Received: 3 November 2021 | Revised: 27 October 2022 | Accepted: 18 November 2022

DOI: 10.1002/spe.3175

RESEARCH ARTICLE

WILEY

Toward a readiness model for secure software coding

Mamoona Humayun[1] | **Mahmood Niazi**[2,3] | **Noor Zaman Jhanjhi**[4] | **Sajjad Mahmood**[2,3] | **Mohammad Alshayeb**[2,3]

[1]Department of Information Systems, College of Computer and Information Sciences, Jouf University, Al-Jouf, Saudi Arabia

[2]Department of Information and Computer Science, King Fahd University of Petroleum and Minerals, Saudi Arabia

[3]Interdisciplinary Research Centre for Intelligent Secure Systems, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

[4]School of Computer Science and Engineering (SCE), Taylor's University, Selangor, Malaysia

Correspondence
Mamoona Humayun, Department of Information Systems, College of Computer and Information Sciences, Jouf University, Al-Jouf, Saudi Arabia.
Email: mahumayun@ju.edu.sa

Funding information
King Fahd University of Petroleum and Minerals, Grant/Award Number: DF201007

Abstract

The heart of the application's secure operation is its software code. If the code contains flaws, the entire program might be hacked. The issue with software vulnerabilities is that they reveal coding flaws that hackers could exploit. The prevention of cybersecurity issues begins with the program code itself. When writing software code, a software developer must consider expressing the application's architecture and design requirements, keeping the code streamlined and efficient, and ensuring the code is safe. Secure code helps save the system from various cyber-attacks by eliminating the weaknesses that many hacks rely on. To assist the software organization in Secure Software Coding (SSC), this article proposes a readiness model for SSC, namely SSCRM. The proposed model has five levels; SSC challenges and best practices (BP) are mapped at each level. The proposed model will help the organizations better understand SSC challenges and BPs and provide a roadmap for developing secure software code. The proposed model was evaluated using three case studies. The findings demonstrate that the proposed approach helps determine an organization's SSC level.

KEYWORDS

secure coding, secure SDLC, readiness model, case study

1 | INTRODUCTION

Secure software is an application designed or engineered by adhering to secure development standards and practices so that its operations and functions continue to operate normally even when harmful assaults are made against it. Further, the environment's systems and resources remain safe, and threats are identified and mitigated.[1-3] Faulty software may severely harm the organization; therefore, using secure software development practices and adhering to standards is inevitable. Furthermore, the cost of detecting and mitigating threats at the early stages is cheap. Security needs to be integrated into the entire development process to deliver high-quality products, which can be accomplished through a secure software development lifecycle (SSDLC).[4,5]

Abbreviations: BP, best practices; CAs, cyber-attacks; CERT, computer emergency response team; CSSLP, certified secure software lifecycle professional; NIST, National Institute of Standard and Technology; NTT, Nippon Telegraph and Telephone Corporation; OWASP, Open Web Application Security Project; SCPs, software coding professionals; SDL, microsoft security development lifecycle; SDLC, software development lifecycle; SEI, software engineering institute; SSC, secure software coding; SSCRM, readiness model for secure software coding; SSDLC, secure software development lifecycle; SSL/TLS, secure socket layer /transport layer security; XML, extensible markup language.

SSDLC is a framework for building secure software from inception to construction by integrating security testing and other security activities into the existing SDLC. One such example is writing security requirements besides functional requirements and carrying out architectural risk analysis in the design phase. Different organizations follow different SDLC depending on project nature and requirements; however, security practices need to be integrated with the existing SDLC to secure it.[6,7] There exist various SSDLC models; one of the well-known SSDLC models is Microsoft SDL which Microsoft developed; Microsoft SDL outlines 12 best practices (BPs) that organizations may adopt to increase the security of their products.[8,9] NIST also proposed an SDLC framework that defines security processes that an organization may integrate into existing SDLC to achieve secure and quality software.[10]

Previously, security-related tasks were generally conducted solely during testing, the last stage of SDLC. They would not uncover faults, weaknesses, and other vulnerabilities until they were considerably more expensive and time-consuming to fix. Worse, they would not be able to identify any security flaws at all. In short, the patch and penetrate strategy was used to create a patch for the identified flaws.[11,12] According to IBM's Systems Sciences Institute, fixing a defect discovered after implementation costs six times more than fixing one found during design. Furthermore, IBM estimates that the cost of correcting defects found during the testing phase might be 15 times higher than the cost of fixing the same bugs found during the design phase. To conclude, it is considerably better to incorporate security testing throughout the SDLC rather than simply at the end to aid in the early detection and reduction of vulnerabilities. Some of the prevailing benefits of SSDLC include; secure and quality software, early detection and mitigation of flaws, saving time and cost, and low risk.[13]

Incorporating security in all phases of SDLC is important, but it is highly desirable in the coding phase because many exploit rely on this phase. Furthermore, insecure coding methods put your consumers in danger and harm your company's brand. Thus secure software coding (SSC) is inevitable as it helps to prevent the software applications from many cyber-attacks.[14] To defend against cyber-attacks and vulnerabilities, SSC is a set of techniques that apply security considerations to the way software is developed and encrypted.

The frequently exploited software vulnerabilities include defects, errors, bugs, and logic flaws. Security experts have determined that many vulnerabilities are caused by a limited number of common software development difficulties.[15-17] Some common vulnerabilities affecting the code include insufficient logging and monitoring, sensitive data exposure, injection flaws, using insecure components, cross-site scripting, broken authentication, and access control, security misconfiguration, and insecure deserialization. Various BPs of SSC has also been proposed by researchers and practitioners, such as authentication and password management, cryptography, error handling and logging, communication security, data protection, and use of security standards. SSC standards establish protections that minimize or eliminate the possibility of code having security flaws. However, despite using SSC standards and practices, many cyber-attacks are reported daily.[18-20]

The above discussion highlights the significance of SSC for organizations and shows that existing solutions are not sufficient to safeguard the applications from CAs. As a contribution to research, this study aims to identify the BPs and guidelines for SSC and propose a readiness model for SSC (SSCRM) based on identified practices and guidelines. The proposed SSCRM is evaluated using three case studies.

The organization of the remaining article is: Section 2 provides the review of related work and motivation for the current study and discusses the challenges and best practices of secure software coding. Section 3 discusses our proposed methodology, and Section 4 evaluates the proposed methodology with the help of three real-life case studies. Section 5 discusses the study's findings and its practical implications and limitation. The last section concludes the article by providing insights into future work.

2 | MOTIVATION AND RELATED REVIEW

This section reviews existing research on SSC to determine the present state-of-the-art and any gaps that exist.

Meng et al.[19] conducted an empirical study on Stack Overflow postings to learn about developers' fears about Java secure code, programming challenges, and insecure coding methods. According to the authors, Spring Security's authentication and authorization capabilities have been widely adopted. The problematic cross-language data handling of cryptography APIs and the sophisticated Java-based or XML-based ways to set up Spring Security were discovered to be common programming problems. In addition, this study discovered security flaws in the proposed code of acceptable Stack Overflow solutions. The flaws included removing the default protection against Cross-Site Request Forgery attacks, circumventing certificate validation in SSL/TLS security, and utilizing unsafe cryptographic hash functions.

The findings indicate a lack of safe coding support and documentation and a significant disconnect between security theories and coding methods. Some limitations of this study include manual inspection and only considering java security posts.

Kang and Part[21] suggested an intelligent fog system that can successfully detect/distinguish the vulnerability of the program, in combination with the black box and white box testing, and also proposed a technique to check if it is vulnerable and a means of automatically exploiting it, misdetection was also minimized. To correctly detect security weaknesses, a symbolic-based weakness analysis approach was used. The recommended approach may ensure open-source software dependability by deducting why open-source software is vulnerable to safe and useful invaluable sectors, precluding flaws early in open-source software. In addition, the software developer and the firm employing it are expected to increase their reliability.

Through a review of the paper and the case study studied, Karim et al.[22] highlighted many essential components such as security policies, methods, and technologies utilized throughout the SDLC. The information obtained in the field suggests no defined policies and guidelines in place at the project management level for each step of the SDLC. To evaluate appropriate tasks for inclusion at each step of the SDLC, recommendations, and verification were obtained. A software security framework is presented and verified to incorporate optimal security practices within the SDLC to fulfill the major study goal. The SDLC may be regulated, and security can be included in different phases using the suggested paradigm; as a result, numerous security concerns and problems can be addressed much earlier. This research substantially contributes edge and practices by presenting a model that allows researchers to continue developing and verifying it in diverse circumstances. Managers and developers can also utilize the model early in the development process to create more secure and well-prepared software.

Based on the CERT-C secure coding Standard, Yang et al.[23] presented a novel vulnerability prediction technique. To test the efficacy of the suggested strategy, the prediction outcomes of the recommended prediction models and other traditional models were compared in terms of prediction accuracy and cost-effectiveness. The findings indicate that the proposed strategy can increase the accuracy of vulnerability prediction. In Reference 24, a qualitative assessment of real-world software security practices was presented, which combines software security BPs taken from the literature into a compact list to aid future research in this area. It also considers how effectively current security methods adhere to recommended standards, identifies significant hazards, and investigates why they arise. Through interviews with developers, it was revealed that real-world security procedures varied significantly from those identified in the literature. BPs are frequently disregarded since compliance would add to the team's workload; in their opinion, teams are making an acceptable cost-benefit trade-off. Furthermore, data reveals that the issue extends up the corporate ladder. The findings underscore the need for innovative, lightweight BPs that accommodate development's realities and pressures. Additional automation or reconsideration of secure programming techniques might be part of this.

