International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Security Risks in the Software Development Lifecycle

Mamdouh Alenezi, Sadiq Almuairfi

Abstract: ***Security is a significant concern in software development. Risks and errors should be reduced and as much as possible eliminated. Especially with how the computer and internet provide numerous technological benefits and are being optimally utilized in the present times, users have high expectations when it comes to secure software products. The development cycle in software design is comprised of a systematic process wherein security risk assessment should be integrated into each phase. In order to ensure quality software with a high level of security, the best practices in risk management should be implemented from the beginning and should be performed throughout the process. While developing highly secure software is a complex task, therefore, it must involve more dedicated security activities that are usually ignored in traditional SDLC. In this paper, the nature and phases of secure software development will be discussed as well as the security risk assessment models and practices involved in it. This will provide software developers, programmers, or engineers the awareness on secure software development cycle which will allow them to plan efficient strategies and enhance their performance, thus, resulting in quality and reliable output. In addition, we present a qualitative study looking at real-life practice employed towards software security risk. We reflect on how well current risk practices follow best practice, identify pitfalls, and explore why these occur & mitigate.***

Keywords: Risk Management, Security, Software Development Cycle, Software Engineering.

I. INTRODUCTION

In the software development cycle, security is an important concept. In the future, it may even become more problematic. A security issue has no easy solution. That is why software developers must integrate security from the beginning of the software development process. Throughout the process, security must be taken into account to make sure that the software product has optimal security as discussed by [18]. Continuous monitoring and evaluation can lead to early detection of errors or problems, prompt implementation of appropriate actions, and most importantly, production of the most desirable secure software product.

Manuscript published on 30 September 2019
* Correspondence Author
Mamdoh Alenezi*, College of Computer and Information Science, Prince Sultan University, Riyadh, Saudi Arabia, malenezi@psu.edu.sa
Sadiq Almuairfi, Information Technology Center, Prince Sultan University, Riyadh, Saudi Arabia. salmuairfi@psu.edu.sa

Devanbu and Stubblebine [8] stressed focusing on security issues at all software development phases and outlined how to focus on requirements and design processes which are the two initial developmental levels. Gupta [11] discussed risk management strategies at the early stages of software development and claimed that late risk management indirectly poses greater threats to secure software development. Addressing, analyzing, and mitigating security risks will result in a variety of advantages for software development projects. It will enable developing truly secure systems, monitoring and protecting critical assets, supporting efficient security decision-making policies, building practical security policies, and providing insightful data for future estimation and even more.

For software systems, software security is an important requirement. A software application is effective if it is written to attain the intended objective and to make sure that it allows a satisfying experience for the user in an entirely safe environment.

Oftentimes, writing applications that cover all the necessary criteria for it to be successful can be very challenging. The key question is" How can a software have the resilience to flaws in security?" Engineers often have less awareness of security approaches that lead to functional software, but particularly susceptible to security threats. And because the development approach has various flaws, security issues will possibly be encountered when the software's functional role is given emphasis. This is why engineers revisit the work after the completion of the development process in an attempt of improving security aspects [4].

In the software world, it is essential to understand the security risks associated with the software system you are building [1]. Software security is a philosophy that includes but not limited to concepts, checklists, tools, processes, and methods [17]. These are important to architect and design, code and construct, and verify and test software systems. Basic security goals such as confidentiality, integrity, and availability, together known as the CIA triad, along with security risks controlling and monitoring activities and concepts such as assets, threats, and vulnerabilities [2]. In addition, even though there are some security measures are in place, there are still risks a software might face that need to be addressed [6].

Risk management systems have played an essential role in the software industry. It involves strategic practices in software engineering that would lead to an efficient software development process.

These practices enable a disciplined environment for effective decision-making through the assessment of existing and potential problems in software development as explained by [20].

The de facto way of handling risks is risk management [2].

Risk management does not eliminate risks but it is a systematic way of finding, analyzing, mitigating, and handling risks. Its main objective is to identify probable issues before they occur in order to handle them in an efficient professional way. It should begin as early as possible and continue throughout the total life cycle of the project [1]. Boehm [6] stated in his masterpiece about risk management that identifying and addressing risks early in the development stages would help in reducing the software project cost. This is applicable also to security risks. Risk management is a continuous process that spans the whole software development cycle. "Fig. 1" shows the Boehm risk management process.

