World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(REVIEW ARTICLE)

Check for updates

# Theory and practice in secure software development lifecycle: A comprehensive survey

Martin Otieno [1], David Odera [2,*] and Jairus Ekume Ounza [3]

[1] *Jaramogi Oginga Odinga University of Science and Technology, Bondo, Kenya.*
[2] *Tom Mboya University, Homa-Bay, Kenya.*
[3] *Kabarak University, Nakuru, Kenya.*

World Journal of Advanced Research and Reviews, 2023, 18(03), 053–078

Publication history: Received on 11 April 2023; revised on 30 May 2023; accepted on 01 June 2023

Article DOI: https://doi.org/10.30574/wjarr.2023.18.3.0944

## Abstract

Software development security refers to the practice of integrating security measures and considerations throughout the software development lifecycle to ensure the confidentiality, integrity, and availability of software systems. It involves identifying, mitigating, and eliminating security vulnerabilities and threats that could be exploited by attackers. The goal of this paper is to survey the various concepts and methodologies directed towards software security, and the identification of any missing gaps. Based on the findings, it is noted that the development of secure software requires a proactive and comprehensive approach. It begins with establishing secure design principles and incorporating security requirements from the initial stages of development. Here, secure coding practices, such as input validation, output encoding, and secure authentication and authorization mechanisms, are employed to prevent common security vulnerabilities. In addition, regular security testing, including penetration testing and vulnerability scanning, helps identify and address potential weaknesses in the software. Normally, code reviews and security audits are conducted to ensure adherence to secure coding practices and identify any security flaws. It is important that security training and awareness programs be provided to developers and other stakeholders to foster a security-conscious culture. To minimize potential vulnerabilities, secure configuration management, which involves properly configuring servers, networks, and dependencies may be utilized. On the other hand, regular updates and patching are essential to address known security vulnerabilities in software components. To guide their software development security practices, organizations may follow established security standards and frameworks such as ISO 27001 or NIST Cybersecurity Framework. By prioritizing software development security, organizations can protect sensitive data, prevent unauthorized access, and mitigate the risk of security breaches and incidents. In the long run, this helps build trust with users and stakeholders, enhances the reputation of the software, and reduces the potential impact of security incidents on the organization.

**Keywords:** Attacks; Coding; Methodologies; Privacy; Security; Software; SDLC

## 1. Introduction

Software development is the process of creating, designing, coding, testing, and maintaining software systems [1]-[3]. It involves a combination of technical skills, problem-solving abilities, and creative thinking to translate concepts and requirements into functional software applications. As explained in [4], software development plays a fundamental role in today's digital world, powering various industries, organizations, and everyday activities. The goal is to produce reliable, efficient, and user-friendly software that meets the needs of its intended users. It encompasses a broad range of activities, including understanding user requirements, designing system architectures, writing code, debugging and testing [5], and deploying software to production environments. Effective software development requires collaboration

*Corresponding author: David Odera

among teams, adherence to best practices, and a strong focus on quality assurance [6]. Basically, it is a dynamic field that continuously evolves due to advancements in technology, changing user expectations, and emerging industry trends [7]. It encompasses various methodologies and approaches, such as Agile, Waterfall, DevOps, and Lean, each with its own principles and practices for managing the software development process.

Successful software development involves a combination of technical expertise, project management skills, and effective communication. It requires developers to understand the problem domain, apply appropriate algorithms and data structures, and adhere to coding standards and best practices [8]-[10]. Collaboration and teamwork are crucial, as developers often work together with designers, testers, and stakeholders to ensure the software meets the desired objectives. Moreover, software development is not a one-time event but rather an ongoing process. After initial development, software requires regular updates, bug fixes, and enhancements to adapt to changing requirements and address evolving security concerns [11]. Maintenance and support are essential components of the software development lifecycle. Basically, software development is a dynamic and critical process that involves designing, coding, testing, and maintaining software systems. It requires a combination of technical expertise, problem-solving skills, and effective collaboration. Successful software development delivers reliable, efficient, and user-friendly software that meets the needs of its users and contributes to the advancement of technology and society as a whole [12].

Software security is a critical aspect of modern software development and usage. With the increasing reliance on software systems in various domains, the need to protect these systems from security threats has become paramount [12]-[14]. It focuses on ensuring the confidentiality, integrity, and availability of software and the data it handles. The primary objective of software security is to identify and mitigate vulnerabilities, weaknesses, and threats that could potentially be exploited by attackers [15]. It involves implementing measures, practices, and controls to protect software systems from unauthorized access, data breaches, information leakage, and other security breaches. It encompasses various layers and aspects, including secure coding practices, secure design principles, secure configurations, secure authentication and authorization mechanisms, encryption, access control, secure communication protocols, and robust security testing [16]-[18]. These measures are designed to minimize the risk of security vulnerabilities, such as injection attacks, cross-site scripting (XSS), buffer overflows, and improper data handling. In addition to technical considerations, software security also involves promoting a security-conscious culture within organizations [19]-[23]. This includes raising awareness about security best practices, providing training to developers and users, and establishing processes for security incident response and management. The authors in [24] and [25] point out that it is important to integrate security into the software development lifecycle from the early stages to address security requirements proactively.

The consequences of software security breaches can be severe, ranging from financial losses and reputational damage to legal liabilities and privacy violations. Software security is not a one-time effort but requires continuous monitoring, updating, and adapting to address emerging threats and vulnerabilities [26]-[28]. Organizations must stay vigilant and proactive in managing software security risks throughout the lifecycle of software systems. As technology evolves, so do the techniques and tactics employed by attackers. Therefore, software security is an ongoing challenge that necessitates staying up-to-date with the latest security practices, emerging vulnerabilities, and countermeasures. Collaboration, information sharing, and adherence to industry standards and best practices are essential in ensuring robust software security [29]. According to [30], software security is of utmost importance in today's interconnected world. It involves implementing security measures, adhering to secure coding practices, and maintaining a security-conscious culture to protect software systems from security threats. By prioritizing software security, organizations can safeguard their software assets, protect user data, and maintain trust in an increasingly digital landscape. In this paper, the following contributions are acclaimed:

- The software development lifecycle is examined in great depth in order to establish how it can be made secure
- The various software development approaches are explored so as to understand how they can shape the development process.
- The concept of software security is discussed and some the key rationale for software security are identified
- The predominant software security risks are established and the techniques for addressing them described
- The theory and practice of software security is discussed, including their key aspects.
- The methodologies for identifying and eliminating security vulnerabilities are illustrated, including the tools for identifying and eliminating these security vulnerabilities.
- The various techniques to prove the absence of vulnerabilities are described, including the essential guidelines for building secure software.

The following sub-sections discuss the major issues in software development process in some greater details. Towards the end of this article, research gaps are identified which need immediate attention.

## 2. The software development lifecycle

The software development lifecycle (SDLC) is a structured and systematic approach to developing software applications [31]. It encompasses a series of phases and activities that guide the development process from initial conception to deployment and maintenance. It plays a crucial role in software development, including software security. According to [32] and [33], SDLC is a structured approach that outlines the steps involved in the development, deployment, and maintenance of software systems. It provides a systematic and structured approach to software development, ensuring that security considerations are integrated at each stage. By following the SDLC, organizations can minimize security risks, address vulnerabilities proactively, and deliver software systems that are more resilient to security threats [34]-[37]. As explained in [38], SDLC serves as a framework to guide developers and teams in building secure software [39] and facilitates the establishment of a security-conscious culture within organizations. While specific SDLC models may vary, the stages in Fig.1 are commonly found in most software development lifecycles:

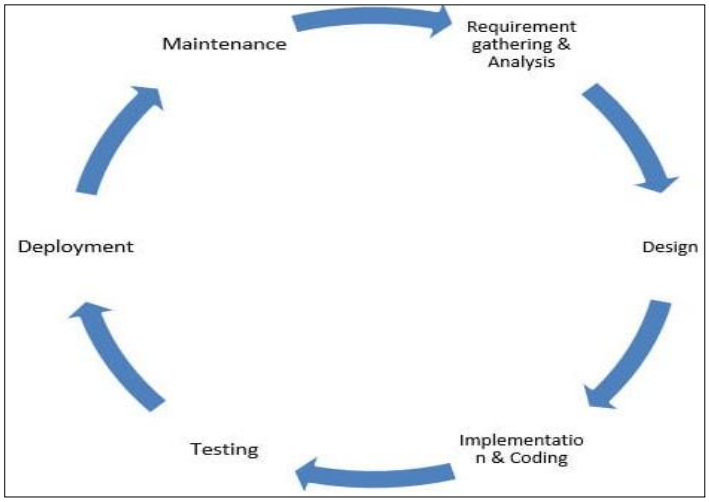


**Figure 1** Typical SDLC model

At the requirements gathering phase, the project team identifies and documents the software requirements based on user needs, business objectives, and stakeholder input [40]. This involves understanding the purpose of the software, its functionality, and any constraints or dependencies. However, at the analysis and design stage, the project team analyzes the requirements and designs the software architecture, data structures, and user interface [41]. This includes creating system and component-level designs, defining the system's behavior and interactions, and selecting appropriate technologies [42]. During implementation (coding) phase, the software development team translates the design specifications into actual code [43], [44]. They follow coding standards, use appropriate programming languages, and apply software engineering best practices to build the software components and functionality.

As explained in [45] and [46], testing is a critical phase to ensure that the software meets the specified requirements and functions correctly. It involves creating and executing test cases, identifying defects or bugs, and verifying that the software performs as intended. Different testing techniques such as unit testing, integration testing, system testing, and acceptance testing are employed to validate the software [47]. Once the software has passed testing and quality assurance, it is deployed to the production environment or made available to end-users [48]. This phase involves activities such as installation, configuration, and integration with existing systems. After deployment, the software enters the maintenance phase [49]- [51]. This involves monitoring its performance, addressing any issues or bugs discovered in production, and making necessary updates or enhancements. Maintenance may include bug fixes, security patches, performance optimizations [52], and feature additions based on user feedback and changing requirements.

Throughout the SDLC, various project management and development methodologies can be applied, such as waterfall, agile, or DevOps, to ensure effective collaboration, manage project risks, and deliver high-quality software [53], [54]. Additionally, continuous integration and continuous delivery (CI/CD) practices are often implemented to automate builds, testing, and deployment processes, enabling faster and more frequent software releases. It is important to note that the software development lifecycle is not always linear or strictly sequential. Iterative and incremental approaches are commonly used, where feedback from testing and user evaluations informs subsequent iterations of development, allowing for continuous improvement and adaptation.

## 3. Software development approaches

Software development approaches play a crucial role in shaping how software is planned, developed, and delivered [55]. These approaches provide frameworks, methodologies, and guidelines for organizing and executing the software development process. According to [56], they offer structure, guidance, and best practices for the software development process. In addition, they support project organization, efficiency, collaboration, risk management, quality assurance, and alignment with business goals. Table 1 presents some of the key roles played by the software development approaches.