A hands-on learning module was proposed in Reference 25 to provide step-by-step SSC approaches and tactics to help with the learning process. The suggested module aided students by creating a list of flaws, ranking those flaws based on risk, focusing on specific security vulnerabilities, and testing mitigation methods. Additionally, the eye-trackers method was used to investigate secure coding practices. Students in the proposed eye-tracking research studied SSC learning content, solved multiple choice programming problems, and built mitigation techniques in source code. In addition, the behavior and performance of 29 students were analyzed by utilizing eye-tracking technology to solve security flaws in a web-based application. According to the findings, students who answered an issue poorly spent less time on the material connected with learning about the problem than students who replied adequately.

A readiness model has been used in numerous studies in software engineering research. Niazi et al.[26] utilized it to measure organizational preparedness for software process improvement. There are three layers to their preparedness model: aware, defined, and optimizing. Critical variables and barriers support each level. Case studies in three software companies were used to validate the researchers' readiness model. Similarly, Ali and Khan[27] proposed a strategy for assessing a software company's preparedness to get into outsourcing partnerships. They used crucial partnership characteristics and analyzed their actual execution to build a preparedness model. Contract, success, readiness, conversion, and maturity are all levels in their readiness model. They believe their readiness model may help software development outsourcing by employing case studies from two software companies. Some other studies have also been done on secure coding, such as References 28-30. However, the current work did not use the systematic approach to assist software development organizations in developing secure code.

2.1 | Need for the study

The above discussion shows that SSC is vital as most of the security breaches exploit this phase of SDLC; various practices of SSC have been proposed in the existing literature, but unfortunately, they did not get enough attention. Therefore, there is a need to develop a readiness model for SSC that may assist software development organizations in developing secure code. To fill this gap, this article will provide an SSCRM that will serve the following objectives.

1. Identify the challenges in SSC
2. Identify the BPs that can be helpful in SSC
3. Develop an SSCRM
4. Evaluate SSCRM through case studies

To better comprehend the topic under investigation, this research provides a taxonomy of Software Security Errors (refer to Figure 1) before moving on to the stated objectives. Knowledge of security vulnerabilities can aid in a better understanding of potential security issues and strategies to mitigate them. These software errors are extracted from the research papers on SSDLC. The "others" under the category of input validation & representation include the errors such as resource injection, struts, string termination errors, unsafe native interface, and unsafe reflection and XML validation.[31-37]

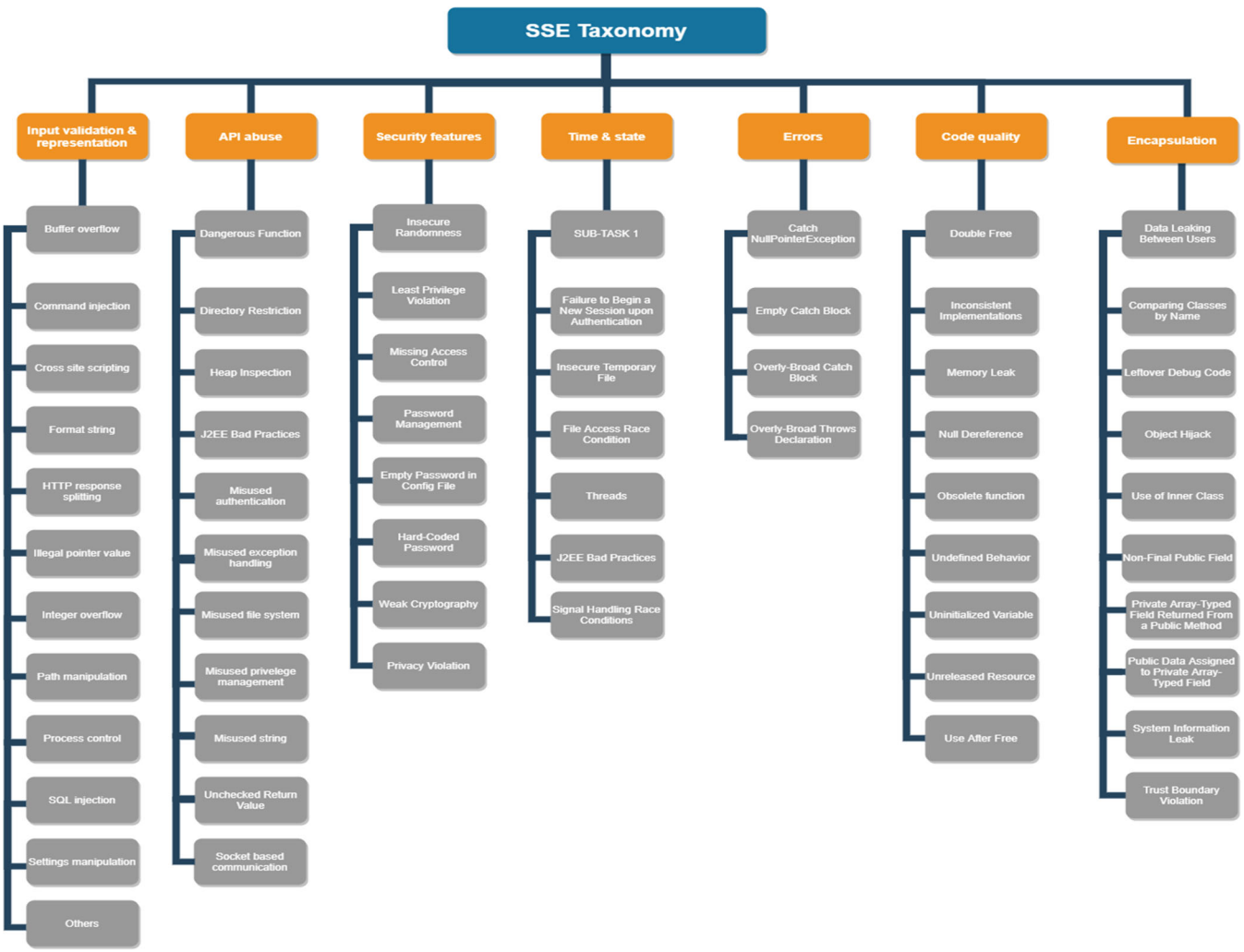

FIGURE 1 Taxonomy of software security errors

2.2 | Secure software coding challenges

SSC is not as simple as it seems to be, various challenges are faced during this process. Some critical challenges involved in SSC are listed below[38-50]:

- Injection,
- Broken authentication and session management,
- Cross-site scripting
- Insecure direct object reference,
- Security misconfiguration,
- Sensitive data exposure,
- Missing function level access control,
- Cross-site request forgery,
- Using components with know vulnerabilities,
- Invalidated redirects and forwards,
- Data validation,
- Authentication,
- Session management,
- Authorization,
- Cryptography,
- Error handling,
- Logging,
- Security configuration,
- Network architecture.

SSC aims to guard the computer software against security vulnerabilities. This process needs a skillful workforce that must consider the following points while coding. These points give a general overview of common practices that all software development organizations must follow to obtain minimum security.

- Follow a risk-based approach where features and components need to be prioritized and sensitive components should be paid more attention,
- Different applications are developed for different purposes; the development team needs to consider the purpose & context of the application under development,
- Different types of risks are associated with different programming languages, and software coding professionals (SCPs) need to be aware of those risks,
- Line of code help to pinpoint the errors in the code, and the more lines of code contain more chances of vulnerabilities,
- Resources, time & deadline need to be considered while writing the code, proper division of resources and utilization of time may help in timely development and code review,
- SCPs should be very clear about users' roles and access rights,
- Sensitive Data need to be protected through cryptography,
- The security features vary for different application types (such as desktop applications, web-based applications, mobile applications, etc.); SCPs must consider the risks associated with each application type,
- Design flaws create problems in coding; therefore, a design review must be performed before coding,
- Standards and guidelines should be followed,
- Data flow needs to be considered while coding,
- SCPs should be aware of entry points that may cause vulnerabilities in the future,

- External dependencies need to be analyzed,
- Strong authentication and access control mechanism,
- Transaction analysis is important to know the related security functions for different transactions.

2.3 | SSC standards, best practices (BPs), and guidelines

It might not be easy to code and implement new software. There are several factors at each stage of the development and approval process. While things might go well, they can also go wrong, and with errors comes the risk of having to start all over again. Different well-known organizations worldwide have proposed different coding standards for various languages. The computer emergency response team (CERT) of the Software engineering institute (SEI) has proposed a Coding Standard for C, C++, Java, Perl, Oracle, and Android OS. Similarly, OWASP also provides SSC BPs to improve software coding quality. The Certified Secure Software Lifecycle Professional (CSSLP) certifies software development and security professionals in implementing BPs in all phases of the SDLC, from software design to implementation to testing and deployment.

Compliance with coding standards is inevitable for software quality, safety, security, reliability, and cost-saving. Table 1 enlists the various practices suggested by SEI, OWASP, and CSSLP.[51-59]

According to Table 1, error handling, memory management, and input validation & data sanitization are the practices that all three well-known standard organizations commonly recommend. Top listed SSC practices of SEI, OWASP, and NTT are given in Table 2

3 | A READINESS MODEL FOR SSC (SSCRM)

A readiness model may be characterized as a strategy for assessing an organization or team's preparedness based on predetermined criteria. This study outlines the problems encountered during SSC and the BPs that can be used. The goal is to create a readiness model with several levels and related practices to determine whether a company is ready for SSC.

