Hindawi
Journal of Engineering
Volume 2024, Article ID 9962691, 15 pages
https://doi.org/10.1155/2024/9962691

Research Article

Mitigating Software Vulnerabilities through Secure Software Development with a Policy-Driven Waterfall Model

Shariq Hussain,[1] **Haris Anwaar**,[2] **Kashif Sultan**,[3] **Umar Mahmud**,[1] **Sherjeel Farooqui**,[1] **Tehmina Karamat**,[1] **and Ibrahima Kalil Toure**[4]

[1]*Department of Software Engineering, Foundation University Islamabad (FUI), Islamabad, Pakistan*
[2]*Department of Electrical, Electronics and Telecommunication Engineering, University of Engineering and Technology, Lahore, Pakistan*
[3]*Department of Software Engineering, Bahria University Islamabad, Islamabad, Pakistan*
[4]*Department of Computer Science, Gamal Abdel Nasser University, Conakry, Guinea*

Correspondence should be addressed to Ibrahima Kalil Toure; ikalil@msn.com

Received 12 November 2023; Revised 24 January 2024; Accepted 14 February 2024; Published 21 February 2024

Academic Editor: Assed Naked Haddad

For the past few years, software security has become a pressing issue that needs to be addressed during software development. In practice, software security is considered after the deployment of software rather than considered as an initial requirement. This delayed action leads to security vulnerabilities that can be catered for during the early stages of the software development life cycle (SDLC). To safeguard a software product from security vulnerabilities, security must be given equal importance with functional requirements during all phases of SDLC. In this paper, we propose a policy-driven waterfall model (PDWM) for secure software development describing key points related to security aspects in the software development process. The security requirements are the security policies that are considered during all phases of waterfall-based SDLC. A framework of PDWM is presented and applied to the e-travel scenario to ascertain its effectiveness. This scenario is a case of small to medium-sized software development project. The results of case study show that PDWM can identify 33% more security vulnerabilities as compared to other secure software development techniques.

1. Introduction

Since the advent of computers in all aspects of our daily life, we have become heavily dependent on them to perform various tasks. Software is the key component to drive a computer to perform its functions. Like other products, a software product is developed through several stages called the software development life cycle which is starting from the initial requirements acquisition to the retirement of the software.

A few years back, the focus of software development was on software functionality rather than security due to fewer risks involved and minimum interaction of the system with the outside world. Nowadays, the systems are no longer isolated. They have to communicate with other systems or environments through different modes of communication. The wide usage of computing environments in today's expanding business world resulted in exposure to newer security risks. Software is becoming more vulnerable due to the increase in complexity, connectivity, and extensibility. This highlights security as a constraint on software development, and it should be addressed properly to mitigate the security risks. In the case of real-time critical systems, these vulnerabilities can result in fatal consequences [1].

Software with a bad design having security flaws is more vulnerable and external protection systems, such as firewalls, intrusion detection, and malware detection, are unable to protect it from external threats. Security is not a requirement, but rather a constraint that affects the quality of the software. It can be ensured at later stages of development in the form of add-ons or some security features. These add-ons and security features do not overcome possible vulnerabilities and hence cannot protect the whole software. It

is pertinent to note that system security is different than software security. Software is a part of a computer system and should have its own security features. These features can be embedded in software during software development life cycle (SDLC). By integrating security into SDLC, we may develop software to proactively defend against potential security threats [2].

The security of the software has become more critical due to the development of smart environments [3]. The deployment of Internet-of-Things (IoT) and subsequent development of Artificial Intelligence of Things (AIoT) in diverse domains including smart health care, smart transportation, smart homes, smart communities, and smart education has elevated security risks [4, 5]. Software in IoT environments interacts with remote services using smart devices [6]. Furthermore, the software running on smart devices can access sensors, personal information, and use machine-learning techniques to learn situations [7, 8]. These IoT and AIoT software constantly monitor sensor information, location, health information, and system information that can be subject to both active and passive security attacks [9, 10]. The information is represented and stored in a common open standard to achieve interoperability among different types of systems [11]. The smart devices not only contain personal information but also keep on monitoring system data including sensor information, power information, location information, and activity information [12]. While smart devices can be made secure and personal and system data can be protected, the software that needs to run on these devices should be developed securely. From information gathering to storage, the data should be kept confidential and shared with trusted parties only [13, 14]. The information storage can also be used as history information which is useful for facilitating a user but is also a security risk [13]. Many researchers have proposed using machine learning for anomaly detection which is only effective once the software is deployed [15]. It is necessary to use secure practices in software development life cycle (SDLC) to ensure better protection once the software is deployed. Microsoft promotes Data Science as the game changer in secure SDLC [16].

It is necessary to secure a system against security risks. However, developing secure software can mitigate such risks. The contributions of this paper are as follows:

(i) This paper aims to highlight the significance of security throughout the SDLC and present a security policy framework PDWM for secure software development using the waterfall method.

(ii) Furthermore, the authors establish how secure development can lead to software that can detect malware and anomalies. Such software can be developed for smart environments where security risks are high.

This paper is organized as follows. Section 2 presents the related work, Section 3 outlines the security policies, and in Section 4, a policy-driven waterfall model for secure software development is proposed. Section 5 describes a case study, and Section 6 summarizes the paper.

2. Related Works

The related works have been searched in major research databases by using terms including SDLC, SSDLC, Secure Software, and Secure Software Development. The primary search sources have been searched using Google Scholar, Institute of Electrical and Electronics Engineers, ACM, Hindawi, Wiley, MDPI, and others. The results have been sifted to check the relevancy for secure software development processes. Relevancy has been established to compare the effectiveness of PDWN with other techniques in literature.

There are two high-profile processes available for the development of secure software, namely, Microsoft's SDL and McGraw's Touchpoints [17, 18]. SDL is rigorous and heavyweight and is more suitable for large organizations. The Touchpoints approach is based on industrial experience. Both processes have defined activities for different phases of secure software development. Some activities are similar, and some differ in both methodologies due to the priority given to certain activities in each process and are more extensive and heavyweight.

Tariq et al. have listed challenges for mission-critical systems that interact with multiple data sources in real-time, remotely [19]. Such systems utilize the concepts of IoT, AIoT, and Industrial IoT (IIoT). The challenges highlighted by the authors can be mitigated by using secure software development frameworks. This concept is enhanced by the authors by utilizing a code-driven trust mechanism for the detection of internal attacks in IoT [20]. Butt et al. have demonstrated the effect of unsecured development practices in IoT-based mobile health (mHealth) environments [21]. The authors have proposed an algorithm to mitigate the effects; however, they have not improved the development process of mHealth applications.

Ahmad and Rana have developed a secure architecture for developing e-commerce websites as a software product [22]. These websites are developed as a Service-Oriented Architecture (SOA) which needs to be made secure as it uses the Internet for communication. The authors propose embedding the existing techniques to mitigate security threats including unauthorized access, phishing, and interference with key data, within the design of the software. However, the authors have not developed a secure software development model for developing e-commerce applications.

Khan et al. have carried out an extensive survey for secure software development and identified that the security risks must be identified and addressed for successful secure software production [23]. Henry argues that organizations should establish secure coding principles to ensure a secure software development life cycle (SSDLC) [24].

Dissanayake et al. have conducted a survey of secure software patch management techniques and tools [25]. Each software goes through a maintenance and upgrade phase during its life cycle. To improve the performance, including security measures, patches are developed and applied to different copies of the software deployed across networks. It is much like the patches of Operating Systems as well as

Software Suites and apps. All these patches need to be managed effectively to secure deployment and upgrade.