As evidence, the above research gaps remain in addressing the human aspects of software security risks and stage level risk assessment. Our paper takes a holistic perspective to explore real-life security risk practices, an important step in improving the current situation.

II. RISKS TYPES AND CLASSIFICATION

Risk is considered a future uncertain event with a certain probability of happening and impact or loss. Risk is the expectation of the loss or damage. It is very essential to quantify the level of uncertainty and the degree of loss associated with each risk. Risk is the factor which should be identified before going through software security. Risks can be broadly divided into two categories which are proactive risks and reactive risks. Proactive risks are the pre-assumptions of risks are to occur in the future. Reactive risks are when there any problem occurs after deployment of software.

Fig. 1.Boeh's Risk Management Process.

Secure software is a need of today life of the internet, the software is secure when its risks are identified earlier and managed. The identification, mitigation, and monitoring of risk is the key factor of secure software.

Risk management is the process of identifying, addressing, and eliminating the risks before they can damage the project. It identifies software risks and plans to avoid risks and minimize their effects if they occur. All risks cannot be avoided but by performing risk management, we can attempt to ensure that the risks are minimized and managed [15].

Any software development project is usually facing unforeseen issues, which may raise risks within the project development. Controlling these risks arise from both the technical and non-technical development is crucial to arrive at a successful project. Karim et al. [14] outlined that in recent studies, it has been discovered that various methodologies in software development do not comprehensively constitute methods for the incorporation of information security into the SDLC. Foreseeably, several software products that undergo live testing are found to have a vulnerability to threats and have a failure in providing a safe environment for clients and users. This is likely caused by the inadequacy of systematic measures including frameworks, procedures, or reviews.

All software has been facing threats from many possible malicious adversaries which have increased every day. Threatened software involves complex telecommunications, accessibility of power systems over the internet, copy protection mechanisms on commodity software, and personal computers running internet-aware applications. These and other threats can significantly challenge software engineers who are responsible for activities in risk management, particularly in designing security measures. Many applications are developed without taking into consideration security services including access control, integrity, confidentiality, and non-repudiation as mentioned in [14].

The risks influences can be external or internal to the company. Therefore, risk can be classified into systematic and unsystematic as explained in [7]. Systematic risk refers to risks caused by external factors. Hacking, Denial of services, virus, fire, and power loss are sources of systematic risk. Whereas, unsystematic risk is the portion of total risk that is unique to the enterprise. Data loss or misuse, human error, application error, and inside attack can be examples for unsystematic risk.

III. RISK ASSESSMENT AND MITIGATION

Risk assessment merges between vulnerability analysis and threat impact evaluation to reach an overall conclusion about the risk level.

Risk assessment involved in finding risks and determining which is considered harmful or not. In addition, it documents the findings, implementations, regular updates, and reviews. Because of the nature of software, the occurrence of a number of risks has become a key concern in the industry. This is why it is important to have efficient strategies for risk management in the software development process. Pandey et al [4] outlined that the occurrence of risk is inevitable; therefore, risk mitigation strategies have been implemented since the past two decades. Thus, Mitigation is one of the actions involved in software risk management.

Quality assurance involves management of essential data needed to evaluate product quality. The process begins with planning and carrying out reviews and inspection. This is an ongoing process throughout the software development cycle wherein the software being developed is routinely checked ensuring that it has met the necessary quality measures.

A procedure called" Software Quality" is used for the assessment, evaluation, and improvement of software performance. It is defined as the extent by which the software has met the requirements for portability, maintainability, reliability as contrasted with interface, functional, and performance requirements that are fulfilled as an output of software engineering. It is a planned and systematic approach in evaluating quality, standards, procedures, and processes of the software product [13]. Since security is considered a quality attribute, security risk has to be investigated in the context of software risk. In the security world, incidents are measured against assets, threats, and vulnerabilities [9] as shown in "Fig.2".

Fig. 2.Security Risk and Related Elements.

IV. SECURE SOFTWARE DEVELOPMENT LIFE CYCLE (SSDLC)

Software Development Life Cycle is a significant concept utilized in software engineering, particularly in describing a method for planning, generating, coding, testing, and implementation of a user requirement specification. It involves a range of software and hardware configurations. It is a step by step procedure for the creation of quality software. It entails several phases that are followed accordingly and are important for software engineers including analysis, planning, coding, design, testing, and implementation. The software development lifecycle consists of many phases. Generally speaking, each software process consists of at least five main phases: Requirements, Architecture and Design, Implementation, Verification, and Release & Maintenance. "Fig.3" shows these phases with practical things that can be done in each phase to make the software more secure.