The proposed SSRCM model is based on the ideas gathered from RMDevOps,[63] SOVRM,[64] CMMI,[65] RiseRM,[66] and SPIRM.[67] It is divided into different levels, designed based on the knowledge gained through screening grey literature, formal literature, and expert opinion. For grey literature, we used the Google search engine. We applied the search string "secure software coding challenges and best practices" and selected relevant grey literature. In formal literature, we studied formal research papers; we collected SSC challenges and BPs from these papers. Most of the practices mentioned in formal literature also belong to the above three categories mentioned in Table 2, namely OWASP, SEI, and NTT. As far as the expert opinion is concerned, it was used for mapping between SSC practices with corresponding levels. The details of the mapping process are given in the explanation of the SSCRM development process.

The detailed process of the SSRCM formulation is given in Figure 2

The proposed SSCRM model development consists of four stages. The first stage of SSCRM was information gathering; the sources used include grey literature and formal literature. The purpose of using both formal and grey literature was to evaluate whether researchers and practitioners were moving at the same pace or if there were any disagreements. For grey literature, we used the Google search engine and retrieved the material related to SSC challenges and BPs. We got millions of results when we used the phrase "secure software coding challenges and best practices" in the Google search engine. We only chose papers from the first six pages since the studies beyond that either repeated themselves or were irrelevant. For formal literature, we retrieved the studies on SSC and identified SSC challenges and BPs from these studies.

During stage 2, we did the screening of grey and formal literature to identify the possible SSC challenges. During this stage of the screening process, the abstracts and conclusions of the retrieved research papers were examined to verify that the chosen research discusses the difficulties associated with SSC and offers recommendations for how these difficulties might be overcome. In addition, the references of studies that were more pertinent were examined to guarantee that no one of the research that was closely linked was overlooked. In stage 3, we identified the BPs and guidelines for SSC after

TABLE 1 SSC practices

Practices	SEI Perl	SEI C++	SEI C	SEI Java	SEI Android	OWASP	CSSLP
Component security					♦		
File I/O and logging	♦	♦	♦	♦	♦	♦	
Network–SSL/TLS					♦		
Permission/authentication & password management					♦	♦	
Cryptography					♦	♦	
Declarations and initialization	♦	♦		♦	♦		
Application programming interfaces					♦		♦
Environment			♦	♦	♦		
Error handling		♦	♦	♦	♦	♦	♦
Memory management		♦	♦		♦	♦	♦
Concurrency		♦	♦		♦		♦
Input validation and data sanitization	♦			♦	♦	♦	♦
Locking				♦	♦		
Serialization				♦	♦		
Thread APIs/safe APIs				♦	♦		
Thread pools				♦			
Thread-safety miscellaneous				♦	♦		
Visibility and atomicity				♦	♦		
Platform security				♦	♦		
Exceptional behavior		♦		♦			♦
Output encoding					♦		
Session management/exception management					♦		♦
Access control					♦		
Data protection					♦		
Communication security					♦		
System configuration					♦		
Database security					♦		
Canonicalization							♦
Secure startup							♦
Sandboxing							♦
Anti-tampering							♦

reading the selected studies in detail. In the last stage, we mapped the identified challenges with BPs and guidelines for identifying the various levels of SSCRM and associated practices.

Other than formal and grey literature, we used expert opinion during stage 4 to correctly map identified SSC best practices with the proposed readiness model levels. The mapping process consists of three steps: in step 1, we studied existing SSC challenges and BPs from relevant grey and formal literature and shortlisted 24 BPs. In stage 2, based on the authors' understanding and concepts gained from the literature review, all the investigated best practices were mapped against the readiness level. In stage 3, expert opinion is used to validate the mapping between BPs and SSCRM levels. For an expert opinion, two experts were chosen, one from academia and the other from the software industry. Expert 1 was a full professor in a world-ranked university with a major in security and more than

TABLE 2 Top SSC practices suggested by SEI, OWASP, and NTT

SEI[60]	OWASP[61]	NTT[62]
Validate input	Input validation	Develop an SSC standard
Heed compiler warnings	Output encoding	Validate input from all external data sources
Architect and design for security policies	Authentication & password management	Deny access by default
Keep it simple	Session management	Enforce security policies
Default deny	Access control	Use the compiler's highest warning level
Adhere to the principle of least privilege	Cryptographic practices	Layered defense
Sanitize data sent to other systems	Error handling & logging	Error handling and logging
Practice defense in depth	Data protection	Threat modeling
Use effective quality assurance techniques	Communication security	
Adopt an SSC standard	System configuration	
	Database security	
	File management	
	Memory management	

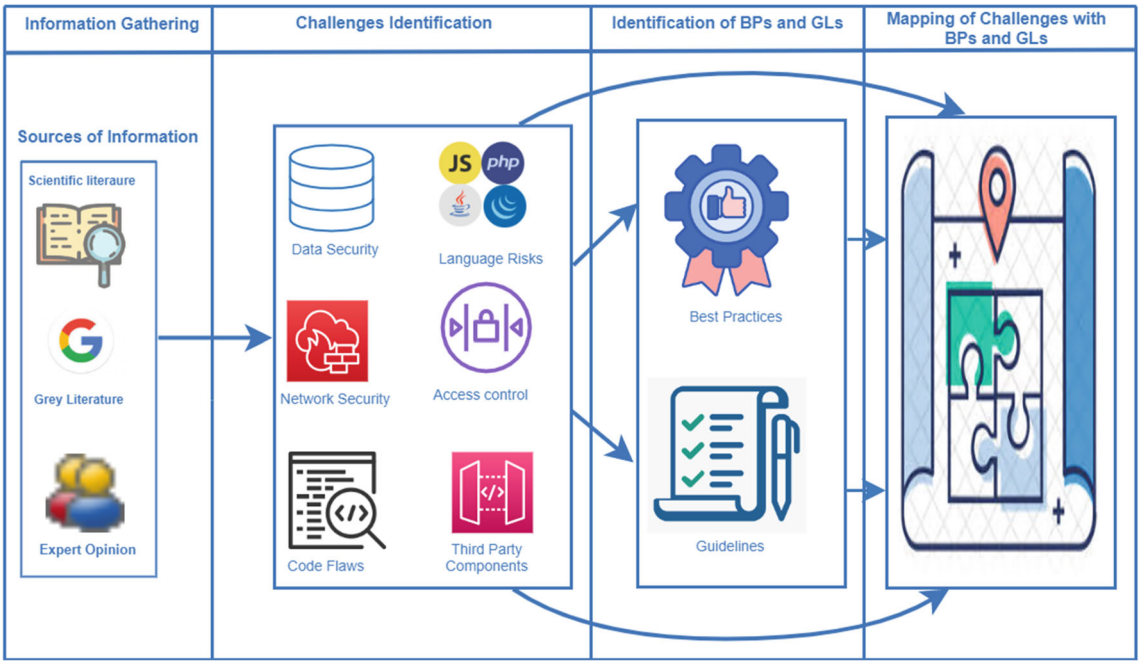

FIGURE 2 SSRCM development Process

20 years of teaching and research experience in software security. Expert 2 was a practitioner from the software industry with more than ten years of experience in software quality and risk assessment. These experts were requested to assess the proposed mapping. The specialists agreed with most of the authors' mapping; however, they asked for a few adjustments. After consulting with an expert, the final mapping of BPs to SSCRM levels was completed, as illustrated in Figure 3.

Figure 3 shows the proposed SSRCM Model; this model has five levels. SSC challenges and BPs are associated with each level. To achieve the next level, an organization needs to overcome the challenges associated with that layer by following the SSC BPs. Below we explain these five levels briefly