The emergence of Development and Operations (DevOps) as a SDLC method has led to faster delivery of complex software products focusing on higher quality [26, 27]. Rajapaske et al. enhanced the concept and integrated concepts of Security in DevOps [28]. The new concept is called Development, Security, and Operations (DevSecOps) and comes with new security challenges for complex software development. DevSecOps is suitable for complex software development and requires a strong development and engineering team. This can be underutilized for in-house simple software development, where traditional [29] modeling is suitable.

Jøsang et al. proposed strengthening of security curriculum in education sectors so that it leads to secure software development [30]. This area receives less attention and a focus on security education can broaden the horizon of software developers, engineers, and architects. Angulo et al. have also emphasized the integration of secure practices including secure programming, threat modeling, and risk assessment in Computer Science and Software Engineering (CS/SE) curricula [29].

Almufareh and Humayun have established a concept of mediating Security and Performance (SAP) verification to improve the safety as well as the security of software [31]. The authors have identified several mediating factors that influence SAP verification.

Mbaaka has identified human as a critical security risk and used STRIDE to identify human factors, including gender, age, and education, to assess the security threats [32].

The related works show that there is a requirement for establishing security policies and a framework to implement these policies for software development using traditional methods. This is evident from Table 1 that compares the related works with PDWM.

PDWM presents a lightweight approach that is more suitable for small and medium organizations that develop medium-sized software products and have higher experience in developing similar products. PDWM outlines the security requirements in depth and describes the existing practices that are limited to specific phases of development. The model provides important insights into the security aspects in different phases of SDLC and adhering to these practices can lead to the development of a secure software system. By adopting PDWM, an organization can reduce the number of vulnerabilities in its software and hence make it more reliable and increase its security.

Like the traditional waterfall model, the PDWM emphasizes documentation which is a critical part of a software project. Besides documentation, a knowledge base can be built to record all facts that will be useful in other development projects, and with time as the knowledge base becomes more mature, the quality, efficiency, reliability, and security of products will improve.

This paper provides a new perspective for the integration of security in SDLC by suggesting a policy-driven waterfall model for secure software development. The work contributes to the understanding of secure software development. For software engineers and practitioners, this will be a more useful source for understanding security throughout the software life cycle, and software developers will be aided with some useful practices to prevent errors during the implementation phase.

3. Security Policy Framework

A policy can be defined as a definite plan or course of action adopted for the sake of expediency, facility, etc. [33]. From a business point of view, a security policy defines how a company plans to protect its physical and information technology assets from potential threats. This policy document is continuously updated to reflect new requirements stemming from the environment. A company's security policy may include an adequate use policy, a training plan to educate employees about how to protect the company's assets, how to enforce the security measures and an auditing mechanism to evaluate the effectiveness of the security policy [34, 35].

In the context of a secure software development approach, the security policy can be defined as a guideline to aid system designers and developers in analyzing and implementing security features throughout the development process. We propose a security policy framework as shown in Figure 1. The security policy framework is developed to address the issue of software security. Its main purpose is to outline the main security requirements that should be taken into account and present a policy-driven waterfall model that addresses these requirements. Organizations that lack security policy will require a lot of effort to introduce the same into their environments as it is difficult to change the habits of employees, especially development staff.

The policy is written in a general way because it is not amended frequently and is reviewed after a certain period. For example, the security policy provides instructions to check buffer overflow vulnerability but does not mention any specific tools for that. As for each project, there might be a different requirement for tools, so developers should select appropriate tools for that purpose.

Activities in the security policy framework define roles and responsibilities for administrators and developers, managing secure software development, rules, and regulations, training guidelines, etc. Audits can be conducted to ensure that policy guidelines are properly followed. The teams should be educated well to implement the security policy framework in true letter and spirit. Designers should have a deep understanding of the security policy framework in order to achieve the goals and for the better transformation of security requirements into features. Developers will get benefit from the security policy, and it will be easy for them to integrate security policy into the system [36]. The security policy framework should cover the following areas [37–39]:

(i) System architecture
(ii) Roles and responsibilities
(iii) Risk management

Table 1: Comparison of related works.

Related works	Secure practices during the development process	Lightweight approach	In-depth security requirements
[17, 18]	✓	✗	✗
[20]	✓	✗	✗
[21]	✗	✗	✓
[22]	✗	✗	✓
[26–29]	✓	✗	✓
[31]	✓	✗	✗
PDWM	✓	✓	✓

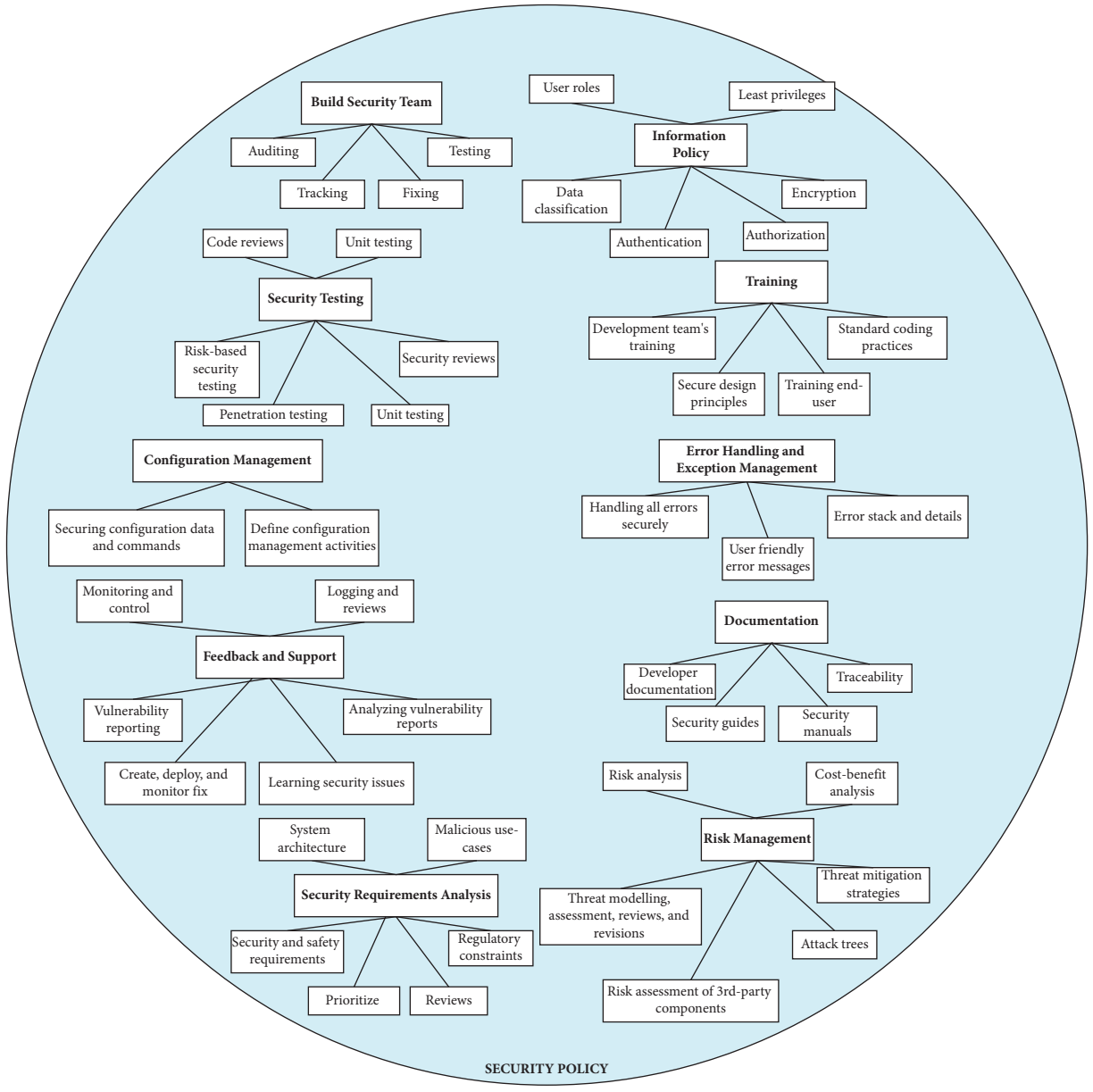

Figure 1: A security policy framework.