**Table 1** Roles played by software development approaches

| Role | Description |
|---|---|
| Alignment with business goals | Help align development efforts with business goals and customer needs. They emphasize the value of delivering working software that meets customer expectations and provides business benefits [57]. By focusing on delivering value and addressing customer priorities, software development approaches enable organizations to achieve their strategic objectives. |
| Process efficiency and agility | They promote process efficiency [58] and agility. They provide methodologies and techniques for optimizing the development workflow, managing dependencies, and minimizing waste. Agile approaches, such as Scrum or Kanban, emphasize iterative and incremental development, allowing for flexibility, adaptability, and faster delivery of software. |
| Project organization and planning | Provide structures for organizing and planning projects. They define roles and responsibilities, establish communication channels, and facilitate coordination among team members. By following a development approach, teams can effectively manage resources, set clear goals and objectives, and establish project timelines [59]. |
| Quality assurance and testing | Incorporate quality assurance and testing practices into the development process. They provide guidelines for conducting various testing activities, including unit testing, integration testing, and user acceptance testing [60]. These approaches help ensure that software meets quality standards, functions as intended, and is free from bugs and vulnerabilities. |
| Continuous improvement and learning | Software development approaches promote continuous improvement and learning [61], [62]. They encourage retrospectives, where teams reflect on their processes, identify areas for improvement, and implement changes to enhance efficiency and quality. Continuous learning and adaptation are core principles in Agile approaches, fostering a culture of continuous improvement and innovation. |
| Security and quality assurance | Software development approaches increasingly emphasize security and quality assurance practices. They integrate security considerations, such as secure coding practices, security testing, and vulnerability [63] management, into the development process. These approaches promote a security-conscious mindset and help address security vulnerabilities proactively [64], [65]. |
| Collaboration and communication | They foster collaboration and effective communication among team members and stakeholders. They provide frameworks for conducting regular meetings, gathering feedback, and aligning the development process with customer needs. Collaboration tools and practices, such as daily stand-up meetings and user story workshops, are often integrated into these approaches to enhance teamwork and stakeholder engagement [66]. |
| Risk management | Software development approaches address risk management by identifying, assessing, and mitigating risks throughout the development lifecycle [67]. They provide mechanisms for risk identification, risk analysis, and risk mitigation planning. By proactively managing risks, development teams can minimize the impact of potential issues and make informed decisions to mitigate risks effectively [68]. |

By following a development approach, organizations can improve development outcomes, enhance customer satisfaction, and deliver high-quality software that meets security requirements. According to [69], there are several software development approaches or methodologies that organizations and development teams can adopt to guide their software development process. Table 2 describes some of the commonly used approaches.

**Table 2** Common software development approaches

| Approach | Particulars |
|---|---|
| Extreme Programming (XP) | Is an Agile methodology that emphasizes teamwork, communication, and high-quality software [70].<br>Promotes frequent releases, continuous testing, pair programming, collective code ownership, and customer involvement.<br>Emphasizes simplicity, flexibility, and adaptability in response to changing requirements. |
| Spiral | Combines elements of both Waterfall and iterative development [71].<br>Involves multiple iterations, each consisting of planning, risk analysis, development, and customer evaluation.<br>Incorporates feedback loops and allows for the incorporation of changes and refinements throughout the development process. |
| Agile | Agile methodologies, such as Scrum and Kanban, prioritize flexibility, collaboration, and iterative development [72].<br>Agile teams work in short cycles called sprints, delivering functional increments of software at the end of each sprint.<br>Requirements and solutions evolve through collaboration between cross-functional teams and stakeholders. |
| Feature-Driven Development (FDD) | Is an iterative and incremental approach that focuses on delivering specific features.<br>Emphasizes the use of domain modelling, feature identification [73], and short development cycles.<br>Divides the development process into manageable feature-driven workflows and encourages regular feature delivery [74]. |
| Waterfall | Is a linear and sequential approach to software development [75].<br>It follows a predefined set of phases: requirements gathering, design, implementation, testing, deployment, and maintenance.<br>Each phase has its specific goals and deliverables, and progress moves in a straightforward, top-down manner. |
| DevOps | Is a cultural and technical approach that emphasizes collaboration and integration between development and operations teams [76].<br>Aims to automate and streamline the software delivery process, allowing for frequent and reliable deployments.<br>Its practices include continuous integration, continuous delivery, infrastructure as code, and automated testing. |
| Lean | Focuses on reducing waste, improving efficiency, and continuously delivering value to customers [77].<br>Emphasizes the elimination of non-value-adding activities, streamlining processes, and optimizing workflow.<br>Lean principles are often combined with Agile methodologies to achieve faster and more efficient [78] software development. |
| Rapid Application Development (RAD) | Focuses on rapid prototyping and iterative development.<br>Aims to accelerate the development process by emphasizing user involvement, iterative feedback, and quick delivery of working software [79].<br>Often involves the use of visual development tools and components to expedite development. |

Each of these approaches has its strengths and suitability for different projects and teams. It is therefore important to select the approach that best aligns with project requirements, team dynamics, and organizational goals. In addition,

many organizations also adopt hybrid approaches, combining elements from multiple methodologies to tailor a development process that suits their specific needs.

## 4. Software security

Software security refers to the practice of protecting computer systems and software applications from unauthorized access, use, disclosure, disruption, modification, or destruction [80]. It involves implementing measures to prevent security vulnerabilities and mitigate the risks associated with potential attacks [81]-[83]. As explained in [84], the role of software security is to protect software systems from security threats, vulnerabilities, and attacks. It involves implementing measures and practices to ensure the confidentiality, integrity, and availability of software and the data it processes. According to [85], software security endervous to protect software systems, data, and users from security threats. It encompasses various practices, controls, and processes that enable organizations to develop, deploy, and maintain secure software systems [86-[88]. By prioritizing software security, organizations can mitigate risks, protect sensitive information, and maintain the trust and confidence of their stakeholders. As discussed in [89], understanding and mitigating software security risks is crucial to protect software systems and the data they handle. This may involve implementing appropriate security controls, conducting regular security assessments, following secure coding practices, and staying updated with security patches and updates. Table 3 describes the key rationale for software security.

**Table 3** Rationale for software security

| Rationale | Details |
|---|---|
| Building trust and confidence | Software security is essential for building trust and confidence among users, customers, and stakeholders [90].<br>When software is secure and protected against security threats, it instills confidence in users that their data and information are safe.<br>Strong security measures help organizations establish a positive reputation, retain customer loyalty, and gain a competitive edge in the market. |
| Preventing exploitation of vulnerabilities | It focuses on identifying and mitigating vulnerabilities that could be exploited by attackers.<br>This includes addressing coding flaws, insecure configurations, and design weaknesses that can lead to security breaches [91].<br>By proactively addressing vulnerabilities, software security reduces the risk of unauthorized access, data manipulation, and system compromise. |
| Fostering a security-conscious culture | Aims to foster a security-conscious culture within organizations.<br>It involves promoting security awareness, providing training to developers and stakeholders, and encouraging secure coding practices.<br>By integrating security into the software development process, organizations prioritize security from the initial stages and ensure a proactive approach to software security [92]. |
| Protecting sensitive information | Aims to safeguard sensitive information such as personal data [93], financial records, intellectual property, and customer information.<br>By implementing appropriate security controls, encryption mechanisms, and access management, software security prevents unauthorized access, data breaches, and information leakage [94]. |
| Mitigating risks and compliance | Plays a vital role in mitigating risks associated with legal and regulatory compliance.<br>Helps organizations meet industry-specific standards, data protection regulations, and privacy requirements [95].<br>Helps organizations avoid legal liabilities, financial losses, and reputational damage associated with non-compliance. |
| Responding to security incidents | Includes incident response and management processes to address security incidents effectively [96].<br>Involves implementing incident detection mechanisms, establishing incident response plans, and conducting forensic investigations.<br>Helps minimize the impact of breaches, identify root causes, and prevent future occurrences. |

| | |
|---|---|
| Ensuring availability and reliability | Aims to ensure the availability and reliability of software systems [97].<br>By implementing measures such as redundancy, load balancing, and disaster recovery mechanisms, software security helps prevent disruptions caused by denial-of-service attacks [98], system failures, or other malicious activities.<br>Ensures that software systems remain operational and accessible to legitimate users. |

By addressing software security risks, organizations can reduce the likelihood and impact of security breaches and safeguard their software assets. Some of the key aspects considered in software security include threat modeling [99]-[102], secure design and development [103], authentication and authorization [104], data protection [105], regular updates and patching [106], secure configuration and deployment, secure testing, incident response and recovery, user education and awareness, compliance and regulations.

In threat modeling, the goal is to analyze and understand the potential threats and vulnerabilities that could affect the software [107]. This involves identifying potential attackers, their motivations, and the potential risks to the system. However, for secure design and development, the aim is to follow secure coding practices, such as input validation, output encoding, and proper error handling [108]. This helps prevent common vulnerabilities like injection attacks such as SQL injection [109] and cross-site scripting. On the other hand, implementing strong authentication mechanisms, such as multi-factor authentication can ensure that only authorized users can access the software [110]. Additionally, implementing proper authorization controls ensures that users have appropriate permissions and access privileges [111]-[114]. For data protection, sensitive data is protected both at rest (stored) and in transit (being transmitted over a network). Basically, encryption techniques, such as using strong algorithms and secure protocols like SSL/TLS, help safeguard data from unauthorized access [115]. On the other hand, keeping software up to date with the latest security patches and updates is crucial. This includes both the software itself and any libraries or dependencies it relies on. Regularly monitoring for security vulnerabilities and promptly applying patches helps mitigate known risks [116].

Proper configuration of software and infrastructure is essential for security. This means that default configurations should be changed, unnecessary features and services should be disabled, and access controls should be properly defined [117]. On the other hand, conducting thorough security testing, such as penetration testing and vulnerability scanning, helps identify weaknesses and vulnerabilities in the software [118]. Essentially, regular testing, including both manual and automated techniques, allows for proactive identification and mitigation of security risks. As explained in [119], having a well-defined incident response plan in place enables the organization to respond promptly and effectively to security incidents. It includes procedures for detecting, responding to, and recovering from security breaches [120]. On the other hand, educating users about good security practices, such as strong password management, recognizing phishing attempts, and exercising caution while using the software, helps create a security-conscious environment [121]. Researchers in [122] explain that depending on the industry and geographical location, software may need to adhere to specific compliance standards and regulations such as GDPR, HIPAA. It is important to understand and implement the necessary security controls to meet these requirements.

