knowledge on SSD implementation. Awareness on the importance of implementing SSD must be given to every stakeholder involved in software development project to ensure required support is provided for SSD initiatives. According to [26], training is a learning process which can help developers to be sensitized to the potential impact of security problems on organizations at large. Educating developers on the need for good coding practices can result in good coding habits that improve security.

3) Automated Tool Support

SSD needs to be supported by automated security development tools. Usage of standard analysis and design tools enables identification of security strategies at design or requirement specification phases of the SDLC [36]. In addition, static analyzers and interactive support for programmers are proven to reduce programming errors [56]. However, automated tool support must be usable, less complex, compatible and customizable to fit into current development environment [33]-[35]. Organization must make sure that developers are given sufficient training and awareness pertaining to the tool and make sure they are following a process that will result in business value [23]. As

mentioned by [23], automating a poor process just gives you poor results faster.

4) Adequate Development Time

Time is highly influential in implementation of SSD practices. This factor is one of the highly cited factors in the review. The common reason for development team to ignore SSD practices is tight deadlines. There has been a trade-off between functionality and project deadlines [37], [22]. Implementing formal SSD methodology is time consuming and developers are normally pressured with short project deadlines [39], [18], [31], [35]. Adequate development time must be given to project team to implement SSD practices efficiently.

TABLE VI
FACTORS BASED ON CATEGORY

Category	Sub Factors
Institutional Context	Change management team
	Enforcement of policy
	Security Training and Awareness
	Provide incentives for project team (S10)
	Organization's objectives, culture and awareness
	Sufficient numbers of security experts
People and Action	Developer's Attitude and Motivation
	Developer's skill, experience and knowledge
	Top Management Support
	Project Manager's Experience, competence and skill
Project Content	Automated tool support
	Adequate budget/money/cost
	Involvement of security expert in the project
	Building and retaining a security team
	Collaboration between security experts, developers and other stakeholders
	Adequate development time
System Development Processes	Software development methodology
	Clear, comprehensive, consistent, unambiguous security requirements
	Security policies, standards and reference guidelines
	Specified internal metrics and KPI

5) Developer's Attitude and Motivation

Developers may have different attitude and value towards security procedures and practices [26]. They need to be consistently motivated to build secure systems. Besides this, developers carry “not my problem” mentality when it comes to securing the system because they do not believe that they make mistakes which cause vulnerabilities in the system [23]. Another study [33] has reported that developer’s inquisitiveness on security tools affect the adoption of the tools. Hence, implementing SSD practices is influenced by developer’s attitude and their motivation.

6) Top Management Support

There is no doubt that top management support has strong influence on implementation of SSD practices. SSD implementation must obtain top management’s approval and support for successful implementation. It’s top management’s responsibility to provide adequate resources for SSD implementation [14]. Reference [13] mentioned that as SSD involves many stakeholders in an organization, senior executives need to mediate between various interest groups to resolve political conflicts when necessary. According to [39], management is the main driver for adoption of SSD methodologies.

7) Building and Retaining a Security Team

Establishing and retaining a security team drives SSD implementation in an organization [24]. The security team must be well trained and skillful in security related matter. They must be to contribute their expertise in development of secure systems. The team must be familiar with security updates and consistently advise the development team on secure development procedures and technology [42], [43], [36]. The organization must retain this security expert team and use their expertise across multiple software projects. This team must be given authority to delay projects if the security requirements are not fulfilled [39], [34].

8) Adequate Budget/Cost

Just like any other software initiatives, cost is considered as an influential factor in SSD implementation. Implementation of SSD also requires adequate budget to be allocated to the project [27], [37], [38], [18]. Security tools are very expensive. Furthermore, when the tools are acquired relevant training must be provided to the developers [35]. Professional trainers need to be hired for this purpose which incurs cost. Due to this constraint, SSD implementation is sidelined by organizations [11].