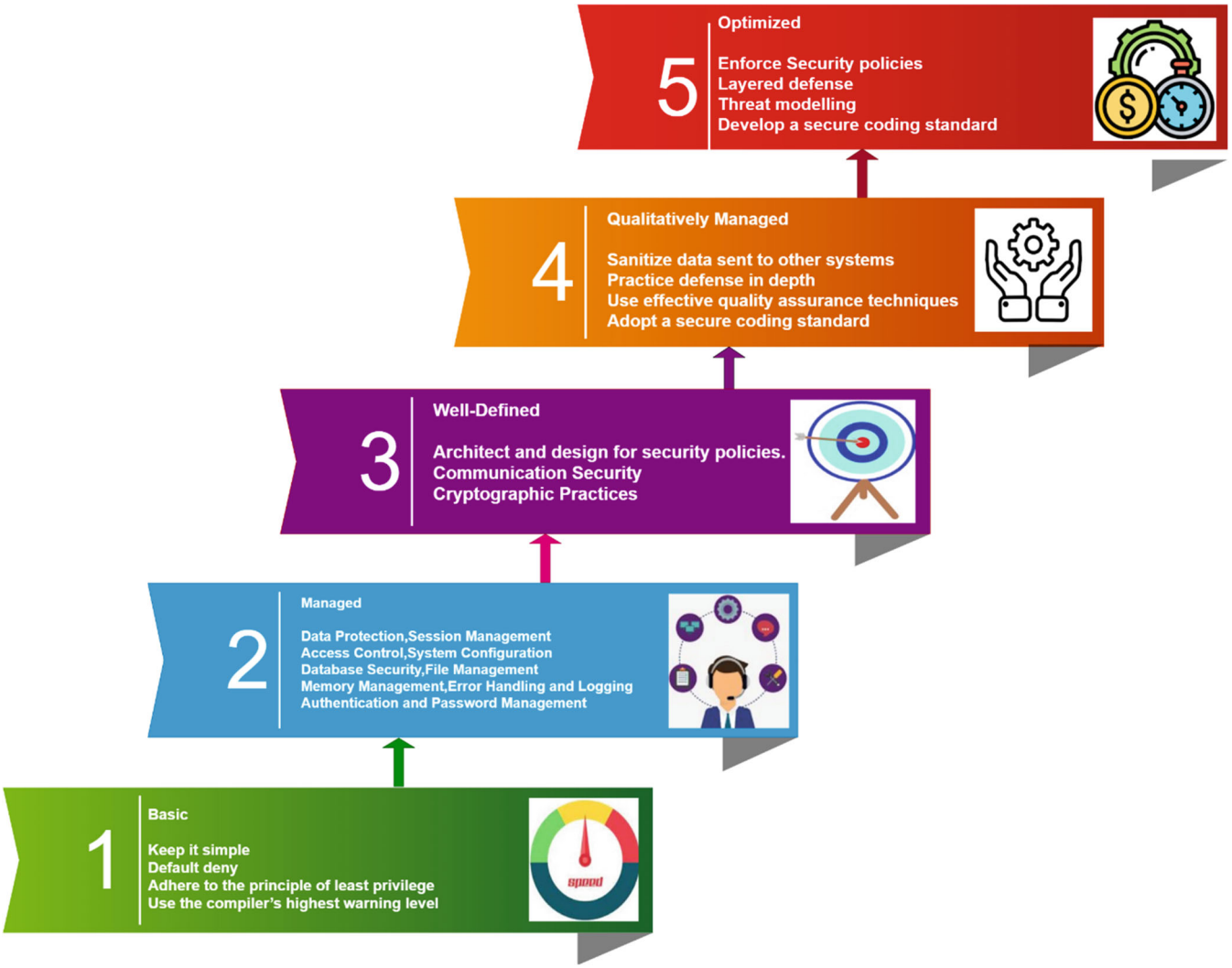

FIGURE 3 SSCRM model

3.1 | Basic

There are no hard practices that organizations need to follow at this level, and they follow the practices gained through past experiences. Some basic strategies that need to be adopted at this level include keeping the design and code as simple as possible, adopting the default deny and least privileges policy, and using the compiler's highest warning level. Software organizations at this level should follow the practices mentioned earlier to move the next level.

3.2 | Managed

At this level, organizations need to use several new practices through knowledge gained from the previous level and increased awareness. Organizations at this level should be able to save their assets such as data and applications from security breaches through session management, access control and logging, file management, memory management, proper error handling and logging and so forth. By following these practices, organizations get further maturity and strive for improvements.

3.3 | Well-defined

The key concern at this level is to plan and follow standardized security policies and practices. Organizations need to design their security policy based on international standards. The data must be protected during communication

through communication security and following standard cryptography practices. All risks associated with programming languages, application types, etc., need to be identified, and proper risk analysis and mitigation strategies must be developed.

3.4 | Qualitatively managed

At this level, the organizations integrate the standardized BPs with their current SDLC. The focus at this level is to achieve the set quality goals. The organizations at this level need to apply defense-in-depth strategy, use of code standards, effective quality assurance techniques, and proper sanitization of data before sending.

3.5 | Optimized

Based on customer/developer feedback and strategic ideas, the organization will investigate and assess novel ways to increase code security at this level. To go one step forward, predictable code restructuring, security capabilities, and high management techniques are used to meet business objectives, and consumer wants in a short period.

4 | SSCRM EVALUATION USING CASE STUDY

A case study is a powerful assessment method that offers sufficient context.[68] Because SSCRM was built for the software business, a case study was used in this study. We evaluated SSCRM using three industrial case studies. The primary goals of this research's case studies are to demonstrate the viability of employing SSCRM in real-world situations.

To perform the case study, we contacted software practitioners from several software companies, explained SSCRM, and offered them to participate. They were requested to analyze their software coding security procedures using the Motorola assessment instrument[69] and SSCRM. They completed the evaluation at work and sent the findings and comments.

To reduce the danger of business size, we focused our case study on one large, medium, and small firms. Organization A is a large global software development firm with over 70 k employees based in Saudi Arabia. It has many clients and offers a variety of IT solutions and services. Organization B is a medium-sized Saudi Arabian company with about 2000 workers that provides various safety solutions and services. Organization C is a modest organization with around 20 individuals that offers community services to the residents in the area.

4.1 | Assessment criteria

As seen in Figure 3, the proposed SSCRM model comprises five readiness levels, each of which includes a set of activities. The respondents from each participating organization were asked to quantify each practice to determine how it was implemented in their respective businesses. Motorola Metrix, which has the following three dimensions, was suggested for measurement.

1. Approach: the organization's commitment and management support for the practice and its capacity to apply the practice are considered here.
2. Deployment: the breadth and consistency of practice application across project domains are critical criteria here.
3. Outcomes: the breadth and consistency of good results over time and across project areas are considered here.

Each practice is assigned a number between 0 and 10, which may be adjusted depending on the organization's preparation.

4.2 | Assessment results for organization A

For organization A, Figure 4 displays the evaluation findings for each practice. The following are the major points discovered in this evaluation using the Motorola assessment tool's guidelines; the assessment details are given in Appendix A.

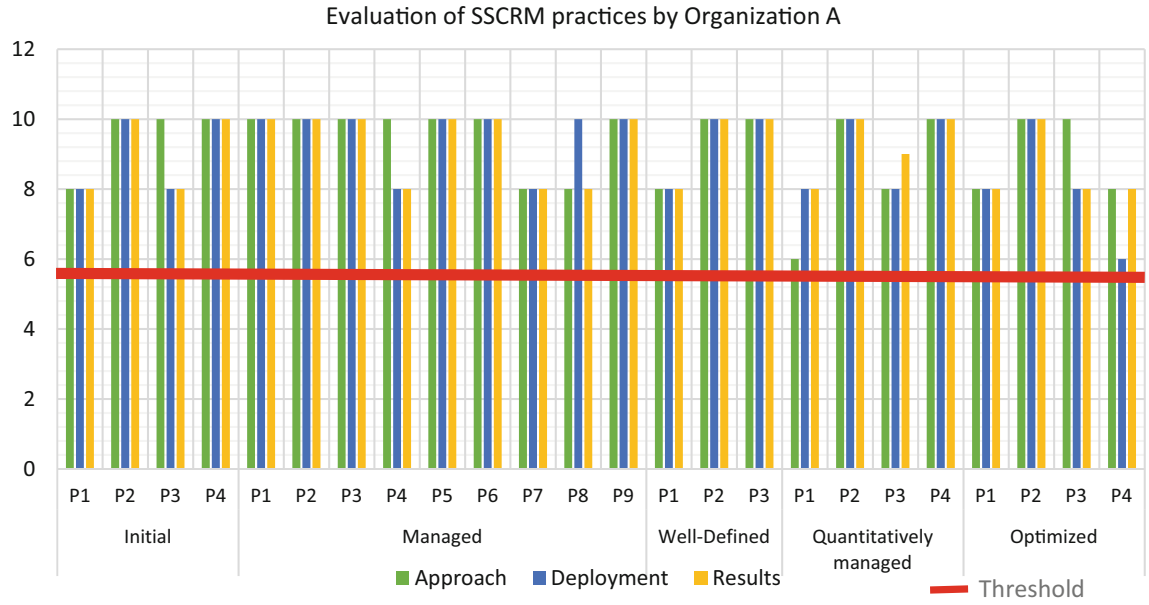

FIGURE 4 Evaluation of SSCRM levels and corresponding practices by organization A

1. The organization is strong in SSCRM Levels 1, 2, and 3 since it received a score of at least 7 in each of these practices.
2. The company complies with all quantitively managed practices, except for sanitization of input and output, which has a score value of less than 7.
3. The company meets all of the Optimized level practices, except for adopting a secure coding standard with a score value of less than 7.

The respondents of the organization were also asked to evaluate the readiness model using the criteria mentioned in Table 3. The three parameters used for the evaluation of SSCRM were Ease of learning, User Satisfaction, and SSCRM

TABLE 3 Criteria for SSCRM evaluation

Criteria	Evaluation parameter 1: Ease of learning
C1	It is easy to understand the practices designed for each level of the Secure Software Coding Readiness Model (SSCRM)
C2	It is easy to understand the assessment method used in the SSCRM
C3	It is easy to use SSCRM to assess organizations' readiness for secure coding
C4	Each practice is easy to understand and unambiguous
C5	Some training needs to be provided for the use of SSCRM
	Evaluation parameter 2: User satisfaction
C1	SSCRM is general and can be applied to most companies.
C2	Using SSCRM would identify strong and weak areas in the company regarding secure software coding
C3	I am satisfied and agreed with the readiness issues identified by the SSCRM
C4	Using SSCRM would improve our secure software coding processes.
	Evaluation parameter 3: The structure of SSCRM
C1	All the levels of the SSCRM are self-explanatory and require no further explanation to be used effectively
C2	The levels of SSCRM are practical and are applicable in the software industry
C3	The distribution of practices among different readiness levels (e.g., managed, well-defined) is useful

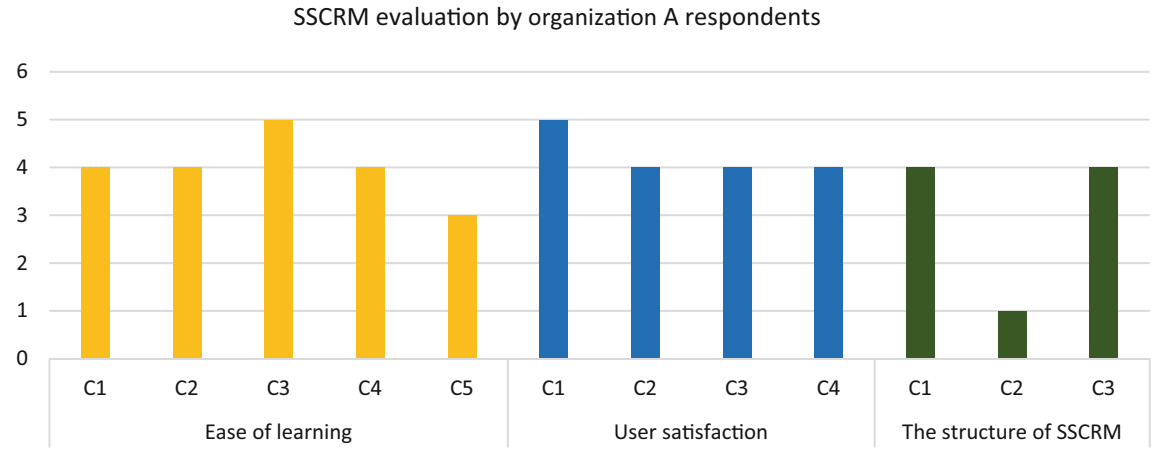

FIGURE 5 SSCRM evaluation by the respondents of organization A

structure. The SSCRM evaluation for each question was done on a Likert scale of 5, ranging from strongly agree to strongly disagree.