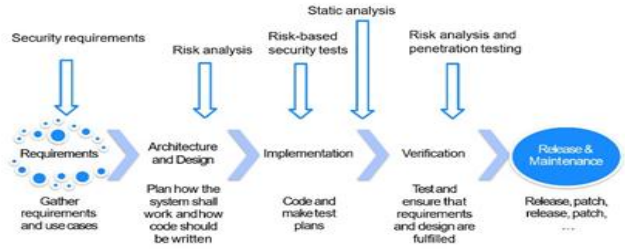

Fig. 3.Typical Software Development Lifecycle.

Software Development Life Cycle (SDLC) needs to consider security in each phase to avoid complexity and problems.

Karim et al [14] illustrate that security is essential in providing authentication, integrity, and availability in SDLC. Since mistakes in software security are common and there have been increasing threats, it is necessary that security is considered in the early stage of the software development cycle and that security principles are applied as a standard element of the cycle. Adopting practices that cause reduction of software defects leading to minimization of potential risk due to lack of attention to security throughout the development process.

A. Pre-requirements Phase

Security actions are done at this phase establish the base of all the actions starting from requirements until testing and maintenance phase. The following tasks are performed during this phase:

1) Security Training and Awareness

Many firms do not give enough attention to the importance of security and how to improve it. If higher management does not support and value security, then they will not support and spend on security activities that are essential to achieve it. Security training can be classified into two categories general training and specific training as explained by [10]. General training is for the company as a whole. The goal is to create security awareness in the company's culture and let everyone starts thinking about security in their projects, task, and actions. Specific training is limited for the project or dedicated team only. It varies from team to team. The goal of specific training is to identify the flaws; threats associated with them, and identify security measures that should be taken to secure software from these threats. General and specific training build a security culture within the organization with enough seriousness in tasks and activities [10]. Security training is successful if each individual is aware of security importance and knows what role he or she has to play towards the development of secure software.

2) Develop a framework for Risk Management

The goal is to identify risk elements early in the development cycle and develop some strategies to resolve risks findings. Set down some agenda to resolve new risk items that are highlighted [16]. Risk management framework should be implemented in the beginning, so that top management and stockholder get educated about the risks and their possible impact. Risk management framework should work as a reminder of security importance over a period of time.

B. Requirements Phase

Systematic measures such as frameworks and procedures can help project managers and developers in ensuring that security processes are being followed consistently in every part of the development process with accordance with the set of rules.

The requirements phase involves the collection of all user requirements done by software engineers in order to proceed with software development. Discussions are held by the software development team with the users to address issues and try to bring out all necessary information on their requirements. The requirements are gathered from the user, functional requirements, and system requirements. Collection of requirements are done through studying the existing software and system, conducting user interviews, looking at the database, or using questionnaires to collect answers. It is important to carefully perform this phase as all collected information can affect software quality.

On the other hand, "Develop Misuse Cases" is usually performed during the requirement phase. Where, the focus is on "what" and not on "how" during requirement gathering. The next step is to handle "how" on an abstract level without going into the design and implementation details when functional, non-functional and security requirements are identified, gathered, analyzed, and documented. Security requirements describe what the system should not do, while functional requirements describe what the system should do [10]. Use cases handle functional requirements whereas misuse cases handle eliciting security requirements and help in understanding possible attacks, actors, relationship with allowed functionalities, estimating the amount of loss, and determining system's behavior before and after the attacks.

C. Architecture Phase

Software architecture entails a discipline in the creation and the documentation of high-level structure in a software system. The practice of designing software is similar and comparable to a building's architecture. Documentation of software ensures communication among stakeholders, facilitates early decisions regarding high-level design, and enables reuse of design elements between projects as mentioned in [12]. The best architecture always involves having sufficient software knowledge and skills as well as the implementation of this knowledge and skills in the software development cycle.

Software architects perform several activities. They usually work with project managers, discuss architecturally significant requirements with stakeholders, come up with software architecture, evaluate, and communicate the architecture with the rest of the team. There are four core activities in software architecture design. These core architecture activities are performed iteratively and at different stages of the initial software development life cycle, as well as over the evolution of a system.