9) Provide Incentives for Project Team

Providing incentives for project team can reflect on SSD implementation. Providing incentives motivates the developer to be more proactive towards potential vulnerability and risks linked with application development. If management desires and demand clean and secure code and also offer perks for application developers, security will be greatly achievable [26].

10) Involvement of Security Expert in the Project

The role of security expert in project team is very important. It is common to have functionality analyst in the team but functionality analyst and security analyst view the project from different perspectives [45], [30]. Security analyst is able to advice on what adversaries want and where and how attacks are performed [46]. In fact, security expert can make assist in SSD implementation.

11) Collaboration between Security Experts, Developers and Other Stakeholders

SSD implementation typically requires a balanced combination of execution teams. Both technical and business competence must be available in a team. In addition, security experts must be present in the team together and collaborate with software developers throughout the development lifecycle [51]. According to Glisson, it is important for business owners to clearly communicate their expectation to development team so that the team knows what is required to

produce secure software [51]. This indicates that composition of the project teams with effective collaboration and communication between project members are essential for SSD implementation.

12) Organization's Objectives, Culture and Awareness

There is no doubt that organization's objective, culture and awareness affects implementation of SSD [13], [36], [41]. Organizations that produce commercial software pay more attention to SSD implementation to be competitive in the market. The awareness of C-level management on the importance of SSD influences its implementation.

13) Security Policies, Standards and Reference Guidelines

Implementation of SSD is much more successful if proper security policies, standards and guidelines are provided [13], [26], [52]. When the policies are in place, the development team understands what needs to be done. Reference guidelines are very helpful for developers who are not security experts [25]. Developers must be encouraged to document their security experience and use them as reference in other projects [27].

14) Change Management Team

The effective implementation of an SSD practices requires change management strategies and an understanding of organizational culture. Change management techniques will need to be included in the process if you are to gain the support and acceptance of the process among developers, project managers, and system architects [13]. Training and education is an important process in change management. This allows the users to understand the overall concepts of the SSD and ensures their acceptance and readiness to implement SSD practices. There must also be an individual or team with the mandate to alter the organization's development processes [34]. This reduces prejudice among development team in implementing the changes.

15) Specified Internal Metrics and KPI

Establishing internal metrics and key performance indicators that can be used to determine the progress and success of the organization's security evolution has an impact on SSD implementation [13]. Metrics and KPI stands as checkpoint for developers to remind themselves on implementing secure development practices [11].

16) Software Development Methodology

Organizations that employ standard software development methodology finds it easier to implement SSD practices as there is a basic structure in place for them to follow [39]. A large number of activities related to security needs to be performed throughout a software's development lifecycle [48].

17) Enforcement of Policy

SSD implementation requires policy enforcement. Organization must enforce their policy which leaves the developers with no choice but to follow SSD practices [52], [11], [25]. There is no use of having policies which are not followed by employees.

18) Project Manager's Experience, Competence and Skill

Project manager's experience, competence and skill are necessary to ensure SSD implementation. Managers must understand the need for building security into the development life cycle and provide full support for their subordinates to achieve the high-quality bar [44]. Project manager must make sure security requirement are well derived and transferred into development by developers. Managers must check and balance security requirements throughout the development lifecycle. This can ensure that developer implements SSD practices in proper.

19) Sufficient Numbers of Security Experts

Lack of sufficient numbers of security experts hinders SSD implementation in some organizations. Hiring security experts is a costly affair. Currently security experts are bottlenecks in some organizations and they are overburdened with existing tasks. Having insufficient number of security expert causes breakdown in SSD implementation as some practices gets ignored due to the unavailability of security expert [46], [40].

20) Clear, Comprehensive, Consistent, Unambiguous Security Requirements

SSD implementation requires determination of security requirement. Deriving security requirements involves identification of stakeholders. Security requirement must be clear, comprehensive, consistent and unambiguous. This allows stakeholders who are business owners and developers to understand what security practices must be implemented to produce the expected outcome [16], [37]. Security policies and standards can aid in determining the security requirement of the software being developed. Absence of security requirement document will create room for developers to ignore SSD practices.