Based on the above discussion, it is clear that software security is an ongoing process that requires continuous monitoring, adaptation, and improvement. By incorporating security practices throughout the software development lifecycle, organizations can mitigate risks and protect their systems, data, and users from potential threats.

## 5. Common software security risks

Software security risks refer to potential vulnerabilities, threats, and weaknesses that can compromise the security of software systems [123]. These risks pose a threat to the confidentiality, integrity, and availability of the software, as well as the data it processes. These risks can lead to unauthorized access, data breaches, system compromise, financial loss, reputational damage, and legal liabilities. Table 4 discusses some of common software security risks.

According to [135], software security risks refer to the vulnerabilities and weaknesses that can be exploited by attackers to compromise the security of software applications. Therefore, understanding and addressing these risks is crucial to protect sensitive data, maintain system integrity, and ensure the confidentiality and availability of software resources. Threats such as injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), Insecure Direct Object References (IDOR), security mis-configurations, weak authentication and authorization, Denial-of-Service (DoS), vulnerabilities in third-party libraries, insider threats and lack of secure communication all present some levels of risks to the software systems [136],[137].

**Table 4** Common software security risks

| Software risk | Description |
|---|---|
| Third-party dependencies | Relying on third-party libraries, frameworks, or components introduces the risk of inheriting vulnerabilities from these dependencies [124].<br>If the third-party software has security flaws or is not regularly updated, it can create security risks within the software system. |
| Insufficient logging and monitoring | Inadequate logging and monitoring capabilities can hinder the detection and response to security incidents [125].<br>Without proper logging, it becomes challenging to identify suspicious activities, track unauthorized access, or investigate security breaches effectively. |
| Software vulnerabilities | Are weaknesses in the software's code or design that can be exploited by attackers [126].<br>These vulnerabilities can include buffer overflows, injection attacks, cross-site scripting (XSS), and insecure direct object references [127].<br>Exploiting these vulnerabilities can lead to data breaches, code execution, or unauthorized system manipulation. |
| Insecure authentication and authorization | Weak or inadequate authentication and authorization mechanisms can allow unauthorized users to gain access to the software system and its resources [128].<br>This can result in unauthorized data access, privilege escalation, and unauthorized actions within the system. |
| Insecure data storage and transmission | Insecure handling of sensitive data, such as storing data in plain text or transmitting data over unsecured channels, can expose data to unauthorized access or interception [130].<br>This can result in data breaches, identity theft, and compromise of confidential information. |
| Inadequate input validation | Insufficient input validation can lead to various security risks, such as injection attacks [131], including SQL injection and command injection.<br>Without proper input validation, malicious inputs can manipulate the software's behavior, leading to unauthorized access, data corruption, or system compromise [132]. |
| Lack of secure configuration | Incorrect or insecure configurations of software components, servers, and network devices can create security vulnerabilities [133].<br>Default passwords, mis-configured access controls, and unnecessary services or ports can provide avenues for attackers to exploit and gain unauthorized access to the system. |
| Social engineering attacks | Software security risks also include social engineering attacks, where attackers manipulate individuals to gain unauthorized access or divulge sensitive information [134].<br>Phishing, pre-texting, and baiting are examples of social engineering techniques that can exploit human vulnerabilities to breach software security. |

According to [138], injection attacks, such as SQL injection and cross-site scripting (XSS), involve malicious code or commands being injected into software components that interpret user input. These attacks can lead to unauthorized data access, data manipulation, or code execution. In particular, XSS vulnerabilities allow attackers to inject malicious scripts into web pages viewed by other users, potentially compromising their browsers and enabling attacks like session hijacking or phishing. On the other hand, CSRF attacks trick authenticated users into performing unintended actions on a website by leveraging their trusted session [139]. This can result in unauthorized changes to user settings or data. As explained in [140], IDOR vulnerabilities occur when an application exposes direct references to internal objects or resources, allowing attackers to manipulate object identifiers and gain unauthorized access to sensitive data or functionalities [141]. On the other hand, security mis-configurations arise from improperly configured software components, servers, databases, or frameworks. This can leave software systems vulnerable to attacks, such as unauthorized access or privilege escalation. The researchers in [142] and [143] point out that weak authentication mechanisms, such as the use of weak passwords or insufficient authentication protocols, can lead to unauthorized access [144]. Inadequate authorization controls may allow unauthorized users to gain elevated privileges or access sensitive functionalities [145], [146]. On the other hand, DoS attacks aim to disrupt the availability of a service or system by overwhelming it with excessive requests or exploiting vulnerabilities that exhaust system resources. These attacks

can render a system unusable, leading to service disruptions or downtime. As discussed in [147], many software applications rely on third-party libraries or components, which may have vulnerabilities. If these vulnerabilities are not properly addressed or updated, they can introduce security risks into the software and be exploited by attackers [148], [149]. On the other hand, insider threats involve individuals with authorized access to systems or information who misuse their privileges for malicious purposes or inadvertently introduce security risks [130]. These threats can come from employees, contractors, or anyone with trusted access to the system. Researchers in [151] point out that inadequate encryption and insecure communication protocols can expose sensitive data to interception or unauthorized access during transmission, compromising data confidentiality and integrity.

Evidently, addressing the above software security risks requires implementing security best practices throughout the software development lifecycle, conducting regular security assessments and testing, and staying informed about emerging vulnerabilities and mitigation strategies. It is essential to prioritize security and adopt a proactive approach to protect software systems and the data they handle. The following section explores the role of theory and practice towards achieving this protection.

## 6. Theory and practice of software security

This encompasses both the conceptual understanding and practical implementation of security measures in software development and maintenance. It also involves various principles, models, methodologies, and techniques aimed at designing, building, and maintaining secure software systems [152]. Table 5 illustrates some of the key aspects of the theory and practice of software security.

**Table 5** Key aspects of the theory and practice of software security

| Aspect | Justification |
|---|---|
| Security Assurance | Encompasses activities that ensure the integrity and trustworthiness of software systems [153].<br>Involves activities such as security audits, compliance assessments, secure configuration management, and continuous monitoring.<br>Helps validate that security controls are in place and effective throughout the software's lifecycle. |
| Secure Development Lifecycle (SDL) | Is a systematic approach to integrating security into the software development process.<br>Typically involves several phases, including requirements analysis, design, coding, testing, deployment, and maintenance [154].<br>Each phase incorporates security activities such as threat modelling, secure coding practices, security testing, and vulnerability management. |
| Secure Design Patterns | Design patterns are reusable solutions to commonly occurring problems. In the context of software security, secure design patterns provide guidance on how to build secure software architectures [155].<br>Examples include input validation, output encoding, access control mechanisms, secure session management, and secure communication protocols. |
| Secure Coding Practices | Focus on writing code that is resistant to various types of attacks.<br>Includes proper input validation, output sanitization, secure error handling, secure memory management, and protection against common vulnerabilities such as injection attacks, cross-site scripting (XSS), and buffer overflows [156]. |
| Security Principles | Software security is guided by fundamental principles such as the CIA triad (Confidentiality, Integrity, and Availability), least privilege, defence in depth, and fail-safe defaults [157].<br>These principles help shape the overall security strategy and decision-making process [158]. |
| Threat Modelling | Is the process of identifying potential threats, vulnerabilities, and risks to a software system [159].<br>Involves analyzing the system's architecture, components, and interactions to understand the potential attack vectors and their potential impact.<br>Helps in making informed security decisions and prioritizing security measures. |

| | |
|---|---|
| Secure Software Maintenance | Software security is not a one-time effort; it requires ongoing maintenance and updates.<br>Patch management, vulnerability management, and timely software updates are crucial to address newly discovered vulnerabilities and security threats [160]. |
| Security Culture and Training | Building a security-conscious culture within an organization is essential.<br>Involves creating awareness, providing training, and promoting security best practices among developers, testers, and other stakeholders involved in the software development process [161]. |
| Security Testing | Aims to identify vulnerabilities and weaknesses in software systems.<br>Includes techniques such as penetration testing, vulnerability scanning, code reviews, and security-focused testing methodologies [162].<br>By uncovering vulnerabilities, organizations can address them before deploying the software. |
| Security Governance | Establishes the framework and processes for managing software security.<br>Involves defining security policies, assigning responsibilities, implementing security controls, and ensuring compliance with applicable regulations and standards [163], [164].<br>Provides the structure and accountability necessary to manage and maintain software security effectively. |

It is important to note that the theory and practice of software security are continually evolving due to the evolving threat landscape. As such, keeping up with emerging security trends, staying informed about new vulnerabilities, and adapting security practices accordingly is vital to ensuring the resilience and security of software systems.

## 7. Methodologies for identifying and eliminating security vulnerabilities

Identifying and eliminating security vulnerabilities in software applications is a critical process to protect against potential attacks and safeguard sensitive data [165]. There are several methodologies and approaches for identifying and eliminating security vulnerabilities in software applications. These include threat modeling, secure coding practices, secure SDLC, security code reviews, penetration testing, security scanning and vulnerability assessment. According to [166], threat modeling is a proactive approach to identify potential security threats and vulnerabilities in the early stages of software development. It involves analyzing the system architecture, components, and interactions to understand the potential attack vectors. By identifying threats and their potential impact, developers can prioritize security measures and design the software with security in mind [167]. To uphold secure coding practices, there is need to adhering to secure coding practices. This helps prevent common vulnerabilities and strengthens the overall security of the software. Practices such as input validation, output encoding, secure error handling, proper session management, and secure communication protocols can significantly reduce the risk of vulnerabilities like injection attacks, cross-site scripting (XSS), and others. According to [168], performing regular code reviews specifically focused on security can help identify vulnerabilities in the codebase. These code reviews involve manual examination of the code to detect insecure coding patterns, potential vulnerabilities, and adherence to secure coding practices. This can be done by experienced developers or security professionals to ensure a thorough assessment.

Penetration testing, also known as ethical hacking, involves simulating real-world attacks on the software to identify vulnerabilities [169]. Here, skilled security professionals use various tools and techniques to probe the software's security defenses, exploit vulnerabilities, and provide recommendations for remediation. This helps identify weaknesses that may not be apparent through other security assessment methods. On the other hand, automated security scanning tools and vulnerability assessment scanners can be used to detect common vulnerabilities in software applications [170]. These tools scan the application's code, configurations, and dependencies to identify potential weaknesses. In so doing, they help uncover issues such as insecure configurations, outdated libraries, and known vulnerabilities, providing a starting point for remediation efforts [171]. As explained in [172], educating developers, testers, and other stakeholders about secure coding practices, common vulnerabilities, and emerging threats is essential. Security training programs increase awareness and provide the knowledge necessary to develop and maintain secure software applications. It helps in cultivating a security-conscious culture within the organization. Similarly, incorporating security practices into the entire software development lifecycle is crucial [173]. This includes integrating security activities at each phase of the SDLC, such as threat modeling, secure design, secure coding, security testing, and security-focused quality assurance. A well-defined and consistently followed SDLC ensures that security is a fundamental aspect of the development process.