The Respondents of organization A evaluated the SSCRM model based on the criteria mentioned in Table 3, the 5 Likert scales were assigned quantitative weights in such a way that strongly agree = 5, agree = 4, neutral = 3, disagree = 2, and strongly disagree = 1. The evaluation results are presented in Figure 5

4.2.1 | Strengths and weaknesses of SSCRM as mentioned by organization A

The respondents were asked to provide their opinion about the proposed SSCRM, its strengths, weaknesses, and suggestions for improvement; below is the detail.

The general opinion of organization A about SSCRM

The technique is meant to touch the most sensitive regions of code, where most typical vulnerabilities occur. The suggested approach can have a significant influence on the examined organization's everyday operations, notably assisting them in avoiding risk beyond acceptable limits.

Strengths of SSCRM as mentioned by organization A

- The proposed model is Measurable and business-oriented,
- Process clarity and simplicity of implementation,
- Emphasize the most important aspects and flaws in code development processes,
- It's cost-effective and efficient (in terms of time and money),
- Defining and recommending a new approach to conducting software development security assessments in a company,
- It has a broad range of applications since it can be used by a variety of companies and business lines,
- Instantly provides comments based on the total score.

Weaknesses of SSCRM as mentioned by organization A

- Based on words (high chance to make mistakes, especially in calculations),
- The evaluation is based on subject matter expert point of views' therefore, it is subjective,
- A deeper understanding of SDLC is required.

Suggestion for improvement of SSCRM as mentioned by organization A

SSCRM is regarded as a simple and reliable model for assessing code readiness; nonetheless, the ideas below may help to enhance SSCRM's quality.

- Converting this document to a web-based form will enable this model to be implemented across various entities and persons inside the targeted company. This improvement aims to prevent basing the final judgment on a single person's opinion, which might be prejudiced in certain circumstances.
- Another change that might enhance the model's dependability is to examine each score separately from the average. This is because an organization may get a score of 7 or higher and yet fail a level if one of the important practices is poorly executed and receives a 0.
- Finally, putting this model in a word document may lead to calculation problems. Converting those tables to excel sheets with built-in algorithms to average the scores and indicate the organization's SSCRM level is one way to improve that aspect of the report.

4.3 | Assessment results for organization B

For organization B, Figure 6 displays the evaluation findings for each proposed model practice. The following are the major points discovered in this evaluation using the Motorola assessment tool's guidelines; the assessment details are given in Appendix B.

1. Company B partially accomplishes some of the SSCRM levels since only a few practices are completely implemented, and their values are above the threshold value. In contrast, certain SSCRM practices still need more attention,
2. The average point value for each practice across all five levels is more than the threshold, indicating that the organization is improving.

The respondents of organization B also evaluated the SSCRM model based on the criteria mentioned in Table 3. The evaluation results are presented in Figure 7; the measurement scale was the same as for organization A. The results of Figure 7 show that the proposed model provides ease of learning, user satisfaction, and a good structure as almost all the responses were either strongly agree or agree.

4.3.1 | Strengths and weaknesses of SSCRM as mentioned by organization B

The respondents of organization B were also asked to provide their opinion about the proposed SSCRM, its strengths, weaknesses, and suggestions for improvement; below are the details

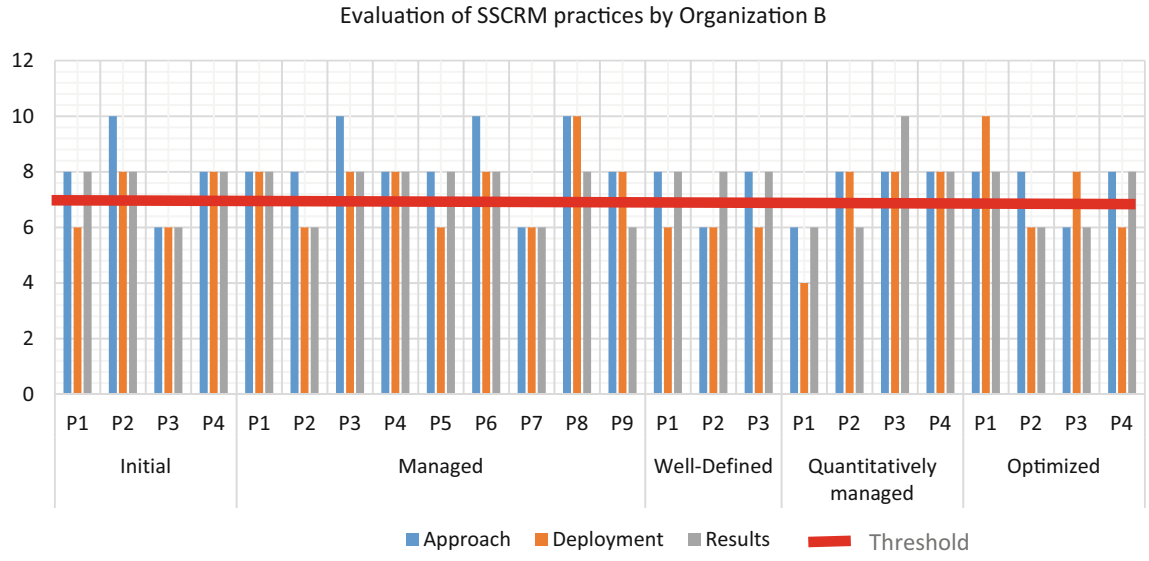

FIGURE 6 Evaluation of SSCRM levels and corresponding practices by organization A

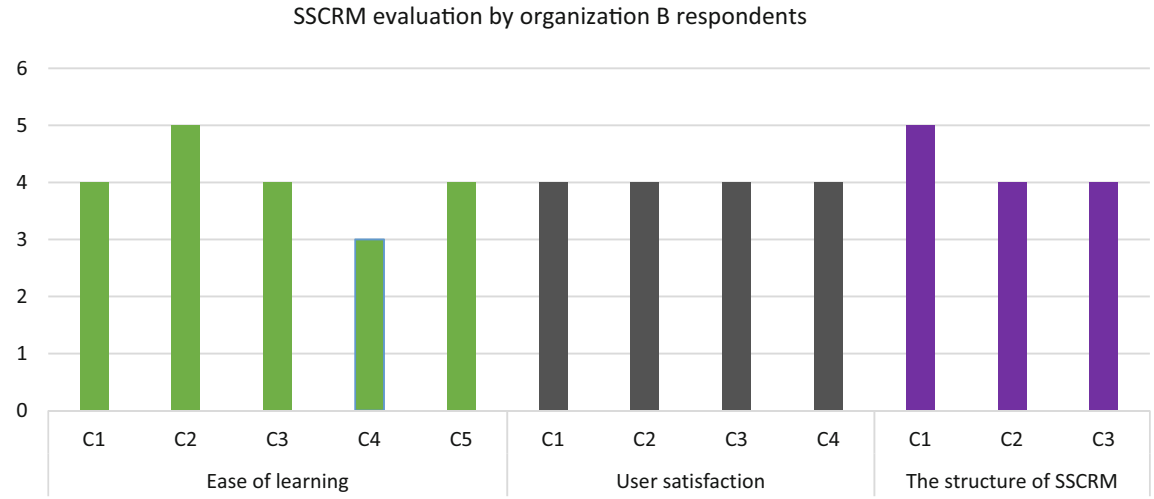

FIGURE 7 SSCRM evaluation by the respondents of organization A

The general opinion of organization B about SSCRM

In order to improve software code and eliminate vulnerabilities, the suggested SSCRM paradigm is both simple and comprehensive.

Strengths of SSCRM as mentioned by organization B

- All businesses and domains may benefit from using the SSCRM, which is a complete approach to securing software development.
- To ensure that secure software development practices are followed repeatedly and consistently, SSCRM may serve as an elicit guideline for the enterprise.
- In order to create a need for risk analysis and risk treatment strategy, it shows the lack of secure software needs.
- It may assist businesses by providing a measurable figure and ranking that makes it simple to compare one's position to others.

Weaknesses of SSCRM as mentioned by organization B

- Organization maturity can only be assessed by security specialists, yet most firms lack security software expertise or consultants.
- There are many phrases and procedures that are unfamiliar to most people, making it difficult to adopt without previous expertise in the security area or without sufficient training or instructions.
- A human error or incorrect data input might occur since it is a manual computation.