VI. Limitations

There are some limitations that need consideration in this study. Although the study was conducted according to guidelines suggested in [4], some papers could have missed due to increase of research in the related fields. However, like other SLR studies, this will not be a systematic omission. [55]

During the data extraction process, several papers were found of lacking sufficient details about the reported projects' contextual factors in SSD implementation. The data was synthesized by identifying and categorizing the themes from the papers included in this review. Since some of the selected papers do not provide detailed information, there is a possibility that the extraction process may have resulted in some inaccuracies.

VII. Conclusion

A systematic review was conducted on SSD implementation. The aim of this review is to identify factors that affect the implementation of SSD practices. As a result, 20 factors were identified which were further classified into 4 (four) main factors. Identification of these factors can assist organizations to evaluate their readiness in implementing SSD

and where they should improve. These factors will be integrated with factors derived from practitioners.

ACKNOWLEDGMENT

The authors would like to thank the Universiti Teknologi Malaysia for their support and cooperation including students and other individuals who are either directly or indirectly involved in this project.

REFERENCES

[1] Shuaibu, B.M., et al., *Systematic review of web application security development model.* Artificial Intelligence Review, 2013: p. 1-18.
[2] WhiteHat, *Web Applications Security Statistics Report 2016*. 2016.
[3] Viega, J. and G. McGraw, *Building secure software: how to avoid security problems the right way*. 2001: Pearson Education.
[4] Keele, S., *Guidelines for performing systematic literature reviews in software engineering*. 2007, Technical report, EBSE Technical Report EBSE-2007-01.
[5] Goertzel, K.M. and T. Winograd, *Enhancing the development life cycle to produce secure software.* Technology Analysis Center (IATAC), USA, October, 2008.
[6] Flechais, I., C. Mascolo, and M.A. Sasse, *Integrating security and usability into the requirements and design process.* International Journal of Electronic Security and Digital Forensics, 2007. 1(1): p. 12-26.
[7] Sodiya, A.S., S.A. Onashoga, and O.B. Ajayi, *Towards building secure software systems.* Issues in Informing Science and Information Technology, 2006. 3.
[8] Viega, J., *Security in the Software Development Lifecycle: An introduction to CLASP, the Comprehensive Lightweight Application Security Process.* Secure Software, Inc., McLean, Virginia, USA, White Paper, 2005.
[9] De Win, B., et al., *On the secure software development process: CLASP, SDL and Touchpoints compared.* Information and software technology, 2009. 51(7): p. 1152-1171.
[10] Lipner, S. *The trustworthy computing security development lifecycle*. in *Computer Security Applications Conference, 2004. 20th Annual*. 2004. IEEE.
[11] Chess, B. and B. Arkin, *Software Security in Practice.* Security & Privacy, IEEE, 2011. 9(2): p. 89-92.
[12] Marback, A., et al., *A threat model-based approach to security testing.* Software: Practice and Experience, 2013. 43(2): p. 241-258.
[13] Jones, R.L. and A. Rastogi, *Secure Coding: Building Security into the Software Development Life Cycle.* Information Systems Security, 2004. 13(5): p. 29-39.
[14] Hein, D. and H. Saiedian, *Secure Software Engineering: Learning from the Past to Address Future Challenges.* Information Security Journal: A Global Perspective, 2009. 18(1): p. 8-25.
[15] Allen, J., *Why is Security a Software Issue?*, in *EDPACS*. 2007, Taylor & Francis. p. 1-13.
[16] Mouratidis, H., P. Giorgini, and G. Manson, *Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems*, in *Advanced Information Systems Engineering*, J. Eder and M. Missikoff, Editors. 2003, Springer Berlin Heidelberg. p. 63-78.
[17] Baca, D., et al. *Static Code Analysis to Detect Software Security Vulnerabilities - Does Experience Matter?* in *Availability, Reliability and Security, 2009. ARES '09. International Conference on*. 2009.
[18] Okubo, T., H. Kaiya, and N. Yoshioka. *Mutual Refinement of Security Requirements and Architecture Using Twin Peaks Model*. in *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*. 2012.
[19] Xie, J., B. Chu, and H. Richter Lipford, *Idea: Interactive Support for Secure Software Development*, in *Engineering Secure Software and Systems*, Ú. Erlingsson, R. Wieringa, and N. Zannone, Editors. 2011, Springer Berlin Heidelberg. p. 248-255.