Architectural Analysis is about understanding the surrounding environment where the system is going to operate and determining what is needed for it to operate. The input or requirements to the analysis activity can come from any number of stakeholders and include items such as:

- (Functional requirements) or what the system will do when it is functional.
- (Non-Functional) or quality attributes such as reliability, efficiency, maintainability, performance, and security.
- Business requirements and environmental contexts of a system that may change over time, such as legal, social, financial, competitive, and technology concerns.

Architectural Synthesis is where the actual creation of the architecture. The input is requirements from the analysis stage, the current state is creating the architecture and then move to the next stage where the created architecture is evaluated and improved.

Architecture Evaluation is determining the health and wealth of the created architecture. The evaluation decides about architecture decisions and how they are achieving or not achieving the desired requirements. Some of the available software architecture evaluation techniques include the Architecture Tradeoff Analysis Method (ATAM).

Architecture Evolution [3] is maintaining and adapting existing software architecture to meet new requirements and environmental changes. Some of these new requirements and technology advancement can cause some changes at the architecture level. As such, architecture evolution is concerned with adding new functionality as well as maintaining existing functionality and system behavior.

D. Design Phase

Today, most researches are emphasizing the idea of addressing the security issues at the initial phase of the software development life cycle. The early detection and correction of security risks are said to help deal with the prevalent security aspects in software development [22]. Therefore, the recognition of different security risks at the design phase will help avoid the loopholes that may pose a threat to the security of the system in the future. If the design itself is prepared in such a way that the security-related risks are evaluated, then it may help in the reduction of time and cost that is spent on system security software. Detecting and rectifying bugs after development is found to be 100 times critical as compared to considering them at the design phase. Therefore, it is suggested to address the security-related risks at early stages of SDLC.

There are rules in designing software which can be used in building secure frameworks, in the improvement of programming frameworks' security, and in handling issues that prevent secure software advancement.

The Design phase involves bringing down the entire knowledge of requirements and analysis for software design. Various designs such as object-oriented design and functional design are available. Several tools such as data dictionaries, entity-relationship diagram, and flow diagram can be utilized for designing. All information gathered is inputted in this phase as discussed by [15].

E. Coding (Implementation) Phase

In the coding phase, software development starts with written program code with the use of appropriate programming language as well as the efficient development of executable programs that are error-free.

The integration of software with the databases, libraries, and other programs may be necessary. In order to develop a quality software product, the software development team must possess expert level programming skills.

The implementation or coding of the software project follows mostly regular software engineering best practices. The code should be checked into a source code repository using version control such as Git or svn [22]. The add-ons onto the version control can make magical help. They can statically analyze the code, identify security concerns in the code and more. A strict coding standard should define and forced. Each environment and program language need different requirements. Some of them require more focus on exception handling or memory management.

Newly added or modified source code should go through a formal code review process. Code review can be manual or automated. Automated code review is now part of the continuous integration process where any addition or modification of the code goes through several steps that make sure the code is up to the standards and does not break existing code [5]. The code review process must be integrated into the development process, working naturally alongside development. There are different available tools that can help in that regard such as GitHub, Gerrit, and many others. For a new project, it is important to evaluate the features of the different systems and to choose the one that best integrates into the development process [22].

F. Testing (Verification) Phase

Software testing is an essential part of software development. Each new commit and release must be thoroughly tested for functionality and security. Testing methodologies such as incorporation testing or unit testing are utilized for testing use of the created software as per [14]. This is to facilitate the removal of mistakes or errors ensuring that the software product has good quality. Testing experts conduct testing at different levels of code including program testing, object-oriented testing, module testing, and product testing at dynamic and static levels.

Security testing is very different from functional testing because functional testing focuses on achieving functional criteria whereas security testing handles an abstract property which is not inherently testable. Crashing test cases indicate some bugs but there is no guarantee that a bug will cause a crash.

Automatic security testing based on fuzz testing, penetration testing, symbolic execution, or formal verification tests security aspects of the project, increasing the probability of a crash during testing as explained in [22].

A dedicated security response team is essential to answer to any threats and discovered vulnerabilities. They are the primary contact for any flaw or vulnerability and will triage the available resources to prioritize how to respond to them including changes to in the environment because of the software evolution.

G. Release and Maintenance Phase

The last stage of a Software Development Lifecycle process is its deployment and maintenance. Here a lifecycle maintenance approach is implied which tends to define how the various releases and changes to the software will be catered and how it will be used to meet different business objectives a developer or two may have to permanently implied towards the maintenance of that product and to cater to various client's needs and requirements.