As pointed out in [174], these methodologies are not mutually exclusive, and a combination of approaches is often necessary to achieve comprehensive vulnerability identification and remediation. Additionally, staying updated with security advisories, industry best practices, and security communities can provide valuable insights into emerging vulnerabilities and mitigation strategies.

## 8. Tools for identifying and eliminating security vulnerabilities

There are numerous tools available that can assist in identifying and eliminating security vulnerabilities in software applications [175]. They automate certain processes, provide analysis, and help identify potential weaknesses. Table 6 presents some of these tools.

**Table 6** Security vulnerabilities identification and elimination tools

| **Tool** | **Examples** |
|---|---|
| Software Composition Analysis (SCA) Tools | *Black Duck*: Scans application dependencies and identifies known vulnerabilities in open-source libraries and components.<br>*Sonatype Nexus Lifecycle*: Helps identify and manage open-source components in software development, ensuring they are free from vulnerabilities and licensing issues.<br>*WhiteSource*: Offers SCA capabilities to identify and manage open-source components and libraries used in software applications. |
| Static Application Security Testing (SAST) Tools | *Fortify*: A popular SAST tool that analyzes source code to identify security vulnerabilities and coding errors.<br>*Veracode*: Offers SAST capabilities for identifying flaws in code and providing recommendations for remediation.<br>*Checkmarx*: Provides static code analysis to identify security vulnerabilities and offers integration with various development environments. |
| Interactive Application Security Testing (IAST) Tools | *Contrast Security*: Provides runtime analysis of applications to detect vulnerabilities during the execution of code.<br>*Seeker*: Offers IAST capabilities for detecting and verifying security vulnerabilities in web applications. |
| Security Information and Event Management (SIEM) Tools | *Splunk*: A SIEM tool that collects and analyzes log data from various sources to detect security incidents and anomalies.<br>*Elastic Stack*: An open-source toolset that includes Elasticsearch, Logstash, and Kibana for log management, analysis, and visualization. |
| Security Scanners and Analyzers | *Qualys*: Offers cloud-based vulnerability management and scanning for identifying and prioritizing vulnerabilities in systems and applications.<br>*Rapid7 AppSpider*: Provides automated scanning for web applications to identify security vulnerabilities and misconfigurations. |
| Dynamic Application Security Testing (DAST) Tools | *Burp Suite*: A comprehensive DAST tool that allows manual and automated security testing of web applications.<br>*OWASP ZAP*: An open-source DAST tool that helps identify common web application vulnerabilities and supports automation.<br>*Acunetix:* Provides DAST capabilities to scan web applications for security vulnerabilities, including injection attacks, XSS, and more. |
| Penetration Testing Tools | *Metasploit*: A popular penetration testing framework that provides a range of tools for exploiting vulnerabilities and assessing security.<br>*Nmap*: A versatile network scanning tool that helps identify open ports, services, and potential vulnerabilities in a network.<br>*Nessus*: A vulnerability scanner that detects security issues in networks, systems, and applications. |

The tools in Table 6 can be integrated into the software development lifecycle and used in combination to provide comprehensive vulnerability identification and remediation. While these tools can assist in vulnerability management, human expertise and analysis are still crucial to interpret results, prioritize vulnerabilities, and apply appropriate remediation measures.

## 9. Techniques to prove the absence of vulnerabilities

The procedures for proving the absence of vulnerabilities in software applications is a challenging task since it is nearly impossible to guarantee complete absence [176]. However, there are several techniques and practices that can help increase confidence in the security of a system. Some of these approaches include security testing, secure coding practices, code reviews and security audits, threat modeling, security by design, compliance with security standards, security verification [177] standard, independent security assessments, security awareness and training, secure development frameworks and libraries. According to [178], security testing involves the conducting of comprehensive security testing using techniques such as penetration testing, vulnerability scanning, and code review. These assessments aim to identify and address security vulnerabilities proactively. By performing rigorous testing, organizations can minimize the chances of undetected vulnerabilities. However, secure coding practices encompasses the adoption of security from the initial stages of development. This includes adhering to secure coding guidelines and best practices, implementing input validation, output encoding, proper error handling, and secure communication protocols. By following these practices, developers can minimize the likelihood of introducing vulnerabilities during the coding process. It is important that regular code reviews and security audits be carried out to identify potential vulnerabilities [179]. Experienced developers or security experts can review the codebase to identify coding flaws, design weaknesses, and implementation errors. This helps uncover and address vulnerabilities before they are deployed. In addition, threat modeling techniques can be applied to identify potential security risks and vulnerabilities. Analyze the system architecture, identify potential threats, and assess the impact of those threats. By proactively addressing security risks during the design phase, organizations can minimize the likelihood of introducing vulnerabilities.

As pointed out in [180], a security-by-design approach need to be adopted throughout the software development lifecycle. This involves integrating security considerations from the early stages of development, incorporating secure design principles, and continuously assessing and mitigating potential risks. By making security an inherent part of the development process, the chances of introducing vulnerabilities are reduced. It is also important to comply with established security standards and frameworks such as ISO 27001, NIST Cybersecurity Framework, or OWASP Application Security Verification Standard [181]. Adhering to these standards ensures that security practices and controls are in place, reducing the likelihood of vulnerabilities. Moreover, organizations need to engage third-party security experts or external auditors to perform independent security assessments of the software application [182]. These professionals can evaluate the system for vulnerabilities [183] and provide objective feedback and recommendations for improvement. For enhanced security, organizations need to foster a culture of security awareness among developers, testers, and other stakeholders involved in the software development process. As explained in [184], regular training sessions on secure coding practices, emerging threats, and security policies help individuals understand their role in maintaining secure software and reduce the likelihood of introducing vulnerabilities. As explained in [185], it is crucial to leverage established secure development frameworks and libraries that have undergone extensive security testing and auditing. These frameworks and libraries are built with security in mind and can help minimize vulnerabilities resulting from incorrect implementations.

While these techniques can help reduce vulnerabilities, it is important to understand that proving the absolute absence of vulnerabilities is challenging. Security is an ongoing effort, and organizations should continuously monitor, assess, and improve the security posture of their software applications.

## 10. Essential guidelines for building secure software

To avoid security holes in new software, it is crucial to incorporate security practices throughout the development lifecycle. Some key ways to mitigate the risk of security vulnerabilities include secure design and secure coding practices [186]. In secure design, developers begin with a strong and secure design phase, considering security requirements, threat modeling, and risk analysis during the design process. Identify potential security vulnerabilities and design the system architecture and components to minimize those risks. However, in secure coding practices, developers need to adhere to validate and sanitize user input, using parameterized queries or prepared statements to prevent injection attacks, implementing proper access controls and authentication mechanisms, and securely handling sensitive data [187], [188]. Apply security-focused coding guidelines and standards.

When building secure software, it is crucial to follow essential guidelines and best practices to minimize the risk of security vulnerabilities. Table 7 presents some of these key guidelines to consider.

**Table 7** Guidelines for building secure software

| **Guideline** | **Explanation** |
|---|---|
| Implement Secure Authentication and Authorization | Use strong and properly implemented authentication mechanisms [189].<br>Enforce strong password policies and consider implementing multi-factor authentication.<br>Implement proper authorization controls to ensure users have appropriate access privileges. |
| Validate and Sanitize User Input | Validate and sanitize all user input to prevent common vulnerabilities such as injection attacks (such as SQL injection, XSS).<br>Implement input validation on both the client and server sides to ensure data integrity [190]. |
| Protect Sensitive Data | Apply encryption algorithms to protect sensitive data, both in transit and at rest [191].<br>Use appropriate cryptographic protocols and algorithms, and ensure proper key management. |
| Follow Secure Coding Practices | Adhere to secure coding practices and coding guidelines that promote security, such as input validation, output encoding, secure error handling, and secure communication protocols [192], [193].<br>Avoid insecure coding patterns and practices, such as using deprecated functions or insecure configurations. |
| Manage Session Security | Implement secure session management techniques, including secure session tokens, session expiration, and secure cookie management [194].<br>Avoid storing sensitive information in session variables or cookies. |
| Handle Errors Securely | Implement proper error handling mechanisms to avoid exposing sensitive information to potential attackers [195].<br>Provide informative error messages to users without revealing system details that could be exploited. |
| Secure Configuration Management | Securely configure servers, frameworks, libraries, and dependencies [196].<br>Follow security best practices for server configurations, network settings, and access controls.<br>Disable unnecessary services, ports, and protocols to minimize the attack surface. |
| Regularly Update and Patch Software | Keep all software components, frameworks, libraries, and dependencies up to date with the latest security patches and updates [197].<br>Stay informed about security advisories and vulnerabilities [198] related to the software used and promptly apply patches. |
| Conduct Security Testing | Perform regular security testing, including penetration testing, vulnerability scanning, and code review [199].<br>Test for common vulnerabilities such as injection flaws, XSS, security mis-configurations, and access control issues.<br>Utilize both automated tools and manual testing techniques. |
| Foster a Security-Conscious Culture | Educate and train developers, testers, and other stakeholders on secure coding practices, emerging threats, and security policies [200].<br>Promote a culture of security awareness and encourage everyone to take responsibility for building secure software. |
| Follow Secure Development Lifecycle (SDLC) | Incorporate security practices into the entire software development lifecycle, from requirements gathering to deployment and maintenance [201].<br>Include security reviews, testing, and risk assessments at each stage of the SDLC. |

| | |
|---|---|
| Stay Informed about Security | Keep up to date with the latest security practices, standards, and emerging threats [202].<br>Monitor security communities, subscribe to security mailing lists, and stay informed about security news and developments. |

By following these guidelines, organizations can build software with a strong security foundation and reduce the risk of security vulnerabilities [203]. It is important to note that security is an ongoing process, and continuous monitoring, testing, and improvement are necessary to address evolving threats and maintain the security of the software over time.

## 11. Research gaps

While significant progress has been made in the field of software security, there are still several research gaps that merit further investigation. Some of the key research gaps in software security include:

*Secure Software Development Processes:* There is a need for more research on effective methodologies, frameworks, and tools for integrating security into the entire software development lifecycle [204]. This includes identifying best practices for secure requirements engineering, secure design, secure coding, and secure testing as shown in Fig.2.

*Automated Vulnerability Detection:* While there are various automated tools available for vulnerability detection, there is room for improvement in their accuracy, coverage, and effectiveness. Research is needed to develop advanced techniques for automated vulnerability detection, including static analysis, dynamic analysis, and hybrid approaches [205].