(iv) Security control mechanism
(v) Regulatory requirements
(vi) Information privacy
(vii) Security-related measures in development procedures
(viii) Testing and documentation
(ix) Auditing mechanism for assurance of policy implementation
(x) Training of developers to adopt best coding practices and usage of various tools
(xi) End user training regarding security features of the system
(xii) Logging and monitoring mechanisms
(xiii) Change management

The main requirements that the security policy framework should take into account are stated.

3.1. Build Security Team. A central security team comprised of security professionals will be an asset to the organization. This team helps the development people by defining process requirements, educating them to adhere to best coding practices, and performing design and code reviews [40]. Clearly define the roles and responsibilities of each team member to have a well-structured approach. An additional task of auditing may be assigned to a small group within the security team. This group will ensure that the security policy framework is implemented within the organization and is followed accordingly.

Provide suitable training to the employees regarding the security policy framework. Surprise checks are helpful to verify that employees have a well understanding of the security policy framework and are performing their roles following the guidelines provided to them. A major contribution of the security team is security testing. Test plans based on threat models and attack patterns provide the basis for security testing. Defects found are analyzed, prioritized, and fixed with the help of design and development teams.

3.2. Security Requirements. Security requirements covering all possible aspects of software's security is a key to the development of secure software. Threat modeling is used to further elaborate the security requirements for better understating and to transform them into implementation details. Functional security requirements should also be defined. Threats should be prioritized to address the highest-rated threat. Security requirements may impose some constraints to comply with regulatory standards. For example, in the health care system, patient information should not be disclosed without taking the consent of the patient. So, it is a major security concern and many agencies have devised rules and regulations for that. A final review may be conducted to assure that there is no uncaptured security requirement.

3.3. Risk Management. Risk management is an important factor in the design of secure software. After evaluating the security requirements, an analysis of anticipated risks is carried out. Threat modeling is used to uncover the threats. Threats are rated so that high-rated threats are given preference. Attack trees are helpful in modeling security threats in a graphical form. In attack trees, the attacker's goal is shown as nodes and branches represent different paths to achieve that goal. Once threats are identified, provide possible solutions to mitigate the threats. Also, carry out a cost-benefit analysis during the risk management phase. Risk management also includes security risks and uses threat modeling to assess the vulnerabilities [41, 42].

3.4. Documentation. Documentation is very essential in every project. All artifacts related to the project are recorded that are used as guidelines during the project life cycle. Apart from the design documents, other documents, such as the administrator manual, user manual, and security, guide for the user should also be developed. This will ease the work of the administrator of the system and will help users to better understand the system and its features.

3.5. Information Privacy. The next component of the security policy framework is information privacy. The company's information policy can be used to draw data classification schema. It should clearly state which user has access to which part of the data. User roles must be defined according to the information privacy policy. Security features like authentication and authorization for user access control should be used. Sensitive data should be processed and transmitted in encrypted form. Users should be granted minimum privileges that are necessary to execute a task.

3.6. Training. Another important aspect of the security policy framework is training. Training is the process to educate people to improve their performance by enhancing their skills. It is also useful for the adoption of new technologies or learning about new developments. Many security vulnerabilities arise from bad coding practices. To overcome this problem, the development team should be given training about standard coding practices and writing secure code [43]. Training material should include case studies and examples that will benefit the development teams in better understanding of learning material and its impact will last longer. Furthermore, some exercises may be included to evaluate the result of the training session. Training of system users is also essential in order to use the system in an efficient and secure manner.

3.7. Error Handling and Exception Management. Error handling is a way to detect system errors and handle them in such a way that system's normal behavior is not affected. By performing validation of data elements during input, output, or processing, errors can be minimized. Use suitable and

user-friendly error messages that can help the user to understand the cause of the error. Moreover, use exception handling mechanisms to capture exceptions that can disrupt the normal operation of the system.

3.8. Security Testing. The security testing component of the security policy framework describes certain techniques that are employed in the security testing phase. This includes risk-based security testing, security features testing, unit testing, penetration testing, code reviews, and security review.

3.9. Configuration Management. Software in its life goes through several changes that are stemmed from its environment with the growing business needs. Configuration management is the process to control and track changes [44, 45]. A configuration mechanism is necessary to keep a history of changes, review, and impact of changes incorporated. Security in configuration management deals with access control, confidentiality, accountability, and auditing. It ensures that only authorized persons have access to configuration items. All actions are logged in a way to track who made what changes at any given time. It also provides a way to review developer actions.

3.10. Feedback and Support. After the deployment of the software, the feedback mechanism can provide us with information regarding the operation of the software. This feedback mechanism can base on monitoring and logging. The analysis of logging information can reveal vulnerabilities, information about attacks, and any unexpected error that occurred during operation. Periodic visits are also helpful to monitor the behavior of the system. Any vulnerability noted is to be analyzed and adopt measures to fix the same.

4. A Policy-Driven Waterfall Model for Secure Software Development

A policy-driven waterfall model (PDWM) is derived from the traditional waterfall model as shown in Figure 2. The PDWM is based on the security policy framework. Its purpose is to integrate security in all phases of software development in order to develop secure software.

The method is linear and sequential and each phase has distinct goals. Before moving to the next phase, it is ensured that the earlier phases are correct. Each phase proceeds in strict order without overlapping. There are feedback loops present between each phase to accommodate changes. Upon discovering new artifacts or some defects, we can go back to the previous phase and incorporate the change. Like the traditional waterfall model, the PDWM emphasizes documentation which is an essential part of a project.

The mechanism starts by identifying and analysing security issues including SQL Injection, outdated software, patch requirements, security risks, data encryption requirements, DDoS awareness, unsecure coding practices, and insecure testing. These and other issues are then included in each phase for which it is appropriate and security requirements are addressed. PDWM is designed in a way that addresses security issues in each phase of SDLC.

After developing the security policy framework, it would be easier to transform security features into a development process. PDWM exhibits security aspects throughout the whole software development process. The output of each phase is provided to the input of the next phase. Every phase may produce new artifacts that will be incorporated into the security policy framework. A knowledge base can also be developed to record the security policy framework which will be useful in other development projects.

4.1. Security Analysis. During the security analysis phase, the operational environment of software is analyzed in detail concerning security aspects. The most important is the security of information. Further study may include intended users of the system, operating system, and underlying hardware. In addition, network infrastructure, communication channels, firewalls, intrusion detection systems, and software including antivirus, antispam, and antispyware should be analyzed in terms of strengths and weaknesses.