This phase may also embark the actual installation of the software product with all its components and database on the production environment. This usually may be struck with various kinds of complexities that come with this sort of integration Payer [22] mentioned that an update and patching strategy defines how to react to said flaws, how to develop patches, and how to distribute new versions of software to the users. Developing a mechanism to securely update the infrastructure is challenging. The component responsible for security updates must be designed in a way to frequently check for new updates while considering the load on the update servers. Updates must be verified and checked for correctness before they are installed. During the operation and maintenance phase, the software is further optimized, and new functionalities and capabilities are added to it with time.

V. RISKS MODELS

Several risk models have been developed in order to improve the security level of SDLC. Neves and Silva [19] discussed that for software projects, analyzing risk-related information can lead to an improvement in the decision-making process of managers.

Based on the obtained information, proper risk management can then be implemented in software development. Identifying the several existing and potential risk factors will greatly contribute to the reduction and elimination of the likelihood of problems.

The risk management in software development cycle involves the use of various models. These models are the Waterfall, Spiral, V-Model, and Multilevel Security Spiral (MSS). The Waterfall model is a development methodology that involves a continuous process wherein the result of a certain development phase becomes the input for the other as shown in [21]. The Spiral model is characterized as a model that is driven by risk and is used in guiding the process of building systems. The V-Model, also known as the Vee-Model, is used in describing the series of steps in the development cycle. The left part of the "V" is a representation of the decomposition of requirements and the formation of system specifications.

The right side of the "V" is a representation of the integration and the verification of parts. The "V" shape is an illustration of how every design phase is associated with a corresponding counterpart during the testing phase as explained in [23].

Waterfall and Spiral models are commonly used models in the development of the system. The principal focus of security spiral is the identification of security risks and management of those risks.

The spiral model is an organized approach for software development and based on risk perception. In Multilevel Security Spiral (MSS), security is considered from the very start of the software development cycle. Each cycle of the MSS begins with identifying the objectives, limitations, and alternatives possible for developing the software. After the identification of risk, a strategy is then developed for the resolution of the risks and uncertainties. After resolving the risks, the next step of creating the software is then performed. While taking into consideration the risks, the programmer must make necessary revisions after every step.

Usually, it is hard to know what to look for and where to start. Software security models are here to help shed some light on what is needed.

These models can broadly be categorized as descriptive and prescriptive models. Descriptive models can be used to determine how a company is doing compared to others. Prescriptive models are models that describe what one should do. An example of a descriptive model is Building Security in Maturity Model (BSIMM) which is a model that describes how the state of software security is at the moment of the study. BSIMM can help in the risk management process by explaining security measures other organizations have. From this, the organization can find security measures most interesting for them. "Fig. 4" shows the Building Security in Maturity Model (BSIMM).

Governance	Intelligence	SSDL Touchpoints	Deployment
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

Fig. 4.. Building Security in Maturity Model (BSIMM).

An example of a prescriptive model is The Open Software Assurance Maturity Model (SAMM)which is an open framework developed by the Open Web Application Security Project (OWASP) to help organizations formulate and implement a strategy for software security that is tailored to the specific risks facing the organization. OpenSAMM offers a roadmap and well-defined maturity model for secure software development and deployment, along with useful tools for self-assessment and planning. SAMM is composed of 12 security practices, in four different business functions, as shown in "Fig.5".

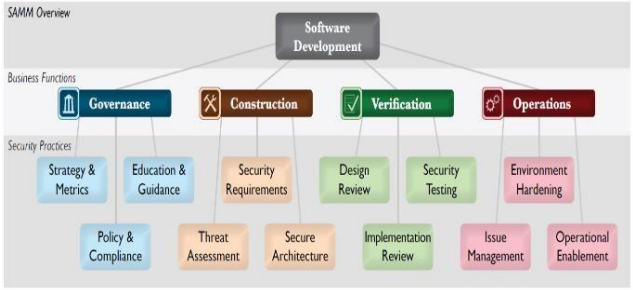

Fig. 5.The structure of SAMM, consisting of 12 security practices in four business functions.