*Security of Emerging Technologies:* With the emergence of new technologies such as Internet of Things (IoT), cloud computing, blockchain, and artificial intelligence (AI), there is a need to explore their unique security challenges and develop robust security solutions and best practices specific to these domains [206].

*Secure Software Architectures:* Research is needed to explore secure software architectures that can effectively protect against modern threats and vulnerabilities [207], [208]. This includes designing architectures that provide strong isolation, access control mechanisms, secure communication channels, and resilience against attacks.

*Secure DevOps and Continuous Security*: As organizations increasingly adopt DevOps practices, there is a need for research on integrating security seamlessly into DevOps processes [209]. This involves developing methodologies and tools for continuous security testing, vulnerability management, and secure deployment pipelines.

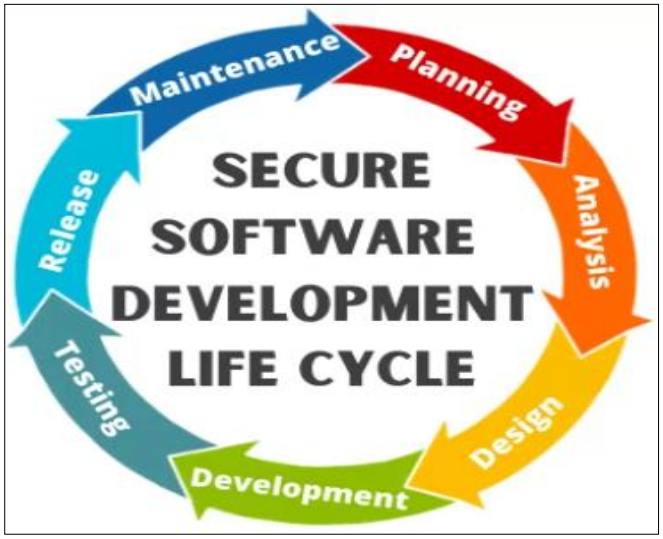


**Figure 2** Secure Software Development Processes

*Human Factors in Software Security:* Human factors, such as user behavior, social engineering, and organizational culture, play a significant role in software security [210]. Further research is needed to understand and address human vulnerabilities, improve user awareness and education, and develop effective security training and awareness programs.

*Threat Intelligence and Information Sharing:* Research is needed to improve the collection, analysis, and sharing of threat intelligence among organizations [211], [212]. This includes developing techniques for automated threat intelligence gathering, effective sharing platforms, and privacy-preserving mechanisms for information exchange.

*Secure Software Updates and Patch Management:* Software updates and patch management are critical for addressing vulnerabilities [213]-[215]. Research is required to develop efficient and secure mechanisms for software updates, including strategies for safe and reliable patch distribution and installation.

*Evaluating Security Controls:* There is a need for research on evaluating the effectiveness and impact of security controls and countermeasures [216], [217]. This includes developing metrics, frameworks, and methodologies to measure the security posture of software systems and assess the effectiveness of security controls.

*Socio-Technical Aspects of Software Security:* Understanding the socio-technical aspects of software security is crucial, as it involves the interaction between technology, people, and organizations [218]-[223]. Research is needed to explore the socio-cultural, economic, and legal factors that influence software security and develop strategies to address them effectively.

Addressing these research gaps will contribute to the advancement of software security, leading to the development of more secure software systems and better protection against evolving threats and vulnerabilities.

## 12. Conclusion

Software security is of paramount importance in today's digital landscape where cyber threats are prevalent. Developing secure software requires a holistic and proactive approach that encompasses all stages of the software development lifecycle. By integrating security practices from the initial design phase, employing secure coding practices, conducting regular security testing, and following secure configuration and patching processes, organizations can significantly mitigate the risk of security vulnerabilities. Investing in software security not only protects sensitive data and intellectual property but also safeguards the reputation and trust of the organization. It helps prevent financial losses, legal liabilities, and potential disruptions caused by security incidents. Moreover, prioritizing software security demonstrates a commitment to customer privacy and compliance with regulatory requirements. However, it is important to acknowledge that software security is an ongoing process. Threats and vulnerabilities evolve continuously, and new attack vectors emerge. Organizations should stay vigilant, keep abreast of the latest security practices and technologies, and continuously assess and improve the security posture of their software applications. Ultimately, software security is a shared responsibility among developers, testers, security professionals, and stakeholders. Collaboration, education, and awareness are key in fostering a security-conscious culture and ensuring that security is ingrained in every aspect of the software development process. By doing so, organizations can build robust, resilient, and trusted software systems that protect against threats and instill confidence in users and stakeholders.

## Compliance with ethical standards

### *Acknowledgments*

We would like to thank everyone who offered support during the development of this paper.

### *Disclosure of conflict of interest*

The authors declare that they do not have any conflict of interest.

## References

[1] Nath, P., Mushahary, J. R., Roy, U., Brahma, M., & Singh, P. K. (2023). AI and Blockchain-based source code vulnerability detection and prevention system for multiparty software development. Computers and Electrical Engineering, 106, 108607.

[2] Khan, A. A., Ahmad, A., Waseem, M., Liang, P., Fahmideh, M., Mikkonen, T., & Abrahamsson, P. (2023). Software architecture for quantum computing systems—A systematic review. Journal of Systems and Software, 201, 111682.

[3] Alam, A., & Mohanty, A. (2023). Discerning the Application of Virtual Laboratory in Curriculum Transaction of Software Engineering Lab Course from the Lens of Critical Pedagogy. In Sentiment Analysis and Deep Learning: Proceedings of ICSADL 2022 (pp. 53-68). Singapore: Springer Nature Singapore.

[4] Borgia, E. (2014). The Internet of Things vision: Key features, applications and open issues. Computer Communications, 54, 1-31.

[5] Eid M.M., Arunachalam R., Sorathiya V., Lavadiya S., Patel S.K., Parmar J., Delwar T.S., Ryu J.Y., Nyangaresi V.O., Zaki Rashed A.N. (2022). QAM receiver based on light amplifiers measured with effective role of optical coherent duobinary transmitter. Journal of Optical Communications, (0).

[6] Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., ... & Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, 103, 102274.

[7] Stradowski, S., & Madeyski, L. (2023). Exploring the challenges in software testing of the 5G system at Nokia: A survey. Information and Software Technology, 153, 107067.

[8] Solanki, P., Grundy, J., & Hussain, W. (2023). Operationalising ethics in artificial intelligence for healthcare: A framework for AI developers. AI and Ethics, 3(1), 223-240.

[9] White, J., Hays, S., Fu, Q., Spencer-Smith, J., & Schmidt, D. C. (2023). Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. arXiv preprint arXiv:2303.07839.

[10] Papagiannidis, E., Enholm, I. M., Dremel, C., Mikalef, P., & Krogstie, J. (2023). Toward AI governance: Identifying best practices and potential barriers and outcomes. Information Systems Frontiers, 25(1), 123-141.

[11] Nyangaresi, V. O. (2023). Extended Chebyshev Chaotic Map Based Message Verification Protocol for Wireless Surveillance Systems. In Computer Vision and Robotics: Proceedings of CVR 2022 (pp. 503-516). Singapore: Springer Nature Singapore.

[12] George, A. S., & George, A. H. (2023). A review of ChatGPT AI's impact on several business sectors. Partners Universal International Innovation Journal, 1(1), 9-23.

[13] Alfadel, M., Costa, D. E., & Shihab, E. (2023). Empirical analysis of security vulnerabilities in python packages. Empirical Software Engineering, 28(3), 59.

[14] Bernholdt, D. E., Doucet, M., Godoy, W. F., Malviya-Thakur, A., & Watson, G. R. (2023). Experiential findings for sustainable software ecosystems to support experimental and observational science. Journal of Computational Science, 102033.

[15] Mohammad, Z., Nyangaresi, V., & Abusukhon, A. (2021, July). On the Security of the Standardized MQV Protocol and Its Based Evolution Protocols. In 2021 International Conference on Information Technology (ICIT) (pp. 320-325). IEEE.

[16] Hasan, M. K., Habib, A. A., Shukur, Z., Ibrahim, F., Islam, S., & Razzaque, M. A. (2023). Review on cyber-physical and cyber-security system in smart grid: Standards, protocols, constraints, and recommendations. Journal of Network and Computer Applications, 209, 103540.

[17] Bagga, P., Das, A. K., & Rodrigues, J. J. (2023). Bilinear pairing-based access control and key agreement scheme for smart transportation. Cyber Security and Applications, 1, 100001.

[18] Nyangaresi, V. O. (2023, February). Target Tracking Area Selection and Handover Security in Cellular Networks: A Machine Learning Approach. In Proceedings of Third International Conference on Sustainable Expert Systems: ICSES 2022 (pp. 797-816). Singapore: Springer Nature Singapore.

[19] Safa, N. S., Sookhak, M., Von Solms, R., Furnell, S., Ghani, N. A., & Herawan, T. (2015). Information security conscious care behaviour formation in organizations. Computers & Security, 53, 65-78.

[20] Tejay, G. P., & Mohammed, Z. A. (2023). Cultivating security culture for information security success: A mixed-methods study based on anthropological perspective. Information & Management, 60(3), 103751.

[21] Safa, N. S., Von Solms, R., & Futcher, L. (2016). Human aspects of information security in organisations. Computer Fraud & Security, 2016(2), 15-18.

[22] Xue, B., Warkentin, M., Mutchler, L. A., & Balozian, P. (2023). Self-efficacy in information security: a replication study. Journal of Computer Information Systems, 63(1), 1-10.

[23] Hussien, Z. A., Abdulmalik, H. A., Hussain, M. A., Nyangaresi, V. O., Ma, J., Abduljabbar, Z. A., & Abduljaleel, I. Q. (2023). Lightweight Integrity Preserving Scheme for Secure Data Exchange in Cloud-Based IoT Systems. Applied Sciences, 13(2), 691.

[24] Assal, H., & Chiasson, S. (2018, August). Security in the Software Development Lifecycle. In SOUPS@ USENIX Security Symposium (pp. 281-296).

[25] Sharma, A., & Misra, P. K. (2017). Aspects of enhancing security in software development life cycle. Advances in Computational Sciences and Technology, 10(2), 203-210.

[26] Jaballah, W. B., Conti, M., & Lal, C. (2020). Security and design requirements for software-defined VANETs. Computer Networks, 169, 107099.

[27] Glenn, C., Sterbentz, D., & Wright, A. (2016). Cyber threat and vulnerability analysis of the US electric sector (No. INL/EXT-16-40692). Idaho National Lab.(INL), Idaho Falls, ID (United States).

[28] Nyangaresi, V. O., & Moundounga, A. R. A. (2021, September). Secure data exchange scheme for smart grids. In 2021 IEEE 6th International Forum on Research and Technology for Society and Industry (RTSI) (pp. 312-316). IEEE.