[20] Mockel, C. and A.E. Abdallah. *Threat modeling approaches and tools for securing architectural designs of an e-banking application.* in *Information Assurance and Security (IAS), 2010 Sixth International Conference on*. 2010.
[21] Haron, G.R. and S. Ng Kang. *Extrapolating security requirements to an established software process: Version 1.0.* in *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*. 2011.
[22] Colley, J., *Why Secure Coding is not Enough: Professionals' Perspective*, in *ISSE 2009 Securing Electronic Business Processes*, N. Pohlmann, H. Reimer, and W. Schneider, Editors. 2010, Vieweg+Teubner. p. 302-311.
[23] Payne, J., *Integrating Application Security into Software Development.* IT Professional, 2010. 12(2): p. 6-9.
[24] Davis, N., et al., *Processes for producing secure software.* Security & Privacy, IEEE, 2004. 2(3): p. 18-25.
[25] Jain, S. and M. Ingle. *Techno-management view of Secured Software Development*. in *Software Engineering (CONSEG), 2012 CSI Sixth International Conference on*. 2012.
[26] Raghavan, V.V. and X. Zhang. *Building security in during information systems development*. in *15th Americas Conference on Information Systems 2009, AMCIS 2009*. 2009. San Francisco, CA.
[27] Bartsch, S. *Practitioners' Perspectives on Security in Agile Development*. in *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*. 2011.
[28] Mitropoulos, D., et al., *Countering code injection attacks: A unified approach.* Information Management and Computer Security, 2011. 19(3): p. 177-194.
[29] Chand, P., *Building India as the Destination for Secure Software Development – Next Wave of Opportunities for the ICT Industry*, in *Information Systems Security*, S. Jajodia and C. Mazumdar, Editors. 2005, Springer Berlin Heidelberg. p. 49-65.
[30] Knauss, E., et al., *Supporting Requirements Engineers in Recognising Security Issues*, in *Requirements Engineering: Foundation for Software Quality*, D. Berry and X. Franch, Editors. 2011, Springer Berlin Heidelberg. p. 4-18.
[31] Kleidermacher, D. and M. Wolf. *Using static analysis to improve communications infrastructure*. in *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th*. 2008.
[32] Wurster, G. and P.C.v. Oorschot, *The developer is the enemy*, in *Proceedings of the 2008 workshop on New security paradigms*. 2008, ACM: Lake Tahoe, California, USA. p. 89-97.
[33] Witschey, J., S. Xiao, and E. Murphy-Hill, *Technical and Personal Factors Influencing Developers' Adoption of Security Tools*, in *Proceedings of the 2014 ACM Workshop on Security Information Workers*. 2014, ACM: Scottsdale, Arizona, USA. p. 23-26.
[34] Byers, D. and N. Shahmehri. *Design of a Process for Software Security*. in *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*. 2007.
[35] Jing, X., H.R. Lipford, and C. Bill. *Why do programmers make security errors?* in *Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on*. 2011.
[36] Zia, T.A. and A. Rizvi. *Source Code EMbedded (SCEM) security framework*. in *9th Australian Information Security Management Conference, AISM*. 2011. Perth, WA.
[37] Guan, H., et al., *Environment-Driven Threats Elicitation for Web Applications*, in *Agent and Multi-Agent Systems: Technologies and Applications*, J. O'Shea, et al., Editors. 2011, Springer Berlin Heidelberg. p. 291-300.
[38] Zuccato, A., N. Daniels, and C. Jampathom. *Service Security Requirement Profiles for Telecom: How Software Engineers May Tackle Security*. in *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*. 2011.
[39] Geer, D., *Are Companies Actually Using Secure Development Life Cycles?* Computer, 2010. 43(6): p. 12-16.
[40] Schneider, K., et al., *Enhancing security requirements engineering by organizational learning.* Requirements Engineering, 2012. 17(1): p. 35-56.
[41] Teodoro, N. and C. Serrão. *Web application security: Improving critical web-based applications quality through in-depth security analysis.* in *International Conference on Information Society, i-Society 2011*. 2011. London.
[42] Abramov, J., et al., *A methodology for integrating access control policies within database development.* Computers & Security, 2012. 31(3): p. 299-314.
[43] Alkussayer, A. and W. Allen, *The ISDF Framework: Integrating Security Patterns and Best Practices*, in *Advances in Information Security and Its Application*, J. Park, et al., Editors. 2009, Springer Berlin Heidelberg. p. 17-28.
[44] Bonver, E. and M. Cohen, *Developing and Retaining a Security Testing Mindset.* Security & Privacy, IEEE, 2008. 6(5): p. 82-85.