VI. BEST PRACTICE AND EXAMPLE OF RISK CONTROLS DURING THE SOFTWARE LIFECYCLE

The most important step of risk management is the risk controls which comes after risk identification, risk analysis, and risk prioritization. Wilson et al [24] found that software development involves groups of people working on computational problems, traditional laboratory, and more daily operations to develop new algorithms, manage and analyze a large number of data, combine different datasets to evaluate computational tasks and problems. There are best practices in software development which entail comprehensive research and experience as its solid foundations. These help in writing more maintainable and reliable code with lesser effort.

Best practices and examples of risk controls include:

1) Writing programs for people rather than for machines wherein:

- a program must not require the users to take in too many facts at once
- names are meaningful, distinctive, and consistent
- the formatting and style of the code are consistent

2) Letting the computer do the job by:

- Making the computer able to repeat tasks
- Saving in a file the recent commands for reuse
- Automating workflows through the use of a build tool.

3) Making incremental changes through:

- Continuous feedback and correction of course while working in small steps
- The use of a version control system
- Putting in version control all that have been manually created

4) Not repeating oneself or others in which:

- The system must have a single authoritative representation for each data
- Codes are modularized instead of being copied and pasted.
- Codes are reused rather than being rewritten

5) Planning for mistakes by:

- Adding assurance that how programs operate will be checked
- Utilizing an existing and reliable testing unit
- Conducting test activities for bugs
- Using a symbolic debugger

6) Making effective use of software only once it is proven that it functions correctly by:

- Identifying bottlenecks through the use of a profiler
- Writing code in the best language level possible

7) Documenting the purpose and design rather than the mechanics wherein:

- Interfaces and reasons are recorded
- Code is refactored to provide an explanation of how it works
- The documentation is embedded for a piece of software

8) Collaborating through:

- The use of pre-merge code reviews
- The use of pair programming when distinctly tricky problems are tackled and when someone new is being brought up to speed.
- The use of a problem tracking tool

Building secure software through the incorporation of security best practices can result in good practices in software engineering. To efficiently deal with existing security measures in various development projects, the incorporation of security-minded thinking should be considered throughout the process of development. This can reduce the risk of lacking in necessary security requirements or committing critical faults in software design. Through utilizing this strategy, early discovery of problems can be possible. Discovering problems at the end of the process will lead to difficulty in handling them and can cost a lot when fixing them During the Software Development Cycle, vulnerabilities linked to the system can be lessened to a minimum level of security is dealt with as a continuing process [14].

Since the software is oftentimes utilized for more than one type of project and is frequently reused, computation of errors can have a disproportionate effect on the process. This kind of cascading impact leads to various prominent retractions if a mistake from the code of another group was not detected until after the publication. Thus, awareness of the best practices is necessary for the improvement of the approaches and for the evaluation of others' computational work [24].

VII. CONCLUSION

In summary, considering that security plays a crucial role in every software product, it is necessary to utilize best practices throughout the software development cycle. Risk assessment should be conducted starting from the beginning of the process and should be performed as an ongoing procedure integrated into each phase of the cycle. Risk management should be efficiently performed when issues or errors are detected. This enables meeting all the requirements for software development ensuring secure software, and in turn, will satisfy clients and users.