A detailed study of the software's operational environmental factors will help to avoid any vulnerability that, if left unattended, may be propagated in the next phases. Brainstorming sessions conducted with all stakeholders, including decision-makers, security policymakers, and information security specialists, are productive in the evaluation of potential threats. It is necessary to perform a threat analysis to identify assets, potential risks to those assets, possible attackers, and how to safeguard those assets from attacks. Furthermore, assets are prioritized based on confidentiality, integrity, and availability to safeguard more valued assets. Threats are rated and prioritized to take countermeasures against high-rated threats.

The possible system's security environment should be carefully analyzed which may include the type of security protection available in the underlying operating system, memory management in the operating system, user policies, the organization's information security policy, user privileges to access information, and what type of information a user can access? The cost-benefit analysis should be carried out keeping in mind the time, budget, and resource constraints.

4.2. Security Requirements. The most overlooked part of security engineering is the security requirement. These are often considered technical issues and are taken into consideration at the implementation stage. Security requirements must be stated in detail in this phase because any uncaptured requirement will be propagated into the next phases of development, consequently leaving flaws in the system that could be exploited as vulnerabilities. Collecting and analyzing the right set of security requirements and performing threat analysis is helpful in the identification of suitable security requirements and mitigating vital threats [46]. A lightweight approach consisting of well-balanced

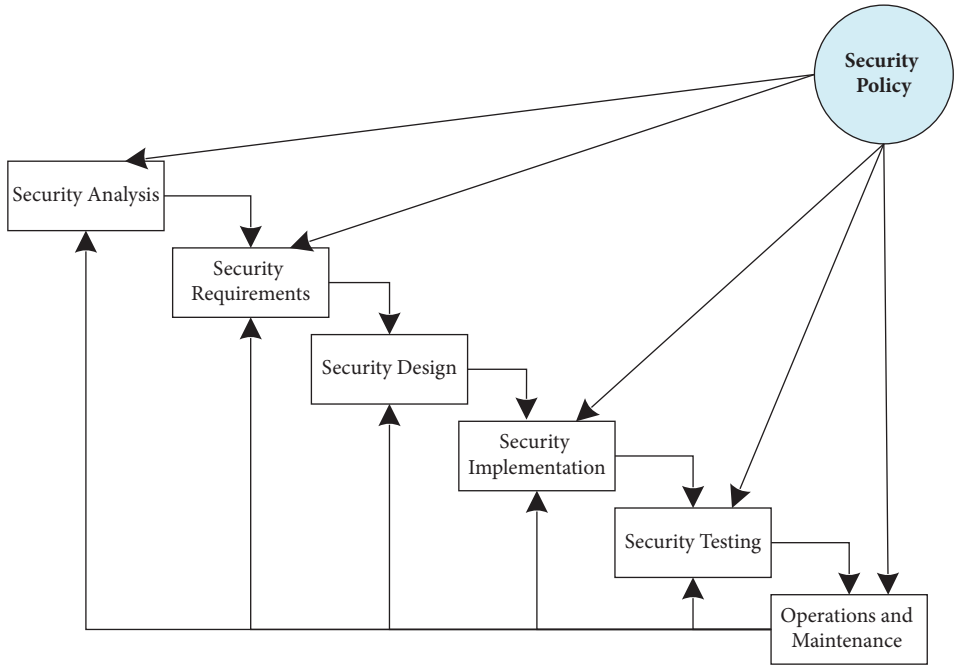

Figure 2: A policy-driven waterfall model for secure software development.

security requirements right from the beginning is very useful to elicit critical security requirements [47].

One best technique is to define misuse cases for possible threats [48]. Peterson and Steven have presented an approach to defining misuse cases [49]. Brainstorming sessions of information security professionals and developers may be productive to discover misuse cases [50]. Using misuse cases, one can define the attacker's goal or ways to exploit the system. Misuse cases may lead to additional nonfunctional and quality requirements that should be documented and included in existing requirements. Knowledge of security analysts is of great value while performing business risk analysis and architectural risk analysis.

Threat modeling helps identify risks and subsequently takes decisions to mitigate those risks in the design, coding, and testing phases [51]. To further elaborate the threat models, attack trees are used. Attack trees allow us to model security threats in a graphical form. It has been observed that attack trees are more effective for finding threats in the absence of use-case diagrams [52]. The graphical representation of the attack tree provides a better understanding of how attacks can be successful and the probability of attacks that are most likely to succeed [53, 54]. The methodology can also reveal the vulnerability of a system, under specified constraints. If we understand the ways in which a system can be attacked, we can develop countermeasures to prevent those attacks.

4.3. Security Design. At the design level, the security framework outlined in the requirements phase must be evaluated in terms of technology and the system must present a unified structure that can be implemented. The designers must review the design keeping in view the security requirements which will help to identify additional risks or threats. Software security is categorized into four areas, namely, input, output, data, and algorithm. They must be made secure [55]. Evaluation of underlying technologies in terms of implementation of the design is crucial. Alternate solutions may be considered to pick the best one that is more secure, efficient, and cost-effective. If the system under development is of classified nature, there may be a requirement to secure the design of the system.

By following the secure design principles, the secure development process can be improved. Sometimes, there is a requirement that some portion of the application's code may be open to the Internet. In this case, the potential risks and what has been vulnerable must be analyzed. If the application consumes untrusted data, enforce a validation mechanism that must be robust in data handling [56]. The design of software can be made more secure by the use of attack patterns that can identify security vulnerabilities at an initial stage. UMLsec can be used to model the security aspects of the system [57]. Once security flaws are identified, designers should adopt appropriate measures to mitigate those vulnerabilities and strengthen the defense mechanism of the system [58]. Designers should keep the design as simple as possible and enforce defense mechanisms in depth.

There are commercial off-the-shelf (COTS) components available that are used in the development of software to reduce cost and development time. As the software is developed and deployed on some operating systems, we have to use some APIs of the operating system for communications or other services. A detailed study of these APIs will help in a better understanding of their structures and implementation. Bad implementation of APIs can lead to vulnerabilities that may pose a risk to the system. Furthermore, if third-party components have been used in the development of the system,

obtain complete documentation to get complete knowledge of software components. Developing own encryption algorithm is not an easy job because it requires deep knowledge of encryption techniques. The use of standard encryption algorithms is a good approach to a secure design. Encryption of passwords or sensitive data is essential to make it secure before transmitting over the network. The principle of least privilege must be used for a user to perform a task. The design needs to be consistent and race conditions should not exist. Data objects need to be defined with a lower bound and upper bound limit.

4.4. Security Implementation. Implementation begins with the selection of the appropriate programming language. Various programming languages are available today with different features. C/C++ are quite popular and flexible languages but are criticized due to security vulnerabilities. However, secure coding can be achieved by proper handling of data holders. To overcome common errors regarding string and integer manipulation in C/C++, alternate solutions are available [59]. The most common vulnerabilities arising from coding problems in C language are buffer overflow, format string vulnerabilities, and integer vulnerabilities [60].

Static analysis tools can be used to detect flaws in code. Although these tools provide help to developers to discover coding errors, their scopes are limited and do not guarantee defect-free software. Code review can be done with the help of tools, but it is recommended that one should not fully depend on these tools. A manual review should also be performed, which is quite productive. Source code review checklists provide a good way to minimize errors.