[29] Stewart, H. (2022). A systematic framework to explore the determinants of information security policy development and outcomes. Information & Computer Security.

[30] Arogundade, O. R. (2023). Network Security Concepts, Dangers, and Defense Best Practical. Computer Engineering and Intelligent Systems, 14(2).

[31] Jindal, T. (2016). Importance of Testing in SDLC. International Journal of Engineering and Applied Computer Science (IJEACS), 1(02), 54-56.

[32] Fan, C. Y., & Ma, S. P. (2017, June). Migrating monolithic mobile application to microservice architecture: An experiment report. In 2017 ieee international conference on ai & mobile services (aims) (pp. 109-112). IEEE.

[33] Kramer, M. (2018). Best practices in systems development lifecycle: An analyses based on the waterfall model. Review of Business & Finance Studies, 9(1), 77-84.

[34] Al Sibahee, M. A., Nyangaresi, V. O., Ma, J., & Abduljabbar, Z. A. (2022, July). Stochastic Security Ephemeral Generation Protocol for 5G Enabled Internet of Things. In IoT as a Service: 7th EAI International Conference, IoTaaS 2021, Sydney, Australia, December 13–14, 2021, Proceedings (pp. 3-18). Cham: Springer International Publishing.

[35] Shukla, A., Katt, B., Nweke, L. O., Yeng, P. K., & Weldehawaryat, G. K. (2022). System security assurance: A systematic literature review. Computer Science Review, 45, 100496.

[36] Sriram, G. S. (2022). Edge computing vs. Cloud computing: an overview of big data challenges and opportunities for large enterprises. International Research Journal of Modernization in Engineering Technology and Science, 4(1), 1331-1337.

[37] Jagatheesaperumal, S. K., Mishra, P., Moustafa, N., & Chauhan, R. (2022). A holistic survey on the use of emerging technologies to provision secure healthcare solutions. Computers and Electrical Engineering, 99, 107691.

[38] Dodson, D., Souppaya, M., & Scarfone, K. (2020). Mitigating the risk of software vulnerabilities by adopting a secure software development framework (ssdf). NIST: Gaithersburg, MD, USA.

[39] Nyangaresi, V. O. (2022, June). Masked Symmetric Key Encrypted Verification Codes for Secure Authentication in Smart Grid Networks. In 2022 4th Global Power, Energy and Communication Conference (GPECOM) (pp. 427-432). IEEE.

[40] Sharma, M. K. (2017). A study of SDLC to develop well engineered software. International Journal of Advanced Research in Computer Science, 8(3).

[41] Aljawarneh, S. A., Alawneh, A., & Jaradat, R. (2017). Cloud security engineering: Early stages of SDLC. Future Generation Computer Systems, 74, 385-392.

[42] Rashed A.N., Ahammad S.H., Daher M.G., Sorathiya V., Siddique A., Asaduzzaman S., Rehana H., Dutta N., Patel S.K., Nyangaresi V.O., Jibon R.H. (2022). Spatial single mode laser source interaction with measured pulse based parabolic index multimode fiber. Journal of Optical Communications.

[43] Raj, G., Singh, D., & Bansal, A. (2014, September). Analysis for security implementation in SDLC. In 2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence) (pp. 221-226). IEEE.

[44] Alshamrani, A., & Bahattab, A. (2015). A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model. International Journal of Computer Science Issues (IJCSI), 12(1), 106.

[45] Singh, V., Kumar, V., & Singh, V. B. (2023). A hybrid novel fuzzy AHP-Topsis technique for selecting parameter-influencing testing in software development. Decision Analytics Journal, 6, 100159.

[46] Nath, P., Mushahary, J. R., Roy, U., Brahma, M., & Singh, P. K. (2023). AI and Blockchain-based source code vulnerability detection and prevention system for multiparty software development. Computers and Electrical Engineering, 106, 108607.

[47] Nyangaresi, V. O., & Ogundoyin, S. O. (2021, October). Certificate based authentication scheme for smart homes. In 2021 3rd Global Power, Energy and Communication Conference (GPECOM) (pp. 202-207). IEEE.

[48] Santos, L. J. D., Ribeiro, S. A., Schimitz, E. A., Silva, M. F. D., & Alencar, A. J. S. M. D. (2022). Risk Factors in the Software Deployment Phase: A Case Study Applied in two Brazilian Government Companies. JISTEM-Journal of Information Systems and Technology Management, 19.

[49] Amado, A., & Belfo, F. P. (2021). Maintenance and support model within the ERP systems lifecycle: Action research in an implementer company. Procedia Computer Science, 181, 580-588.

[50] Hutchinson, B., Smart, A., Hanna, A., Denton, E., Greer, C., Kjartansson, O., ... & Mitchell, M. (2021, March). Towards accountability for machine learning datasets: Practices from software engineering and infrastructure. In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (pp. 560-575).

[51] Rahman, H. U., Raza, M., Afsar, P., & Khan, H. U. (2021). Empirical investigation of influencing factors regarding offshore outsourcing decision of application maintenance. IEEE Access, 9, 58589-58608.

[52] Zaki Rashed A.N., Ahammad S.H., Daher M.G., Sorathiya V., Siddique A., Asaduzzaman S., Rehana H., Dutta N., Patel S.K., Nyangaresi V.O., Jibon R.H. (2022). Signal propagation parameters estimation through designed multi layer fibre with higher dominant modes using OptiFibre simulation. Journal of Optical Communications, (0).

[53] Mohammad, S. M. (2017). DevOps automation and Agile methodology. International Journal of Creative Research Thoughts (IJCRT), ISSN, 2320-2882.

[54] Hemon, A., Lyonnet, B., Rowe, F., & Fitzgerald, B. (2020). From agile to DevOps: Smart skills and collaborations. Information Systems Frontiers, 22, 927-945.

[55] Kude, T., Foerderer, J., Mithas, S., & Heinzl, A. (2023). How deadline orientation and architectural modularity influence software quality and job satisfaction. Journal of Operations Management.

[56] Jemine, G., Puyou, F. R., & Dubois, C. (2023). The diffusion of management fashions as software in an intermediated market: The case of continuous accounting. Management Accounting Research, 100852.

[57] Alzoubi, H., Alshurideh, M., Kurdi, B., Akour, I., & Aziz, R. (2022). Does BLE technology contribute towards improving marketing strategies, customers' satisfaction and loyalty? The role of open innovation. International Journal of Data and Network Science, 6(2), 449-460.

[58] Nyangaresi, V. O., & Morsy, M. A. (2021, September). Towards privacy preservation in internet of drones. In 2021 IEEE 6th International Forum on Research and Technology for Society and Industry (RTSI) (pp. 306-311). IEEE.

[59] Govindaras, B., Wern, T. S., Kaur, S., Haslin, I. A., & Ramasamy, R. K. (2023). Sustainable Environment to Prevent Burnout and Attrition in Project Management. Sustainability, 15(3), 2364.

[60] Pillai, N. S. R., & Hemamalini, R. R. (2022). Hybrid User Acceptance Test Procedure to Improve the Software Quality. International Arab Journal of Information Technology, 19(6), 956-964.

[61] Yilmaz, M., & O'Connor, R. V. (2016). A Scrumban integrated gamification approach to guide software process improvement: a Turkish case study. Tehnički vjesnik, 23(1), 237-245.

[62] García-Mireles, G. A., Moraga, M. Á., García, F., & Piattini, M. (2015). Approaches to promote product quality within software process improvement initiatives: a mapping study. Journal of Systems and Software, 103, 150-166.

[63] Abduljabbar, Z. A., Omollo Nyangaresi, V., Al Sibahee, M. A., Ghrabat, M. J. J., Ma, J., Qays Abduljaleel, I., & Aldarwish, A. J. (2022). Session-Dependent Token-Based Payload Enciphering Scheme for Integrity Enhancements in Wireless Networks. Journal of Sensor and Actuator Networks, 11(3), 55.

[64] Lin, C., Wittmer, J. L., & Luo, X. R. (2022). Cultivating proactive information security behavior and individual creativity: The role of human relations culture and IT use governance. Information & Management, 59(6), 103650.

[65] Wolf, F., Kuber, R., & Aviv, A. J. (2018). An empirical study examining the perceptions and behaviours of security-conscious users of mobile authentication. Behaviour & Information Technology, 37(4), 320-334.

[66] Krehbiel, T. C., Salzarulo, P. A., Cosmah, M. L., Forren, J., Gannod, G., Havelka, D., ... & Merhout, J. (2017). Agile Manifesto for Teaching and Learning. Journal of Effective Teaching, 17(2), 90-111.

[67] Chauhan, A. S., Nepal, B., Soni, G., & Rathore, A. P. S. (2018). Examining the state of risk management research in new product development process. Engineering Management Journal, 30(2), 85-97.

[68] Nyangaresi, V. O. (2023). Privacy Preserving Three-factor Authentication Protocol for Secure Message Forwarding in Wireless Body Area Networks. Ad Hoc Networks, 142, 103117.

[69] Venkatesh, D., & Rakhra, M. (2020). Agile adoption issues in large scale organizations: A review. Materials Today: Proceedings.

[70] Tabassum, A., Manzoor, I., Bhatti, S. N., Asghar, A. R., & Alam, I. (2017). Optimized quality model for agile development: extreme programming (XP) as a case scenario. International Journal of Advanced Computer Science and Applications, 8(4).

[71] Akinsola, J. E., Ogunbanwo, A. S., Okesola, O. J., Odun-Ayo, I. J., Ayegbusi, F. D., & Adebiyi, A. A. (2020). Comparative analysis of software development life cycle models (SDLC). In Intelligent Algorithms in Software Engineering: Proceedings of the 9th Computer Science On-line Conference 2020, Volume 1 9 (pp. 310-322). Springer International Publishing.

[72] Parsons, D., Thorn, R., Inkila, M., & MacCallum, K. (2018, December). Using Trello to support agile and lean learning with Scrum and Kanban in teacher professional development. In 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE) (pp. 720-724). IEEE.

[73] Nyangaresi, V. O., El-Omari, N. K. T., & Nyakina, J. N. (2022). Efficient Feature Selection and ML Algorithm for Accurate Diagnostics. Journal of Computer Science Research, 4(1), 10-19.

[74] Firdaus, A., Ghani, I., & Jeong, S. R. (2014). Secure feature driven development (SFDD) model for secure software development. Procedia-Social and Behavioral Sciences, 129, 546-553.

[75] Christanto, H. J., & Singgalen, Y. A. (2023). Analysis and Design of Student Guidance Information System through Software Development Life Cycle (SDLC) and Waterfall Model. Journal of Information Systems and Informatics, 5(1), 259-270.

[76] Amaro, R., Pereira, R., & da Silva, M. M. (2023). Capabilities and Metrics in DevOps: A Design Science Study. Information & Management, 103809.