REFERENCES

1. Ibrahim Abunadi and Mamdouh Alenezi. An empirical investigation of security vulnerabilities within web applications. Journal of Universal Computer Science, 22(4):537–551, 2016.
2. Thamer Al Hamed and Mamdouh Alenezi. Business continuity management & disaster recovery capabilities in Saudi Arabia ICT businesses. International Journal of Hybrid Information Technology, 9(11):99–126, 2016.
3. Mamdouh Alenezi and Fakhry Khellah. Architectural stability evolution in open-source systems. In Proceedings of The International Conference on Engineering & MIS 2015, page 17. ACM, 2015.
4. Md Tarique Jamal Ansari, Dhirendra Pandey, and Mamdouh Alenezi. Store: Security threat oriented requirements engineering methodology. Journal of King Saud University-Computer and Information Sciences, 2018.
5. Hala Assal and Sonia Chiasson. Security in the software development lifecycle. In the Fourteenth Symposium on Usable Privacy and Security ({SOUPS} 2018), pages 281–296, 2018.
6. Barry W. Boehm. Software risk management: principles and practices. IEEE Software,8(1):32–41, 1991.
7. Abdullah Al Murad Chowdhury and Shamsul Arefeen. Software risk management: importance and practices. International Journal of Computer and Information Technology (IJCIT), pages 2078–5828, 2011.
8. Premkumar T Devanbu and Stuart Stubblebine. Software engineering for security: a roadmap. In Proceedings of the Conference on the Future of Software Engineering, pages 227–239. ACM, 2000.
9. Nan Feng, Harry Jiannan Wang, and Minqiang Li. A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis. Information sciences, 256:57–73, 2014.
10. Radek Fujdiak, Petr Mlynek, Pavel Mrnustik, Maros Barabas, Petr Blazek, Filip Borcik, and Jiri Misurec. Managing the secure software development. In 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pages1–4. IEEE, 2019.
11. Sandeep Gupta. A proactive approach to information security. Technical report, 2003. Also available as https://www.giac.org/paper/gsec/3532/ proactive-approach-information-security/105749.
12. Mohammad Imran, Abdulrahman A Alghamdi, and Bilal Ahmad. Software engineering: Architecture, design and frameworks. International Journal of Computer Science and Mobile Computing, 5(3):801–815, 2016.
13. Shareeful Islam. Software development risk management model: a goal driven approach. In Proceedings of the doctoral symposium for ESEC/FSE on Doctoral Symposium, pages 5–8. ACM, 2009.
14. Nor Shahriza Abdul Karim, Arwa Albuolayan, Tanzila Saba, and Amjad Rehman.The practice of secure software development in SDLC: An investigation through existing model and a case study. Security and Communication Networks, 9(18):5333–5345, 2016.
15. Jasleen Kaur, Alka Agrawal, and Raees Ahmad Khan. Major software security risks at design phase. ICIC Express Letters, 12(11):1155–1162, 2018.
16. Armin Lunkeit and Hartmut Pohl. Model-based security engineering for secure systems development. In ARCS Workshop 2018; 31th International Conference on Architecture of Computing Systems, pages 1–10. VDE, 2018.
17. Gary McGraw. Four software security findings. Computer, 49(1):84–87, 2016
18. Hanif Mohaddes Deylami, Iman Ardekani, Ravie Chandren Muniyandi, and Hossein Sarrafzadeh. Effects of software security on software development life cycle and related security issues. International Journal of Computational Intelligence and Information Security.
19. Sandra Miranda Neves and Carlos Eduardo Sanches da Silva. Risk management applied to software development projects in incubated technology-based companies: literature review, classification, and analysis. Gest˜ao & Produc˜ao, 23(4):798–814, 2016.
20. Maruf Pasha, Ghazia Qaiser, and Urooj Pasha. A critical analysis of software risk management techniques in large scale systems. IEEE Access, 6:12412–12424, 2018.
21. Rupali Pravinkumar Pawar. A comparative study of agile software development methodology and traditional waterfall model. IOSR Journal of Computer Engineering (IOSR-JCE), 2(2):1–8, 2015.
22. Mathias Payer. Software Security: Principles, Policies, and Protection. HexHive Books, 0.35 edition, April 2019.
23. Giuditta Pezzotta, Sergio Cavalieri, and Paolo Gaiardelli. A spiral process model to engineer a product service system: an explorative analysis through case studies. CIRP Journal of Manufacturing Science and Technology, 5(3):214–225, 2012.

24. Greg Wilson, Dhavide A Aruliah, C Titus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven HD Haddock, Kathryn D Huff, Ian M Mitchell, Mark D Plumb- ley, et al. Best practices for scientific computing. PLoS biology, 12(1):e1001745, 2014.

AUTHORS PROFILE

Dr. Mamdouh Alenezi is currently the Dean of Educational Services at Prince Sultan University. Dr. Alenezi received his MS and Ph.D. degrees from DePaul University and North Dakota State University in 2011 and 2014, respectively. He has extensive experience in data mining and machine learning where he applied several data mining techniques to solve several Software Engineering problems. He conducted several research areas and development of predictive models using machine learning to predict fault-prone classes, comprehend source code, and predict the appropriate developer to be assigned to a new bug.

Dr Sadiq Almuairfi is currently an E-Service director and a researcher at Security Engineering Lab in Prince Sultan University, Riyadh, Saudi Arabia. He received his PhD in Cyber-security from La Trobe University, Melbourne, Australia, in 2014 and his Master degree in Information Management from King Abdulaziz University, Jeddah, Saudi Arabia, in 2005 and his Bachelor degree in Computer Engineering from KFUPM, Dhahran, Saudi Arabia in 2001. His research interests include Cyber-security, Network Security, E-Commerce Security .