Common security flaws can be removed by cryptography and with improved quality procedures [61]. Programmers should define passwords with alphanumeric combinations to make them strong enough, with suitable length and expiration periods. All inputs and outputs are to be analyzed and validated. The length and type of input fields must be clearly defined. It is necessary to implement typecasting carefully and properly and destroy memory objects after use for better memory management. Validation of function calls and parameter passing like pass-by-value or pass-by-reference needs to be performed. Sensitive data including user authentication should be transmitted in encrypted form. By following best coding practices, errors like buffer overflow, stack overflow, type mismatch, and divide by zero can be avoided. All exceptions must be handled with a try-catch block and use suitable error messages for user information. For debugging and auditing purposes, logs can be generated. The security of the logging mechanism has to be handled properly.

4.5. Security Testing. Testing is a crucial part of the software development life cycle. Security testing is the process to determine that the application is securing data and performing its intended functionality. Parameters including authentication, authorization, confidentiality, integrity, and nonrepudiation should be kept in mind for security testing. The security team performs various tests to check the behavior of the system under possible attacks. Dynamic software security testing is useful for the system developed using multisource components [62]. Although it would be quite hard to develop such a security testing system at the initial stages, later on, it will be more helpful for the development of secure systems. A security test plan comprising security functionality and risk-based security testing is useful in the validation of security aspects and identification of security defects [63].

Code review is a time-consuming process but produces good results. The quality of the review depends upon the reviewer's competency and professionalism. Code review can produce better results for security testing [64]. Another technique in security testing is the use of checklists. Checklists are used to verify specific measures needed for software security [65]. Test cases can be generated from misuse cases to validate the defense mechanism against an attack. A test team plays a vital role in the testing phase. Penetration testing is very useful to identify potential vulnerabilities. Penetration testing applied at the unit and system level can improve the software development life cycle [66]. Fuzz testing is also very helpful to discover software defects. Testers should employ themselves as hackers of the system to perform testing to evaluate the system for potential vulnerabilities, bugs, or flaws.

4.6. Operation and Maintenance. Mostly operation and maintenance phases are not considered in the security framework. However, it is as important as other phases of the software development life cycle. Proper deployment and configuration of the system can ease the work of system administrators. Furthermore, to keep the system updated against security threats, constant updating and monitoring are required. During the operation of software, monitoring the software for security breach attempts is helpful to analyze and remove the defects. A response process may be adopted to evaluate vulnerabilities and respond to these by releasing an update and removing other defects.

Deploying the application safely in its intended environment and running it accordingly will have a positive effect on information security, and monitoring mechanisms help in incident response operations. Other techniques, such as code isolation, protection of executables, and monitoring programs for executables, can be used to safeguard the system from environmental threats. The introduction of a feedback mechanism is very helpful for continuous improvements and updating of the system. For tracking activities, logging must be used to analyze the attacks or vulnerabilities so that a countermeasure action can be taken and implemented into the system.

5. Case Study

We use an e-travel system, an online flight ticketing application, to exemplify PDWM. Figure 3 shows the scenario of e-travel that includes inquiries about flights and makes online bookings.

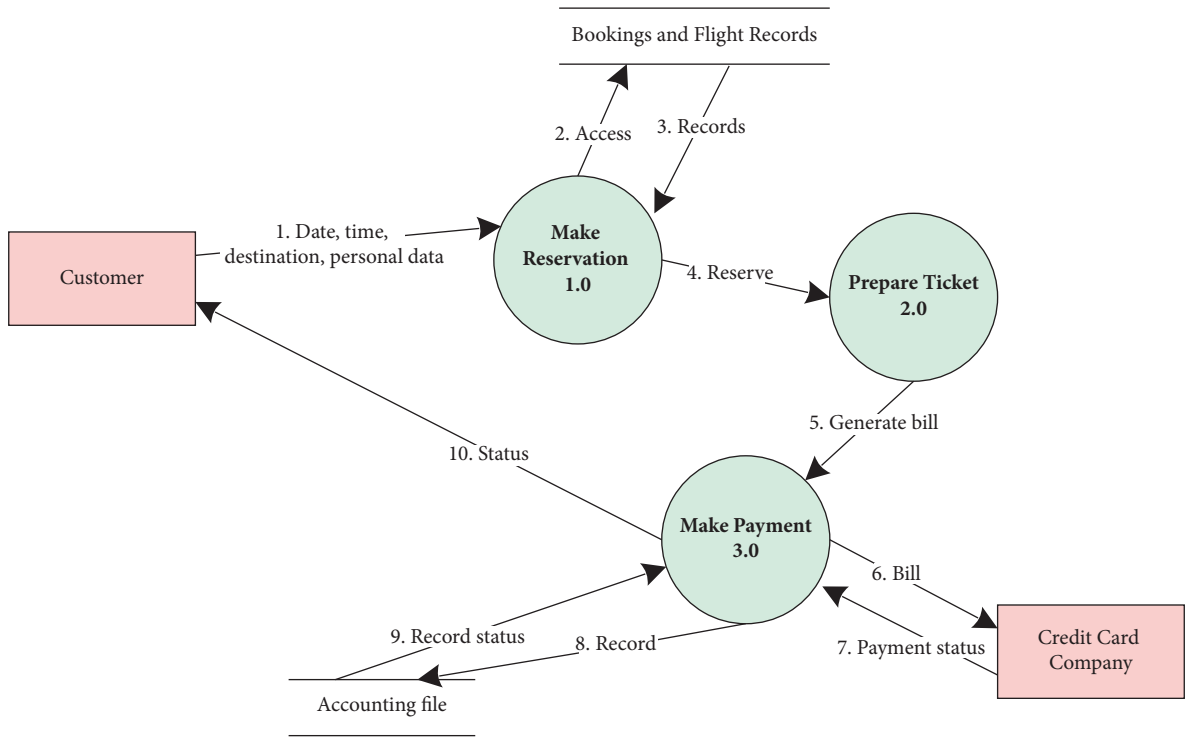

Figure 3: e-travel scenario.

Flight inquiries can be made by any customer as there is no need for the provision of personal data. However, in the case of a complete transaction, that is from reservation to printing of e-ticket, customer information is required which should be kept confidential and hence raises the requirement to make the process secure. We will focus on this scenario for the applicability of the PDWM.

For the e-travel application, we must analyze the environment to identify threats because the environment of the application is also a major factor that influences possible threats. As e-travel is a client-server application communicating over the Internet, attackers can misuse the system to collect customers' data like the credit card number. So, the main security requirements are to secure the customer's information as well as data transmission over the Internet. These two requirements are the security policies that must be adhered to when developing this application. e-travel is exposed to eavesdropping (information disclosure) threats as well [67].

Now, we can define the security requirements for e-travel application. There are two security requirements that are to be addressed: one is transmitting customer information in a secure way and the other is the e-travel's database privacy as it contains valuable data on customer bookings.

The attack tree for obtaining customer information is depicted in Figure 4. "Obtain customer's information" is the root node of the attack tree. Branches represent different paths that an attacker can follow to achieve that goal. OR-nodes represent alternative paths while AND-nodes are subgoals that must be satisfied to accomplish an attack. "Looking over the shoulder" and "Threaten" attacks were deleted from the attack tree as they are related to physical security.

When a customer is interested in buying a ticket, then the customer's credentials will be transmitted in an encrypted form. 256-bit AES can be used to encrypt the information on the client side and then transmit it over the communication channel. In addition to this, auto-generated session keys with expiration duration can be generated to enhance the security features of the communication. On the server side, information will be decrypted and processed accordingly.

Several programming languages can be used for the implementation of e-travel. For e-travel, we selected PHP as it is open source and suitable for our sample application and for the database MySQL (CE).