[77] Hosseinizadeh Mazloumi, S. H., Moini, A., & Agha Mohammad Ali Kermani, M. (2023). Designing synchronizer module in CMMS software based on lean smart maintenance and process mining. Journal of Quality in Maintenance Engineering, 29(2), 509-529.

[78] Nyakomitta, P. S., Nyangaresi, V. O., & Ogara, S. O. (2021). Efficient authentication algorithm for secure remote access in wireless sensor networks. Journal of Computer Science Research, 3(4), 43-50.

[79] Ardhana, V. Y. P., Sapi'i, M., Hasbullah, H., & Sampetoding, E. A. (2022). Web-Based Library Information System Using Rapid Application Development (RAD) Method at Qamarul Huda University. The IJICS (International Journal of Informatics and Computer Science), 6(1), 43-50.

[80] Möller, D. P., Vakilzadian, H., & Haas, R. E. (2022, May). Cybersecurity Certificate in Digital Transformation. In 2022 IEEE International Conference on Electro Information Technology (eIT) (pp. 556-561). IEEE.

[81] Lezzi, M., Lazoi, M., & Corallo, A. (2018). Cybersecurity for Industry 4.0 in the current literature: A reference framework. Computers in Industry, 103, 97-110.

[82] Kitchin, R., & Dodge, M. (2019). The (in) security of smart cities: Vulnerabilities, risks, mitigation, and prevention. Journal of urban technology, 26(2), 47-65.

[83] Nyangaresi, V. O. (2021). A Formally Verified Authentication Scheme for mmWave Heterogeneous Networks. In the 6th International Conference on Combinatorics, Cryptography, Computer Science and Computation (605-612).

[84] Khan, R. A., Khan, S. U., Khan, H. U., & Ilyas, M. (2021). Systematic mapping study on security approaches in secure software engineering. IEEE Access, 9, 19139-19160.

[85] Patil, B. P., Kharade, K. G., & Kamat, R. K. (2020). Investigation on data security threats & solutions. International Journal of Innovative Science and Research Technology, 5(1), 79-83.

[86] Mirakhorli, M., Galster, M., & Williams, L. (2020). Understanding software security from design to deployment. ACM SIGSOFT Software Engineering Notes, 45(2), 25-26.

[87] Adkins, H., Beyer, B., Blankinship, P., Lewandowski, P., Oprea, A., & Stubblefield, A. (2020). Building secure and reliable systems: best practices for designing, implementing, and maintaining systems. O'Reilly Media.

[88] Abduljabbar, Z. A., Nyangaresi, V. O., Ma, J., Al Sibahee, M. A., Khalefa, M. S., & Honi, D. G. (2022, September). MAC-Based Symmetric Key Protocol for Secure Traffic Forwarding in Drones. In Future Access Enablers for Ubiquitous and Intelligent Infrastructures: 6th EAI International Conference, FABULOUS 2022, Virtual Event, May 4, 2022, Proceedings (pp. 16-36). Cham: Springer International Publishing.

[89] Brass, I., & Sowell, J. H. (2021). Adaptive governance for the Internet of Things: Coping with emerging security risks. Regulation & Governance, 15(4), 1092-1110.

[90] Udvaros, J., Forman, N., & Avornicului, S. M. (2023). Agile Storyboard and Software Development Leveraging Smart Contract Technology in Order to Increase Stakeholder Confidence. Electronics, 12(2), 426.

[91] Nguyen, D. T., Song, C., Qian, Z., Krishnamurthy, S. V., Colbert, E. J., & McDaniel, P. (2018, December). IotSan: Fortifying the safety of IoT systems. In Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies (pp. 191-203).

[92] Assal, H., & Chiasson, S. (2018, August). Security in the Software Development Lifecycle. In SOUPS@ USENIX Security Symposium (pp. 281-296).

[93] Nyangaresi, V. O., & Petrovic, N. (2021, July). Efficient PUF based authentication protocol for internet of drones. In 2021 International Telecommunications Conference (ITC-Egypt) (pp. 1-4). IEEE.

[94] Cheng, L., Liu, F., & Yao, D. (2017). Enterprise data breach: causes, challenges, prevention, and future directions. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 7(5), e1211.

[95] Plant, O. H., van Hillegersberg, J., & Aldea, A. (2022). Rethinking IT governance: Designing a framework for mitigating risk and fostering internal control in a DevOps environment. International Journal of Accounting Information Systems, 45, 100560.

[96] Mughal, A. A. (2022). Building and Securing the Modern Security Operations Center (SOC). International Journal of Business Intelligence and Big Data Analytics, 5(1), 1-15.

[97] Somasekaram, P., Calinescu, R., & Buyya, R. (2022). High-availability clusters: A taxonomy, survey, and future directions. Journal of Systems and Software, 187, 111208.

[98] Alsamhi, S. H., Shvetsov, A. V., Kumar, S., Shvetsova, S. V., Alhartomi, M. A., Hawbani, A., ... & Nyangaresi, V. O. (2022). UAV computing-assisted search and rescue mission framework for disaster and harsh environment mitigation. Drones, 6(7), 154.

[99] Rao, S. P., Chen, H. Y., & Aura, T. (2023). Threat modeling framework for mobile communication systems. Computers & Security, 125, 103047.

[100] Wong, A. Y., Chekole, E. G., Ochoa, M., & Zhou, J. (2023). On the Security of Containers: Threat Modeling, Attack Analysis, and Mitigation Strategies. Computers & Security, 128, 103140.

[101] Khalil, S. M., Bahsi, H., Ochieng'Dola, H., Korõtko, T., McLaughlin, K., & Kotkas, V. (2023). Threat Modeling of Cyber-Physical Systems-A Case Study of a Microgrid System. Computers & Security, 124, 102950.

[102] Kunz, I., Weiss, K., Schneider, A., & Banse, C. (2023). Privacy Property Graph: Towards Automated Privacy Threat Modeling via Static Graph-based Analysis. Proceedings on Privacy Enhancing Technologies, 2, 171-187.

[103] Abdulkadhim, F. G., Yi, Z., Tang, C., Onaizah, A. N., & Ahmed, B. (2023). Design and development of a hybrid (SDN+ SOM) approach for enhancing security in VANET. Applied Nanoscience, 13(1), 799-810.

[104] Nyangaresi, V. O., & Alsamhi, S. H. (2021, October). Towards secure traffic signaling in smart grids. In 2021 3rd Global Power, Energy and Communication Conference (GPECOM) (pp. 196-201). IEEE.

[105] Brauneck, A., Schmalhorst, L., Kazemi Majdabadi, M. M., Bakhtiari, M., Völker, U., Baumbach, J., ... & Buchholtz, G. (2023). Federated Machine Learning, Privacy-Enhancing Technologies, and Data Protection Laws in Medical Research: Scoping Review. Journal of Medical Internet Research, 25, e41588.

[106] George, A. S., & Sagayarajan, S. (2023). Securing Cloud Application Infrastructure: Understanding the Penetration Testing Challenges of IaaS, PaaS, and SaaS Environments. Partners Universal International Research Journal, 2(1), 24-34.

[107] Zahid, S., Mazhar, M. S., Abbas, S. G., Hanif, Z., Hina, S., & Shah, G. A. (2023). Threat modeling in smart firefighting systems: Aligning MITRE ATT&CK matrix and NIST security controls. Internet of Things, 22, 100766.

[108] Whitney, M., Lipford, H. R., Chu, B., & Thomas, T. (2018). Embedding secure coding instruction into the ide: Complementing early and intermediate cs courses with eside. Journal of Educational Computing Research, 56(3), 415-438.

[109] Hussain, M. A., Hussien, Z. A., Abduljabbar, Z. A., Ma, J., Al Sibahee, M. A., Hussain, S. A., Nyangaresi V.O., & Jiao, X. (2022). Provably throttling SQLI using an enciphering query and secure matching. Egyptian Informatics Journal, 23(4), 145-162.

[110] Kwon, B. W., Sharma, P. K., & Park, J. H. (2019). CCTV-based multi-factor authentication system. Journal of Information Processing Systems, 15(4), 904-919.

[111] Mahmood, G. S., Huang, D. J., & Jaleel, B. A. (2019). A secure cloud computing system by using encryption and access control model. Journal of Information Processing Systems, 15(3), 538-549.

[112] Alhassan, M. M., & Adjei-Quaye, A. (2017). Information security in an organization. International Journal of Computer (IJC), 24(1), 100-116.

[113] Annane, B., Alti, A., Laouamer, L., & Reffad, H. (2022). Cx-CP-ABE: Context-aware attribute-based access control schema and blockchain technology to ensure scalable and efficient health data privacy. Security and Privacy, 5(5), e249.

[114] Nyangaresi, V. O., & Mohammad, Z. (2022, June). Session Key Agreement Protocol for Secure D2D Communication. In The Fifth International Conference on Safety and Security with IoT: SaSeIoT 2021 (pp. 81-99). Cham: Springer International Publishing.

[115] Susmitha, C., Srineeharika, S., Laasya, K. S., Kannaiah, S. K., & Bulla, S. (2023, February). Hybrid Cryptography for Secure File Storage. In 2023 7th International Conference on Computing Methodologies and Communication (ICCMC) (pp. 1151-1156). IEEE.

[116] Mughal, A. A. (2018). The Art of Cybersecurity: Defense in Depth Strategy for Robust Protection. International Journal of Intelligent Automation and Computing, 1(1), 1-20.

[117] He, W., Golla, M., Padhi, R., Ofek, J., Dürmuth, M., Fernandes, E., & Ur, B. (2018, August). Rethinking Access Control and Authentication for the Home Internet of Things (IoT). In USENIX Security Symposium (pp. 255-272).

[118] Al Shebli, H. M. Z., & Beheshti, B. D. (2018, May). A study on penetration testing process and tools. In 2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT) (pp. 1-7). IEEE.

[119] Shinde, N., & Kulkarni, P. (2021). Cyber incident response and planning: a flexible approach. Computer Fraud & Security, 2021(1), 14-19.

[120] Nyangaresi, V. O., & Ma, J. (2022, June). A Formally Verified Message Validation Protocol for Intelligent IoT E-Health Systems. In 2022 IEEE World Conference on Applied Intelligence and Computing (AIC) (pp. 416-422). IEEE.

[121] Saleem, J., & Hammoudeh, M. (2018). Defense methods against social engineering attacks. Computer and network security essentials, 603-618.

[122] Vayena, E., Blasimme, A., & Cohen, I. G. (2018). Machine learning in medicine: addressing ethical challenges. PLoS medicine, 15(11), e1002689.

[123] Santos, J. C., Tarrit, K., & Mirakhorli, M. (2017, April). A catalog of security architecture weaknesses. In 2017 IEEE International Conference on Software Architecture Workshops (ICSAW) (pp. 220-223). IEEE.