Suggestion for improvement of SSCRM as mentioned by organization B

- In order to make this evaluation easier to complete, we recommend grouping these activities into roles and duties and explaining them clearly from the start. If the senior management is in charge of maintaining policy and procedure, then they should be evaluating "Approach," however, the "Deployment" review should be left to the technical teams.
- We recommend that this evaluation be conducted electronically, with the system doing all of the computations rather than the evaluator. As a result, mistakes in computation and human error may be reduced, making this evaluation more accurate and less time-consuming.
- As few questions are asked in multiple ways to match the information, it is desirable that the assessment include some checking and validation methods.

4.4 | Assessment results for organization C

Figure 8 displays the evaluation findings for each practice of the proposed model by organization C, a small organization. The following are the major points discovered in this evaluation using the Motorola assessment tool's guidelines, the details of the assessment details are given in Appendix C.

1. Company B completes several SSCRM procedures entirely, but most practices at each level are still not met since the accumulative value for each practice is less than the threshold value.
2. Only levels 2 and 3 have accumulative values that are more than the threshold values, while the other three levels have accumulative values that are less than the threshold values.

The Respondents of organization C also evaluated the SSCRM model based on the criteria mentioned in Table 3. The evaluation results are presented in Figure 9, the scale of measurement was the same as for the other two participating organizations.

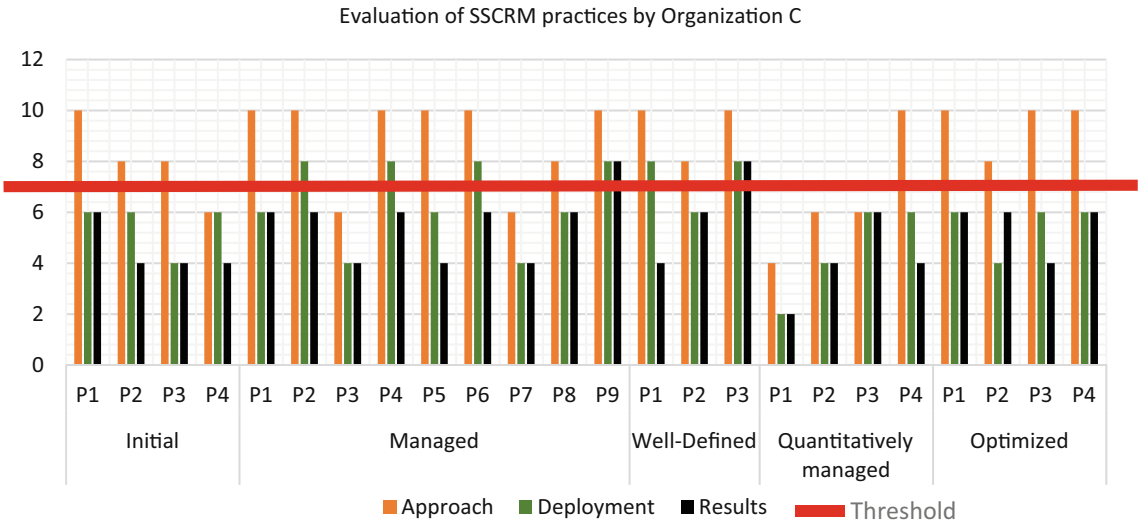

FIGURE 8 Evaluation of SSCRM levels and corresponding practices by organization A

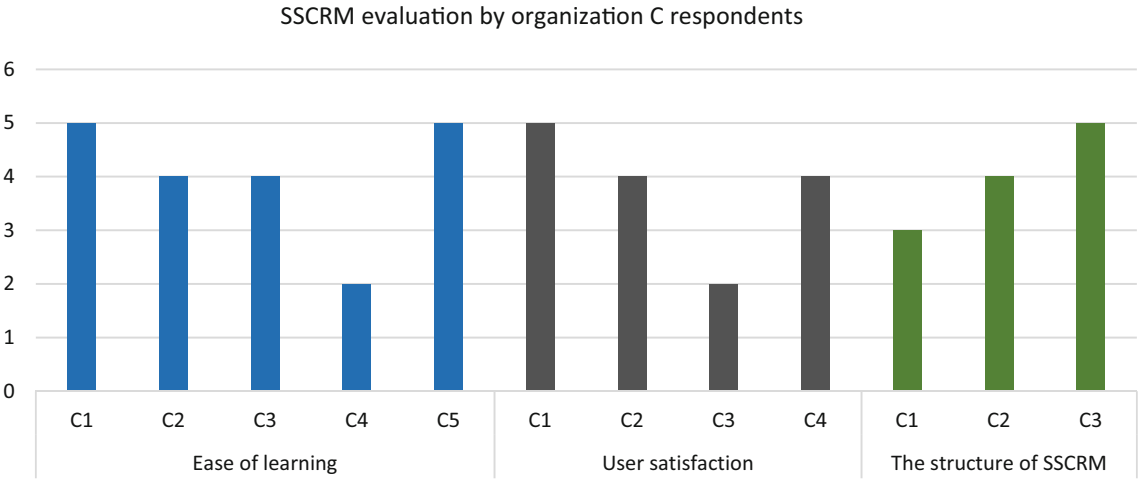

FIGURE 9 SSCRM evaluation by the respondents of organization A

4.4.1 | Strengths and weaknesses of SSCRM as mentioned by organization C

The respondents of organization C were also asked to provide their opinion about the proposed SSCRM, its strengths, weaknesses, and suggestions for improvement, the details of evaluation is provided in the following sub-sections

Strengths of SSCRM as mentioned by organization C
Organization C cited the following SSCRM set of needed standards and rules as a strength.

- Simple and straightforward to comprehend and assess.
- Expands on solid cybersecurity principles such as; input sanitization; least privileges; use cryptography; default deny, and defense-in-depth.
- Broad and applicable to most businesses.

Weaknesses of SSCRM as mentioned by organization C
Organization C highlighted the following flaws in the proposed SSCRM

- It does not concentrate on code purpose, but rather on high-level measures.
- It does not specify the nature of the problem.
- It does not give direction for a solution.
- Noncompliance with local regulations and standards such as NCA ECC, ISO27001, and NIST.
- SSCRM does not offer any privacy recommendations or metrics.

Suggestion for improvement of SSCRM as mentioned by organization C
SSCRM may improve in the following ways:

- Allow the assessment to take into consideration the software's purpose.
- Comply with local laws and international standards.
- Initiate and assess privacy standards.
- Finally, what action should be done in the event of a failure?

5 | DISCUSSION AND IMPLICATIONS

This research article aims to develop a readiness model for SSC, namely SSCRM. The study results provide a roadmap to the organization for securing code from attacks. The organizations can improve their maturity with time by mitigating the SSC challenges and adopting SSC BPs. The following objectives were achieved in this article:

- Identifying the issues involved in SSC through screening of existing literature on SSC,
- Identifying the BPs that can be helpful for SSC,
- Develop a layered model named SSCRM for indicating SSC maturity levels,
- Evaluation of SSCRM through Case studies.

The findings of three case studies suggest that the proposed SSCRM is beneficial in evaluating coding security. The results, however, reveal that big firms use secure coding methods more than small and medium-sized organizations. Furthermore, senior management and security experts inside the firm may play a role in reaching the different levels of SSCRM.

Organizations A and B suggested using web-based forms to avoid human error. In the future work, we plan to develop a simple web-based application with a database at the backend. This application will not only help collect and store the data,

but it will also provide some other benefits, including; saving respondents' profiles for future use, avoiding incomplete or incorrect data, and reducing human error.

Based on the state-of-the-art literature, this study revealed numerous aspects and practices. The research investigated the elements that have a detrimental impact on SSC. The explicit contribution of the study is as follows.

- The study gives industrial coding specialists and university researchers a body of information to focus on the most critical SSC problems while building software.
- The problems examined aid specialists in developing methods and solutions to solve code-related concerns, which are essential for creating and evolving critical applications in the software industry.
- Furthermore, by mapping all elements and practices to each model level, the SSCRM model was developed based on other existing models. By tackling the SSC issues with BPs, this approach will assist practitioners in improving their organization's standards and processes.
- The case study evaluations further assist academics and practitioners by providing a flexible way to assess their security level.

5.1 | Study limitations

Some limitations of this study include that we attempted to cover all relevant literature. However, because it was not an SLR, other materials may have been overlooked. Studies that have been published since this research project began may be overlooked. However, we believe that our findings are comprehensive and cover all relevant literature. Concerning the threat to the mapping of BPs with SSCRM challenges, we validated it by consulting experts and made changes according to their suggestions before the final SSCRM development.

6 | CONCLUSION AND FUTURE WORK

There are several reasons why a software developer may have difficulty implementing security. To begin with, most developers are preoccupied with the application's release and overlook a critical component of security. Similarly, some developers may miss that the program is vulnerable to cyber-attack and fail to include any safeguards to protect the software. SSC techniques identify and eliminate flaws in the final code that cyber attackers might exploit in such a situation. Cyber attackers will find it harder to break secure code and access applications and systems, resulting in fewer data breaches. We created a readiness model for the successful execution of SSC operations because of the relevance of SSC in the software sector. We presented the SSRCM readiness model in this study. Organizations may overcome the obstacles they experience during SSC by using SSRCM. The knowledge obtained from grey literature, formal literature, and expert opinion was used to structure the SSCRM. The suggested SSRCM model is divided into five layers, each comprising different BPs. The identified practices and challenges were mapped with different levels of the model.