256-bit AES is used to encrypt the customer's information on the client side using JavaScript before transmitting it over the Internet to the server. This will ensure the confidentiality of data over a public network. Validation of all inputs will be done for type and length. Typecasting if not implemented properly can produce errors. To avoid buffer overflow errors, the buffer size will be fixed, so that it can be checked before usage.

Here are two PHP coding examples for reference. One is the validation of input for alphanumeric and the other is relevant to limiting the length of the text string.

```
//Check if string contains characters other than alphanumeric
$alphaNum = "1234567teststring@#&-]";
```

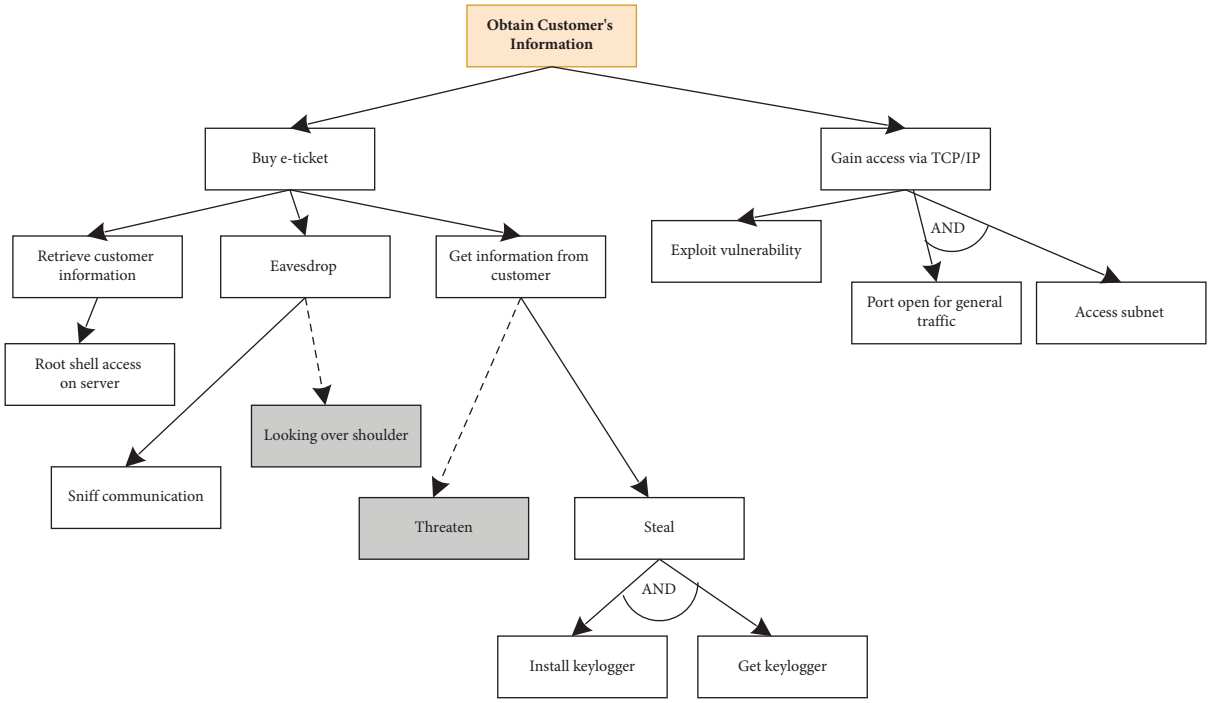

Figure 4: Attack tree for obtaining customer information.

```
if (ereg('[A-Za-z0-9]', $alphaNum))
{
   echo "Text contains characters other than alphanumeric";
}
else
{
   echo "Text contains only letters and numbers";
}
//Limit the length of string to 20 characters
if (strlen($sample) > "20")
{
   exit("Text is too long, maximum length is 20.").
}
```

We can protect our scripts from SQL injection by using a built-in PHP function. This function removes special characters from the string. An example is shown.

```
//Clean the string before passing to SQL query
$userName = mysql_real_escape_string($_POST ['user_name'], $mydb).
$queryResult = mysql_query("INSERT INTO User (name) VALUE ('{$userName}')").
if ($queryResult)
{
   echo "User name stored successfully.";
}
else
{
   echo "User name cannot be stored, please try again.";
}
```

Security testing of e-travel can be performed through web application security testing tools. Various commercial and open-source tools are available for web security testing that can be used to track vulnerabilities. Two web security testing tools, Websecurify and skipfish, are used for the testing of e-travel [68, 69]. Both tools are open source, support multiplatform, are easy to use, have cutting-edge web technologies, and with low false positive rate. Any detected bugs in this stage should be rated so that bugs with a higher priority should be removed in the first place. A port scanner tool can be used to scan ports on a web server. Any unused open ports should immediately be closed or disabled to avoid any vulnerability or attack. e-travel system generates logs for thrown exceptions. These logs will be analyzed for errors and potential attacks and remedial measures will be taken to secure the system. Table 2 shows the security vulnerabilities and their potential risks for the e-travel scenario based on STRIDE.

Table 1 shows that the critical areas of focus are the boundary processes and databases. It is necessary for the development team to explore security requirements for these areas in all phases of SDLC. It would be necessary to write functional requirements as well as quantify nonfunctional requirements during the requirements phase of development. The requirements should be modifiable and traceable as new requirements emerge. During the design phase, the security aspects should be included in the design, such as encryption, session identifiers, authentication, authorization, lease, and checksums. Furthermore, access lists could

Table 2: Security vulnerabilities and risks for e-travel.

S. No.	Component	Type	Security vulnerability	Risk
1	Make reservation	Process	Tampering, spoofing, repudiation, information disclosure, DoS, elevated access	High
2	Prepare ticket	Process		Low
3	Make payment	Process	Spoofing, repudiation	Moderate
4	Bookings and flight records	Database	Information disclosure, elevated access	Moderate
5	Accounting file	Datastore	Information disclosure, elevated access	Moderate

Table 3: Comparison of security test cases between PDWM and other techniques.

S. No.	Component	Number of security test cases	Number of security test cases identified during software development			
			PDWM	Secure software patch management	Using SOA	Traditional waterfall development
1	Make reservation	6	6	—	6	—
2	Prepare ticket	—	—	—	—	—
3	Make payment	2	2	—	2	—
4	Bookings and flight records	2	2	—	—	—
5	Accounting file	2	2	—	—	—
	Total	**12**	**12**	—	**8**	—

This signifies the total number of security text cases.

Table 4: Limitations and their responses in PDWM.

Limitation	Response in PDWM
Serial execution	PDWM is developed much more on the lines of the waterfall method of SDLC with a serial execution. This ensures a stable development model useful for an experienced team
Rigidity	Since PDWM is developed sequentially, there is little room for error. However, the inclusion of security vulnerabilities and their solution in each phase reduces the risks. However, a risk management phase can further reduce the need for change once a phase is completed
Change handling	A change that occurs after a phase is completed can be handled as there are feedback loops in PDWM; however, the cost is higher than agile techniques. PDWM suffers from cost vs security tradeoff; while the security handling is enhanced, the cost of change handling cannot be reduced. This is also true for late discovery of requirements that could lead to newer security vulnerabilities
Flexibility	PDWM is not a flexible model like agile techniques. However, when considering security vulnerabilities, flexibility is a desired feature. PDWM addresses flexibility by employing experienced team and enlisting all security requirements for all stages of SDLC exhaustively
Delayed feedback	PDWM prioritizes security over all other requirements. It is envisaged that the user involvement should increase in each phase to reduce delayed feedback by the user
Exhaustive requirement gathering	Since the team is composed of experienced members, it is impossible to exhaustively gather all requirements during the requirements phase. However, change is still possible that can be handled using feedback loop at a higher cost
Large projects	While PDWM is not suitable for large projects, a component-based approach can be utilized in which multiple teams develop components using PDWM
Risk management	There is no risk management phase explicitly embedded in the model; however, risks of each security vulnerability are considered, and security requirements are generated for each phase

ensure that there is no unnecessary elevation of privileges. Logs should be maintained to ensure security audits. Based on the requirements and the design, suitable test cases should be developed to ensure the effectiveness of the security countermeasures.