[45] Riaz, M., et al., *Using templates to elicit implied security requirements from functional requirements - a controlled experiment*, in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 2014, ACM: Torino, Italy. p. 1-10.
[46] Okubo, T. and H. Tanaka, *Web security patterns for analysis and design*, in *Proceedings of the 15th Conference on Pattern Languages of Programs*. 2008, ACM: Nashville, Tennessee, USA. p. 1-13.
[47] Diamant, J., *Resilient Security Architecture: A Complementary Approach to Reducing Vulnerabilities.* Security & Privacy, IEEE, 2011. 9(4): p. 80-84.
[48] Ma, Z., et al., *Model-driven secure development lifecycle.* International Journal of Security and its Applications, 2012. 6(2): p. 443-448.
[49] Zhu, J., et al., *Mitigating Access Control Vulnerabilities through Interactive Static Analysis*, in *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies*. 2015, ACM: Vienna, Austria. p. 199-209.
[50] Karpati, P., G. Sindre, and A. Opdahl, *Visualizing Cyber Attacks with Misuse Case Maps*, in *Requirements Engineering: Foundation for Software Quality*, R. Wieringa and A. Persson, Editors. 2010, Springer Berlin Heidelberg. p. 262-275.
[51] Glisson, W.B. and R. Welland. *Web development evolution: the assimilation of Web engineering security*. in *Web Congress, 2005. LA-WEB 2005. Third Latin American*. 2005.
[52] Díaz, G. and J.R. Bermejo, *Static analysis of source code security: Assessment of tools against SAMATE tests.* Information and Software Technology, 2013. 55(8): p. 1462-1476.
[53] Salini, P. and S. Kanmani, *Model Oriented Security Requirements Engineering (MOSRE) framework for web applications*, in *2nd International Conference on Advances in Computing and Information Technology, ACITY 2012*. 2013: Chennai. p. 341-353.
[54] McLeod, L. and S.G. MacDonell, *Factors that affect software systems development project outcomes: A survey of research.* ACM Computing Surveys (CSUR), 2011. 43(4): p. 24.
[55] Hossain, E., M.A. Babar, and H.-y. Paik. *Using scrum in global software development: a systematic literature review*. in *2009 Fourth IEEE International Conference on Global Software Engineering*. 2009. Ieee.
[56] Xie, J., H.R. Lipford, and B. Chu. Evaluating interactive support for secure programming. in 30th ACM Conference on Human Factors in Computing Systems, CHI 2012. 2012. Austin, TX.