To assess our model, we used three case studies. The results of the case studies show that the proposed model helps evaluate organizations' secure coding levels. The proposed SSCRM will contribute in many ways; Software developers and academics may use the model to identify the most pressing SSC issues while working on new software. Experts may use the challenges reviewed to create approaches and solutions for solving code-related issues, which are vital for the development and evolution of key software programs. As a result, practitioners will be able to improve their organization's standards and procedures by using this strategy. Researchers and practitioners alike will benefit from the recommended approach since it provides a flexible method to measure their security level. The organization may use the suggested assessment to analyze the present degree of security and the influence on the overall security of applying a particular practice.

In the future, we plan to refine the model in light of suggestions provided by the respondents of organizations involved in conducting case studies. We will also add automation and more intelligent components to the proposed model to make it more usable.

ACKNOWLEDGMENT

The authors would like to acknowledge the support provided by The Deanship of Research Oversight and Coordination via Project number DF201007 at King Fahd University of Petroleum and Minerals, Saudi Arabia.

AUTHOR CONTRIBUTIONS

Mamoona Humayun: Conceptualization; data curation; formal analysis; methodology; resources; writing - original draft. **Mahmood Niazi:** Conceptualization; formal analysis; investigation; methodology; resources. **NZ Jhanjhi:** Investigation; project administration. **Sajjad Mahmood:** Formal analysis; project administration. **Mohammad Alshayeb:** Formal analysis; project administration.

DATA AVAILABILITY STATEMENT

Will be furnished on request.

ORCID

Mamoona Humayun https://orcid.org/0000-0001-6339-2257
Sajjad Mahmood https://orcid.org/0000-0001-5786-5118
Mohammad Alshayeb https://orcid.org/0000-0001-7950-0099

REFERENCES

1. Fernandez EB. A methodology for secure software design. *Software Engineering Research and Practice*. Citeseer; 2004.
2. Jayaram K, Mathur AP. *Software engineering for secure software-state of the art: A survey*. Department of Computer Science; 2005.
3. Siavvas M, Gelenbe E, Kehagias D, Tzovaras D. Static analysis-based approaches for secure software development. *International ISCIS Security Workshop*. Springer, Cham; 2018.
4. Sugiantoro B, Anshari M, Sudrajat D. Developing framework for web based e-commerce: secure-SDLC. *J Phys Conf Series*. 2020;1566: 012020.
5. Fernandes AM, Pai A, Colaco LMM. Secure SDLC for IoT based health monitor. *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)* , May 4-9, 2019, Glasgow, Scotland, UK, IEEE; 2018.
6. Assal H, Chiasson S. "Think secure from the beginning" A Survey with Software Developers. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*:2019.
7. Khan RA, Khan SU, Khan HU, Ilyas M. Systematic mapping study on security approaches in secure software engineering. *IEEE Access*. 2021;9:19139-19160.
8. Khan RA, Khan SU, Ilyas M, Idris MY. The state of the art on secure software engineering: a systematic mapping study. EASE '20: Proceedings of the Evaluation and Assessment in Software Engineering April 2020;487-492.
9. Farhan AS, Mostafa GM. A methodology for enhancing software security during development processes. *2018 21st Saudi Computer Society National Computer Conference (NCC)*. IEEE; 2018.
10. Dodson D, Souppaya M, Scarfone K. *Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework (SSDF) (Draft)*. National Institute of Standards and Technology; 2019.
11. Kang S, Kim S. CIA-Level Driven Secure SDLC Framework for Integrating Security into SDLC Process. *J Korea Ins Informat Security Cryptol*. 2020;30(5):909-928.
12. Al-Matouq H, Mahmood S, Alshayeb M, Niazi M. A Maturity Model for Secure Software Design: A Multivocal Study. *IEEE Access*. 2020;8:215758-215776.
13. Eian IC, Yong LK, Li MYX, Hasmaddi NABN, Zahra F-t. Integration of Security Modules in Software Development Lifecycle Phases. arXiv preprint, arXiv:2012.05540. 2020.
14. Jhanjhi N, Humayun M, Almuayqil SN. Cyber Security and Privacy Issues in Industrial Internet of Things. *Comput Syst Sci Eng*. 2021;37(3):361-380.
15. Kurachi R, Takada H, Tanabe M, et al. Improving secure coding rules for automotive software by using a vulnerability database. *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. IEEE; 2018.
16. Jones RL, Rastogi A. Secure coding: building security into the software development life cycle. *Inf Secur J A Glob Perspect*. 2004;13(5):29-39.
17. Humayun M, Niazi M, Jhanjhi NZ, Alshayeb M, Mahmood S. Cyber security threats and vulnerabilities: a systematic mapping study. *Arabian J Sci Eng*. 2020;45(4):3171-3189.
18. Antunes N, Vieira M. Defending against web application vulnerabilities. *Computer*. 2012;45(02):66-72.
19. Meng N, Nagy S, Yao D, Zhuang W, Arango-Argoty G. Secure coding practices in java: Challenges and vulnerabilities. ICSE '18: Proceedings of the 40th International Conference on Software Engineering May ACM; 2018:372-383.
20. Almrezeq N. Cyber Security Attacks and Challenges in Saudi Arabia during COVID-19. *Turkish J Comput Math Educat*. 2021;12(10):2982-2991.
21. Kang J, Park JH. A secure-coding and vulnerability check system based on smart-fuzzing and exploit. *Neurocomputing*. 2017;256:23-34.

22. Karim NSA, Albuolayan A, Saba T, Rehman A. The practice of secure software development in SDLC: an investigation through existing model and a case study. *Security Commun Networks*. 2016;9(18):5333-5345.
23. Yang J, Ryu D, Baik J. Improving vulnerability prediction accuracy with secure coding standard violation measures. *2016 International Conference on Big Data and Smart Computing (BigComp)*. IEEE; 2016.
24. Assal H, Chiasson S. Security in the software development lifecycle. *Fourteenth Symposium on Usable Privacy and Security (SOUPS)*; Usenix Association; 2018.
25. Davis D, Zhu F. Understanding and improving secure coding behavior with eye tracking methodologies. ACM SE '20: Proceedings of the 2020 ACM Southeast Conference April 2020;107-114: ACM.
26. Keshta IM, Niazi M, Alshayeb M. Towards the implementation of requirements management specific practices (SP 1.1 and SP 1.2) for small-and medium-sized software development organisations. *IET Software*. 2020;14(3):308-317.
27. Ali S, Khan SU. Software outsourcing partnership model: An evaluation framework for vendor organizations. *J Syst Software*. 2016;117:402-425.
28. Andrade R, Borba P. Privacy and security constraints for code contributions. *Software Practice Exp*. 2020;50(10):1905-1929.
29. Akram J, Luo P. SQVDT: A scalable quantitative vulnerability detection technique for source code security assessment. *Software Practice Exp*. 2021;51(2):294-318.
30. Baca D, Carlsson B, Petersen K, Lundberg L. Improving software security with static automated code analysis in an industry setting. *Software Practice Exp*. 2013;43(3):259-279.
31. Huang Z. *Fixing Software Vulnerabilities and Configuration Errors*. University of Toronto; 2018.
32. Acar Y, Stransky C, Wermke D, Weir C, Mazurek ML, Fahl S. Developers need support, too: A survey of security advice for software developers. *2017 IEEE Cybersecurity Development (SecDev)*. IEEE; 2017.
33. Howard M, LeBlanc D, Viega J. *24 deadly sins of software security: Programming flaws and how to fix them*. McGraw-Hill Education; 2010.
34. Zhivich M, Cunningham RK. The real cost of software errors. *IEEE Secur Privacy*. NIST;2009;7(2):87-90.
35. Du W, Mathur AP. Categorization of software errors that led to security breaches. *21st National Information Systems Security Conference*; 1998.
36. Xie J, Lipford HR, Chu B. Why do programmers make security errors? *2011 IEEE symposium on visual languages and human-centric computing (VL/HCC)*. IEEE; 2011.
37. Tsipenyuk K, Chess B, McGraw G. Seven pernicious kingdoms: A taxonomy of software security errors. *IEEE Secur Privacy*. 2005;3(6):81-84.
38. Vargas S, Vera M, Rodriguez J. Security strategy for vulnerabilities prevention in the development of web applications. *Journal of Physics: Conference Series*. IOP Publishing; 2019;1414:012017.
39. Fredj OB, Cheikhrouhou O, Krichen M, Hamam H, Derhab A. An OWASP top ten driven survey on web application protection methods. *International Conference on Risks and Security of Internet and Systems*. Springer; 2020.
40. Shahidullah M. *Vulnerability Assessment Penetration Testing for Web Application*; EasyChair preprint; 2019.
41. Shivakumar SK, Sethii S. DXP Security. *Building Digital Experience Platforms*. Springer; 2019:183-200.
42. Appiah V, Nti IK, Nyarko-Boateng O. Investigating websites and web application vulnerabilities: Webmaster's perspective. *Int J Appl Informat Syst*. 2017;12(3):10-15.
43. Touseef P, Alam KA, Jamil A, et al. Analysis of automated web application security vulnerabilities testing. *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*; 2019: ACM.
44. Ayachi Y, Ettifouri EH, Berrich J, Toumi B. Modeling the owasp most critical web attacks. *International Conference Europe Middle East & North Africa Information Systems and Technologies to Support Learning*. Springer; 2018.
45. Sulatycki R, Fernandez EB. Two threat patterns that exploit "security misconfiguration" and "sensitive data exposure" vulnerabilities. *Proceedings of the 20th European Conference on Pattern Languages of Programs*; 2015: ACM.
46. KumarShrestha A, Singh Maharjan P, Paudel S. Identification and illustration of insecure direct object references and their countermeasures. *Int J Comput Appl*. 2015;114(18):39-44.
47. Rodríguez GE, Torres JG, Flores P, Benavides DE. Cross-site scripting (XSS) attacks and mitigation: A survey. *Comput Networks*. 2020;166:106960.
48. Hydara I, Md. Sultan AB, Zulzalil H, Admodisastro N. Current state of research on cross-site scripting (XSS)–A systematic literature review. *Informat Software Technol*. 2015;58:170-186.
49. Hassan MM, Nipa SS, Akter M. Broken authentication and session management vulnerability: a case study of web application. *Int J Simulat Syst Sci Technol*. 2018;19(2):6.1-6.11.
50. Ali NS, Shibghatullah A. Protection web applications using real-time technique to detect structured query language injection attacks. *Int J Comput Appl*. 2016;149(6):26-32.
51. Grover M, Cummings J, Janicki T. Moving beyond coding: why secure coding should be implemented. *J Informat Syst Appl Res*. 2016;9(1):38.
52. Higuera JRB, Higuera JB, Sicilia Montalvo JA, Villalba JC, Pérez JJN. Benchmarking approach to compare web applications static analysis tools detecting OWASP top ten security vulnerabilities. *Comput Mater Continua*. 2020;64:3.
53. Li J. Vulnerabilities mapping based on OWASP-SANS: a survey for static application security testing (SAST). *Ann Emerg Technol Comput*. 2020;1-8.
54. Sahu DR, Tomar DS. Analysis of web application code vulnerabilities using secure coding standards. *Arabian J Sci Eng*. 2020;42(3):885-895.