When compared with the traditional waterfall model, the security aspects would emerge once the product is deployed, causing errors, and exceptions, which could potentially destroy a business by increasing the cost of the fix. Using PDWM could ensure that security policies are embedded in all phases so that appropriate countermeasures could be included in the design and implementation.

We have further compared the effectiveness of PDWM by developing security test cases. These security test cases are developed for the case given in Figure 3. There are a total of 12 security test cases for each security vulnerability of the components of the e-travel scenario as shown in Table 2. Compared with the techniques for secure software development listed in Section 2, Table 3 shows that PDWM can identify security vulnerabilities at an earlier stage of development. It can be seen that PDWM can identify 33% more security vulnerabilities when using SOA.

6. Conclusions

This paper presents PDWM, which uses security policies in software development. The security policies are embedded in each phase of waterfall-based software development. The security policies include the security-related requirements that must be considered during SDLC. A framework that supports the security policies is given in this paper. This framework is applied to an e-travel case study to ascertain its effectiveness.

PDWM embodies best practices in each phase of software development starting from requirements up to maintenance. These best practices help system analysts and

developers to develop secure software products. Adhering to these practices can result in a secure, reliable, and efficient system that can proactively defend against security threats, especially when it comes to developing software for smart environments. PDWM is limited to the secure development of medium-sized and low-risk software products having stable requirements. There is a need to explore the effectiveness of PDWN for developing high-risk software products that have dynamically changing requirements, using agile methods. The authors list some limitations and how PDWM addresses them in Table 4.

The limitations presented in Table 4 can be used as future work. While PDWM lacks the advantages of the iterative development model, a search for a policy driven agile development goes on.

Data Availability

The authors confirm that the data generated or analyzed and supporting the findings of this study are available within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Zhivich and R. K. Cunningham, "The real cost of software errors," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 7, no. 2, pp. 87–90, 2009.

[2] G. McGraw, "Building secure software: better than protecting bad software," *Institute of Electrical and Electronics Engineers Software*, vol. 19, no. 6, pp. 57-58, 2002.

[3] U. Mahmud, S. Hussain, A. J. Malik, S. Farooqui, and N. A. Malik, "Realizing IoE for smart service delivery: case of museum tour guide," in *Smart Systems Design, Applications, and Challenges*, IGI Global, Hershey, PA, USA, 2020.

[4] U. Mahmud, S. Hussain, A. Sarwar, and I. K. Toure, "A distributed emergency vehicle transit system using artificial intelligence of Things (DEVeTS-AIoT)," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 9654858, 12 pages, 2022.

[5] S. Hussain, U. Mahmud, and S. Yang, "Car e-talk: an IoT-enabled cloud-assisted smart fleet maintenance system," *Institute of Electrical and Electronics Engineers Internet of Things Journal*, vol. 8, no. 12, pp. 9484–9494, 2020.

[6] U. Mahmud, S. Hussain, and I. K. Toure, "Gathering contextual data with power information using smartphones in internet of everything," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 4445751, 14 pages, 2022.

[7] U. Mahmud and M. Y. Javed, "Context inference engine (CiE): inferring context," *International Journal of Advanced Pervasive and Ubiquitous Computing*, vol. 4, no. 3, pp. 13–41, 2012.

[8] U. Mahmud and M. Y. Javed, "Context inference engine (CiE): classifying activity of context using minkowski distance and standard deviation-based ranks," in *Systems and Software Development, Modeling, and Analysis: New Perspectives and Methodologies*, pp. 65–112, IGI Global, Hershey, PA, USA, 2014.

[9] U. Mahmud, N. Iltaf, A. Rehman, and F. Kamran, "Context-Aware paradigm for a pervasive computing environment (CAPP)," in *WWW\Internet 2007*, Lambert Academic Publisher (LAP), Villa Real, Portugal, 2007.

[10] Y. Li and Q. Liu, "A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments," *Energy Reports*, vol. 7, pp. 8176–8186, 2021.

[11] U. Mahmud, U. Farooq, M. Y. Javed, and N. A. Malik, "Representing and organizing ContextualData in context aware environments," *Journal of Computing*, vol. 4, no. 3, pp. 61–67, 2012.

[12] U. Mahmud, S. Hussain, and S. Yang, "Power profiling of context aware systems: a contemporary analysis and framework for power conservation," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 1347967, 15 pages, 2018.

[13] N. A. Malik, M. Y. Javed, and U. Mahmud, "Estimating user preferences by managing contextual history in context aware systems," *Journal of Software*, vol. 4, no. 6, pp. 571–576, 2009.

[14] U. Mahmud, N. Iltaf, and F. Kamran, "Context congregator: gathering contextual information in CAPP," in *Proceedings of the Frontiers of Information Technology*, Islamabad, Pakistan, December 2007.

[15] D. Kim and T.-Y. Heo, "Anomaly detection with feature extraction based on machine learning using hydraulic system IoT sensor data," *Sensors*, vol. 22, no. 7, pp. 1–24, 2022.

[16] Microsoft, "Secure the software development," 2020, https://www.microsoft.com/security/blog/2020/04/16/secure-software-development-lifecycle-machine-learning/.

[17] M. Howard and S. Lipner, *The Security Development Lifecycle (SDL): A Process for Developing Demonstrably More Secure Software*, Microsoft Press, New York, NY, USA, 2006.

[18] G. McGraw, *Software Security: Building Security in*, Addison Wesley, New York, NY, USA, 2006.

[19] N. Tariq, M. Asim, and F. A. Khan, "Securing SCADA-based critical infrastructures: challenges and open issues," *Procedia Computer Science*, vol. 155, pp. 612–617, 2019.

[20] N. Tariq, M. Asim, F. A. Khan, T. Baker, U. Khalid, and A. Derhab, "A blockchain-based multi-mobile code-driven trust mechanism for detecting internal attacks in internet of Things," *Sensors*, vol. 21, no. 1, pp. 1–27, 2020.

[21] S. A. Butt, T. Jamal, M. A. Azad, A. Ali, and N. S. Safa, "A multivariant secure framework for smart mobile health application," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 8, pp. 1–18, 2019.

[22] T. Ahmed and T. Rana, "Secure architecture for E-commerce websites," *Sir Syed University Research Journal of Engineering and Technology*, vol. 9, no. 1, pp. 13–17, 2019.

[23] R. A. Khan, S. U. Khan, H. U. Khan, and M. Ilyas, "Systematic literature review on security risks and its practices in secure software development," *Institute of Electrical and Electronics Engineers Access*, vol. 10, pp. 5456–5481, 2022.

[24] O. N. Henry, "Secure software development: industrial practice- a review," *I-manager's Journal on Software Engineering*, vol. 16, no. 3, pp. 60–71, 2022.

[25] N. Dissanayake, A. Jayatilaka, M. Zahedi, and M. A. Babar, "Software security patch management- A systematic literature review of challenges, approaches, tools and practices," *Information and Software Technology*, vol. 144, Article ID 106771, 2022.

[26] A. Mishra and Z. Otaiwi, "DevOps and software quality: a systematic mapping," *Computer Science Review*, vol. 38, pp. 1–18, 2020.

[27] N. M. Noorani, A. T. Zamani, M. Alenezi, M. Shameem, and P. Singh, "Factor prioritization for effectively implementing DevOps in software development organizations: a SWOT-AHP approach," *Axioms*, vol. 11, no. 10, pp. 1–29, 2022.

[28] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, "Challenges and solutions when adopting DevSecOps: a systematic review," *Information and Software Technology*, vol. 141, Article ID 106700, 2022.

[29] A. A. Reyes Angulo, X. Yang, Q. Niyaz, S. Paheding, and A. Y. Javaid, "A secure software engineering design framework for educational purpose," in *Proceedings of the 2022 Institute Of Electrical And Electronics Engineers International Conference on Electro Information Technology (eIT)*, Mankato, MN, USA, May 2022.

[30] A. Jøsang, M. Ødegaard, and E. Oftedal, "Cybersecurity through secure software development," in *Proceedings of the IFIP World Conference on Information Security Education (WISE 2015)*, Hamburg, Germany, February 2015.

[31] M. F. Almufareh and M. Humayun, "Improving the safety and security of software systems by mediating SAP verification," *Applied Sciences*, vol. 13, no. 1, p. 647, 2023.

[32] W. Mbaka, "Towards unveiling effects of human factors within security risk assessment," *Association for Computing Machinery Special Interest Group on Software Engineering-Software Engineering Notes*, vol. 48, no. 1, pp. 70–75, 2023.

[33] Dictionary.com, "Policy," 2022, http://dictionary.reference.com/browse/policy.

[34] B. Lutkevich, "What is security policy," 2022, http://searchsecurity.techtarget.com.

[35] R. Grimmick, "What is a security policy?" 2022, https://www.varonis.com/blog/what-is-a-security-policy.

[36] J. Steer, "Security policies in the application development process," 2011, http://technet.microsoft.com/en-us/library.

[37] Fordham University, "Web applicataion security," 2019, https://www.fordham.edu/information-technology/it-security--assurance/it-policies-procedures-and-guidelines/web-application-security-policy/.

[38] CertMike, "Security policy framework," 2018, https://www.certmike.com/security-policy-framework/.

[39] gov uk, "Security policy framework: protecting government assets," 2022, https://www.gov.uk/government/publications/security-policy-framework.

[40] M. Howard, "Building more secure software with improved development processes," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 2, no. 6, pp. 63–65, 2004.

[41] N. A. Malik, Y. M. Javed, and U. Mahmud, "Threat modeling in pervasive computing paradigm," in *Proceedings of the 2008 New Technologies, Mobility and Security*, Marrakesh, Morocco, November 2008.

[42] U. Mahmud and N. A. Malik, "Flow and threat modelling of a context aware system," *International Journal of Advanced Pervasive and Ubiquitous Computing*, vol. 6, no. 2, pp. 58–70, 2014.

[43] J. N. Kotey, *A Functioning Code May Not Be a Secure Code: A Preliminary Study on the Students' Complacency with Secure Coding*, Montclair State University, Montclair, NJ, USA, 2023.

[44] S. S. Fauzi, P. L. Bannerman, and M. Staples, "Software configuration management in global software development: a systematic map," in *Proceedings of the 2010 Asia Pacific Software Engineering Conference*, Sydney, Australia, November 2010.

[45] N. Fareghzadeh, "Integrated approach of software configuration management," *Quarterly Journal of Information and Communication Technology*, vol. 2, no. 4, pp. 41–48, 2021.

[46] K. Beznosov and B. Chess, "Security for the rest of us: an industry perspective on the secure-software challenge," *Institute of Electrical and Electronics Engineers Software*, vol. 25, no. 1, pp. 10–12, 2008.

[47] I. A. Tondel, M. G. Jaatun, and P. H. Meland, "Security requirements for the rest of us: a survey," *Institute of Electrical and Electronics Engineers Software*, vol. 25, no. 1, pp. 20–27, 2008.

[48] G. McGraw, "Software security," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 2, no. 2, pp. 80–83, 2004.

[49] G. Peterson and J. Steven, "Defining misuse within the development process," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 4, no. 6, pp. 81–84, 2006.

[50] K. van Wyk and G. McGraw, "Bridging the gap between software development and information security," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 3, no. 5, pp. 75–79, 2005.

[51] N. Davis, W. Humphrey, S. T. Redwine Jr., G. Zibulski, and G. McGraw, "Processes for producing secure software," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 2, no. 3, pp. 18–25, 2004.

[52] A. L. Opdahl and G. Sindre, "Experimental comparison of attack trees and misuse cases for security threat identification," *Information and Software Technology*, vol. 51, no. 5, pp. 916–932, 2009.

[53] P. A. Wortman and J. Chandy, "Translation of AADL model to security attack tree (TAMSAT) to SMART evaluation of monetary security risk," *Information Security Journal: A Global Perspective*, vol. 32, no. 4, pp. 297–313, 2022.

[54] N. M. Scala, P. L. Goethals, J. Dehlinger, Y. Mezgebe, B. Jilcha, and I. Bloomquist, "Evaluating mail-based security for electoral processes using attack trees," *Risk Analysis*, vol. 42, no. 10, pp. 2327–2343, 2022.

[55] J. Whittaker, "Why secure applications are difficult to write?" *Institute of Electrical and Electronics Engineers Security and Privacy*, vol. 1, no. 2, pp. 81–83, 2003.

[56] M. Howard, "Becoming a security expert," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 6, no. 1, pp. 71–73, 2008.

[57] J. Jurgens, "UMLsec: extending UML for secure systems development," in *Proceedings of the 5th International Conference on The Unified Modeling Language (UML 2002)*, Dresden, Germany, September 2002.

[58] M. Gegick and L. Williams, "On the design of more secure software-intensive systems by use of attack patterns," *Information and Software Technology*, vol. 49, no. 4, pp. 381–397, 2007.

[59] R. Seacord, "Secure coding in C and C++ of strings and integers," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 4, no. 1, pp. 74–76, 2006.

[60] S. Mancoridis, "Software analysis for security," in *Proceedings of the Frontiers of Software Maintenance (FoSM 2008)*, NewYork, NY, USA, October 2008.

[61] B. Snow, "Four ways to improve security," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 3, no. 3, pp. 65–67, 2005.

[62] M. R. Stytz and S. B. Banks, "Dynamic software security testing," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 4, no. 3, pp. 77–79, 2006.

[63] B. Potter and G. McGraw, "Software security testing," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 2, no. 5, pp. 81–85, 2004.
[64] A. Apvrille and M. Pourzandi, "Secure software development by example," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 3, no. 4, pp. 10–17, 2005.
[65] M. Bishop and D. A. Frincke, "Teaching secure programming," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 3, no. 5, pp. 54–56, 2005.
[66] B. Arkin, S. Stender, and G. McGraw, "Software penetration testing," *Institute of Electrical and Electronics Engineers Security and Privacy Magazine*, vol. 3, no. 1, pp. 84–87, 2005.
[67] Microsoft, "Microsoft threat modeling tool threats," 2022, https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats.
[68] websecurify, "Websecurify-web application security scanner and manual penetration testing tool," 2022, https://websecurify.com/.
[69] Kali, "skipfish," 2022, https://www.kali.org/tools/skipfish/.