55. Long F, Mohindra D, Seacord R, Sutherland D, Svoboda D. *The CERT Oracle Secure Coding Standard for Java*. Addison-Wesley Professional; 2011.
56. Ballman A, Svoboda D. Avoiding Insecure C++—How to Avoid Common C++ Security Vulnerabilities. *2016 IEEE Cybersecurity Development (SecDev)*. IEEE; 2016.
57. Nguyen T-T, Aoki T, Tomita T, Yamada I. Multiple Program Analysis Techniques Enable Precise Check for SEI CERT C Coding Standard. *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE; 2019.
58. Standard CC. *SEI CERT*; Software Engineering Institute; 2016.
59. Nguyen TT, Maleehuan P, Aoki T, Tomita T, Yamada I. Reducing false positives of static analysis for SEI cert C coding standard. *2019 IEEE/ACM Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER&IP)*. IEEE; 2019.
60. https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices.
61. https://www.securecoding.com/blog/owasp-secure-coding-checklist/.
62. https://www.whitehatsec.com/glossary/content/secure-coding-standards.
63. Rafi S, Yu W, Akbar MA, Mahmood S, Alsanad A, Gumaei A. Readiness model for DevOps implementation in software organizations. *J Software Evolu Process*. 2021;33(4):e2323.
64. Khan SU. *Software outsourcing vendors' readiness model (SOVRM)*. Keele University; 2011.
65. Chrissis MB, Konrad M, Shrum S. *CMMI for development: guidelines for process integration and product improvement*. Pearson Education; 2011.
66. Garcia VC. *RiSE reference model for software reuse adoption in Brazilian companies*; PhD thesis RiSE Maturity Model for Reuse Adoption; 2010.
67. Niazi M, Wilson D, Zowghi D. A maturity model for the implementation of software process improvement: an empirical study. *J Syst Software*. 2005;74(2):155-172.
68. Rashid Y, Rashid A, Warraich MA, Sabir SS, Waseem A. Case study method: A step-by-step guide for business researchers. *Int J Qual Methods*. 2019;18:1609406919862424.
69. Daskalantonakis MK. Achieving higher SEI levels. *IEEE Software*. 1994;11(4):17-24.

How to cite this article: Humayun M, Niazi M, Jhanjhi NZ, Mahmood S, Alshayeb M. Toward a readiness model for secure software coding. *Softw: Pract Exper*. 2023;53(4):1013-1035. doi: 10.1002/spe.3175

APPENDIX A

Practices	Key activity evaluation dimensions: Approach (score range: 0,2,4,6,8,10)	Deployment (score range: 0,2,4,6,8,10)	Results (score range: 0,2,4,6,8,10)	Average score (average of the three dimensions values)
Level 1: Initial				
1. Keep the code simple	8	8	8	8
2. Adapt deny by default policy	10	10	10	10
3. Adhere to the principle of least privileges	10	8	8	8.7
4. Use the compiler's highest warning level	10	10	10	10
Sum of "average scores"				36
Overall score				9.17
Level 2: Managed				
1. Protect your data	10	10	10	10
2. Define and implement a session management strategy	10	10	10	10
3. Define and implement access control mechanism,	10	10	10	10
4. Define and implement a policy for system configuration	10	8	8	8.7
5. Design and implement database security policy	10	10	10	10
6. Design and implement file management policy	10	10	10	10
7. Design and implement a memory management strategy	8	8	8	8
8. Design and implement error handling and exception strategy	8	10	8	8.7
9. Design and implement authentication and password management strategy	10	10	10	10
Sum of "average scores"				85.4
Overall score				9.5
Level 3: Well-defined				
1. Develop policies for secure architecture and design	8	8	8	8
2. Implement communication security mechanism	10	10	10	10
3. Adapt cryptographic practices	10	10	10	10
Sum of "average scores"				28
Overall score				9.33
1. Sanitize data for input and output purposes	6	8	8	7.3
2. Practice defense in depth	10	10	10	10
3. Use effective quality assurance techniques	8	8	9	8.3
4. Adapt a secure coding standard	10	10	10	10
Sum of "average scores"				35.6
Overall score				8.9
Level 5: Optimized				
1. Enforce security policies	8	8	8	8
2. Implement layered defense	10	10	10	10
3. Define and implement threat modeling policy	10	8	8	8.3
4. Develop a secure coding standard	8	6	8	7.3
Sum of "average scores"				33.6
Overall score				8.4

APPENDIX B

Level 1: Initial				
	Key activity evaluation dimensions			
Practices	Approach (score range: 0,2,4,6,8,10)	Deployment (score range: 0,2,4,6,8,10)	Results (score range: 0,2,4,6,8,10)	Average score (average of the three dimensions values)
1. Keep the code simple	8	6	8	7
2. Adapt deny by default policy	10	8	8	9
3. Adhere to the principle of least privileges	6	6	6	6
4. Use the compiler highest warning level	**8**	8	8	8
Sum of "average scores"				30
Overall score				7.50
Level 2: Managed				
1. Protect your data	8	8	8	8
2. Define and implement a session management strategy	8	6	6	7
3. Define and implement access control mechanism,	10	8	8	9
4. Define and implement a policy for system configuration	8	8	8	8
5. Design and implement database security policy	8	6	8	7
6. Design and implement file management policy	10	8	8	9
7. Design and implement a memory management strategy	6	6	6	6
8. Design and implement error handling and exception strategy	10	10	8	9
9. Design and implement authentication and password management strategy	8	8	6	7
Sum of "average scores"				70
Overall score				7.78
Level 3: Well-defined				
1. Develop policies for secure architecture and design	8	6	8	7
2. Implement communication security mechanism	6	6	8	7
3. Adapt cryptographic practices	8	6	8	7
Sum of "average scores"				21
Overall score				7
1. Sanitize data for input and output purposes	6	4	6	5
2. Practice defense in depth	8	8	6	7
3. Use effective quality assurance techniques	8	8	10	9
4. Adapt a secure coding standard	8	8	8	8
Sum of "average scores"				29.33
Overall score				7.33
Level 5: Optimized				
1. Enforce security policies	8	10	8	9
2. Implement layered defense	8	6	6	7
3. Define and implement threat modeling policy	6	8	6	7
4. Develop a secure coding standard	8	6	8	7
Sum of "average scores"				29.33
Overall score				7.33

APPENDIX C

Practices	Key activity evaluation dimensions: Approach (score range: 0,2,4,6,8,10)	Deployment (score range: 0,2,4,6,8,10)	Results (score range: 0,2,4,6,8,10)	Average score (average of the three dimensions values)
Level 1: Initial				
1. Keep the code simple	10	6	6	5
2. Adapt deny by default policy	8	6	4	6
3. Adhere to the principle of least privileges	8	4	4	3
4. Use the compiler highest warning level	6	6	4	3
Sum of "average scores"				17
Overall score				4.25
Level 2: Managed				
1. Protect your data	10	6	6	7
2. Define and implement a session management strategy	10	8	6	8
3. Define and implement access control mechanism,	6	4	4	5
4. Define and implement a policy for system configuration	10	8	6	8
5. Design and implement database security policy	10	6	4	7
6. Design and implement file management policy	10	8	6	8
7. Design and implement a memory management strategy	6	4	4	5
8. Design and implement error handling and exception strategy	8	6	6	7
9. Design and implement authentication and password management strategy	10	8	8	9
Sum of "average scores"				64
Overall score				7.11
Level 3: Well-defined				
1. Develop policies for secure architecture and design	10	8	4	7
2. Implement communication security mechanism	8	6	6	7
3. Adapt cryptographic practices	10	8	8	9
Sum of "average scores"				23
Overall score				7.66
1. Sanitize data for input and output purposes	4	2	2	3
2. Practice defense in depth	6	4	4	5
3. Use effective quality assurance techniques	6	6	6	6
4. Adapt a secure coding standard	10	6	4	7
Sum of "average scores"				21
Overall score				5.25
Level 5: Optimized				
1. Enforce security policies	10	6	6	7
2. Implement layered defense	8	4	6	6
3. Define and implement threat modeling policy	10	6	4	7
4. Develop a secure coding standard	10	6	6	7
Sum of "average scores"				27
Overall score				6.75