[124] Zimmermann, M., Staicu, C. A., Tenny, C., & Pradel, M. (2019, August). Small World with High Risks: A Study of Security Threats in the npm Ecosystem. In USENIX security symposium (Vol. 17).

[125] Bertino, E., & Islam, N. (2017). Botnets and internet of things security. Computer, 50(2), 76-79.

[126] Abduljabbar, Z. A., Abduljaleel, I. Q., Ma, J., Al Sibahee, M. A., Nyangaresi, V. O., Honi, D. G., ... & Jiao, X. (2022). Provably secure and fast color image encryption algorithm based on s-boxes and hyperchaotic map. IEEE Access, 10, 26257-26270.

[127] Appiah, V., Nti, I. K., & Nyarko-Boateng, O. (2017). Investigating websites and web application vulnerabilities: Webmaster's perspective. Int. J. Appl. Inf. Syst, 12(3), 1015.

[128] Khan, A., Ahmad, A., Ahmed, M., Sessa, J., & Anisetti, M. (2022). Authorization schemes for internet of things: requirements, weaknesses, future challenges and trends. Complex & Intelligent Systems, 8(5), 3919-3941.

[129] Waked, L., Mannan, M., & Youssef, A. (2020). The sorry state of TLS security in enterprise interception appliances. Digital Threats: Research and Practice, 1(2), 1-26.

[130] Stasinopoulos, A., Ntantogian, C., & Xenakis, C. (2019). Commix: automating evaluation and exploitation of command injection vulnerabilities in web applications. International Journal of Information Security, 18, 49-72.

[131] Nyangaresi, V. O. (2022, July). Provably Secure Pseudonyms based Authentication Protocol for Wearable Ubiquitous Computing Environment. In 2022 International Conference on Inventive Computation Technologies (ICICT) (pp. 1-6). IEEE.

[132] Zhang, M., Zhang, Y., Zhang, L., Liu, C., & Khurshid, S. (2018, September). DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (pp. 132-142).

[133] Jiang, X., Lora, M., & Chattopadhyay, S. (2020). An experimental analysis of security vulnerabilities in industrial IoT devices. ACM Transactions on Internet Technology (TOIT), 20(2), 1-24.

[134] Salahdine, F., & Kaabouch, N. (2019). Social engineering attacks: A survey. Future Internet, 11(4), 89.

[135] Khan, R. A., Khan, S. U., Khan, H. U., & Ilyas, M. (2022). Systematic literature review on security risks and its practices in secure software development. ieee Access, 10, 5456-5481.

[136] Al Sibahee, M. A., Abdulsada, A. I., Abduljabbar, Z. A., Ma, J., Nyangaresi, V. O., & Umran, S. M. (2021). Lightweight, Secure, Similar-Document Retrieval over Encrypted Data. Applied Sciences, 11(24), 12040.

[137] Al-Mhiqani, M. N., Ahmad, R., Zainal Abidin, Z., Yassin, W., Hassan, A., Abdulkareem, K. H., ... & Yunos, Z. (2020). A review of insider threat detection: classification, machine learning techniques, datasets, open challenges, and recommendations. Applied Sciences, 10(15), 5208.

[138] Pang, Z., Fu, Y., Guo, H., & Sun, J. (2023). Analysis of stealthy false data injection attacks against networked control systems: Three case studies. Journal of Systems Science and Complexity, 1-16.

[139] Pagey, R., Mannan, M., & Youssef, A. (2023, April). All Your Shops Are Belong to Us: Security Weaknesses in E-commerce Platforms. In Proceedings of the ACM Web Conference 2023 (pp. 2144-2154).

[140] Ahlawat, D. (2023). Automating Security Test-cases using DevSecOps approach for AWS Serverless application with WebSockets (Doctoral dissertation, Dublin, National College of Ireland).

[141] Nyangaresi, V. O. (2022). Lightweight anonymous authentication protocol for resource-constrained smart home devices based on elliptic curve cryptography. Journal of Systems Architecture, 133, 102763.

[142] Michailidis, E. T., & Vouyioukas, D. (2022). A review on software-based and hardware-based authentication mechanisms for the Internet of Drones. Drones, 6(2), 41.

[143] Sodhro, A. H., Awad, A. I., van de Beek, J., & Nikolakopoulos, G. (2022). Intelligent authentication of 5G healthcare devices: A survey. Internet of Things, 100610.

[144] Khan, A., Ahmad, A., Ahmed, M., Sessa, J., & Anisetti, M. (2022). Authorization schemes for internet of things: requirements, weaknesses, future challenges and trends. Complex & Intelligent Systems, 8(5), 3919-3941.

[145] García-Teodoro, P., Camacho, J., Maciá-Fernández, G., Gómez-Hernández, J. A., & López-Marín, V. J. (2022). A novel zero-trust network access control scheme based on the security profile of devices and users. Computer Networks, 212, 109068.

[146] Nyakomitta, S. P., & Omollo, V. (2014). Biometric-Based Authentication Model for E-Card Payment Technology. IOSR Journal of Computer Engineering (IOSRJCE), 16(5), 137-44.

[147] Zhang, J., Beresford, A. R., & Kollmann, S. A. (2019, July). Libid: reliable identification of obfuscated third-party android libraries. In Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 55-65).

[148] Assegie, T. A., & Nair, P. S. (2019). A review on software defined network security risks and challenges. TELKOMNIKA (Telecommunication Computing Electronics and Control), 17(6), 3168-3174.

[149] Wang, L., Jajodia, S., Singhal, A., Singhal, A., & Ou, X. (2017). Security risk analysis of enterprise networks using probabilistic attack graphs (pp. 53-73). Springer International Publishing.

[150] Nyangaresi, V. O. (2021, September). Lightweight key agreement and authentication protocol for smart homes. In 2021 IEEE AFRICON (pp. 1-6). IEEE.

[151] Diro, A., Reda, H., Chilamkurti, N., Mahmood, A., Zaman, N., & Nam, Y. (2020). Lightweight authenticated-encryption scheme for internet of things based on publish-subscribe communication. IEEE Access, 8, 60539-60551.

[152] Peixoto, M., Ferreira, D., Cavalcanti, M., Silva, C., Vilela, J., Araújo, J., & Gorschek, T. (2023). The perspective of Brazilian software developers on data privacy. Journal of Systems and Software, 195, 111523.

[153] Mohamad, M., Jolak, R., Askerdal, Ö., Steghöfer, J. P., & Scandariato, R. (2023). CASCADE: An Asset-driven Approach to Build Security Assurance Cases for Automotive Systems. ACM Transactions on Cyber-Physical Systems, 7(1), 1-26.

[154] Umeugo, W. (2023). Secure software development lifecycle: a case for adoption in software SMES. International Journal of Advanced Research in Computer Science, 14(1).

[155] Naghdipour, A., Hasheminejad, S. M. H., & Barmaki, R. L. (2023). Software design pattern selection approaches: A systematic literature review. Software: Practice and Experience, 53(4), 1091-1122.

[156] Larios-Vargas, E., Elazhary, O., Yousefi, S., Lowlind, D., Vliek, M. L., & Storey, M. A. (2023). DASP: A Framework for Driving the Adoption of Software Security Practices. IEEE Transactions on Software Engineering.

[157] Siriwardena, P., & Siriwardena, P. (2020). Designing Security for APIs. Advanced API Security: OAuth 2.0 And Beyond, 33-67.

[158] Omollo VN, Musyoki S. Global Positioning System Based Routing Algorithm for Adaptive Delay Tolerant Mobile Adhoc Networks. International Journal of Computer and Communication System Engineering. 2015 May 11, 2(3): 399-406.

[159] Khalil, S. M., Bahsi, H., Ochieng'Dola, H., Korõtko, T., McLaughlin, K., & Kotkas, V. (2023). Threat Modeling of Cyber-Physical Systems-A Case Study of a Microgrid System. Computers & Security, 124, 102950.

[160] Bastías, O. A., Díaz, J., & López Fenner, J. (2023). Exploring the Intersection between Software Maintenance and Machine Learning—A Systematic Mapping Study. Applied Sciences, 13(3), 1710.

[161] Tejay, G. P., & Mohammed, Z. A. (2023). Cultivating security culture for information security success: A mixed-methods study based on anthropological perspective. Information & Management, 60(3), 103751.

[162] Algarni, A., Attaallah, A., Eassa, F., Khemakhem, M., Jambi, K., Aljihani, H., ... & Albalwy, F. (2023). A security testing mechanism for detecting attacks in distributed software applications using blockchain. PloS one, 18(1), e0280038.

[163] Selimoglu, S. K., & Saldi, M. H. (2023). Blockchain Technology for Internal Audit in Cyber Security Governance of Banking Sector in Turkey: A SWOT Analysis. In Contemporary Studies of Risks in Emerging Technology, Part B (pp. 23-55). Emerald Publishing Limited.

[164] Nyangaresi, V. O., Ahmad, M., Alkhayyat, A., & Feng, W. (2022). Artificial neural network and symmetric key cryptography based verification protocol for 5G enabled Internet of Things. Expert Systems, 39(10), e13126.

[165] Tabrizchi, H., & Kuchaki Rafsanjani, M. (2020). A survey on security challenges in cloud computing: issues, threats, and solutions. The journal of supercomputing, 76(12), 9493-9532.

[166] Farooqui, M. N. I., Arshad, J., & Khan, M. M. (2022). A Layered Approach to Threat Modeling for 5G-Based Systems. Electronics, 11(12), 1819.

[167] Battina, D. S. (2017). Best Practices for Ensuring Security in Devops: A Case Study Approach. International Journal of Innovations in Engineering Research and Technology, 4(11), 38-45.

[168] Baum, T., Liskin, O., Niklas, K., & Schneider, K. (2016, August). A faceted classification scheme for change-based industrial code review processes. In 2016 IEEE International conference on software quality, reliability and security (QRS) (pp. 74-85). IEEE.

[169] Heiding, F., Süren, E., Olegård, J., & Lagerström, R. (2023). Penetration testing of connected households. Computers & Security, 126, 103067.

[170] Braghin, C., Lilli, M., & Riccobene, E. (2023). A model-based approach for vulnerability analysis of IoT security protocols: The Z-Wave case study. Computers & Security, 127, 103037.

[171] Omollo VN, Musyoki S. Blue bugging Java Enabled Phones via Bluetooth Protocol Stack Flaws. International Journal of Computer and Communication System Engineering. 2015 Jun 9, 2 (4):608-613.

[172] Larios-Vargas, E., Elazhary, O., Yousefi, S., Lowlind, D., Vliek, M. L., & Storey, M. A. (2023). DASP: A Framework for Driving the Adoption of Software Security Practices. IEEE Transactions on Software Engineering.

[173] Ahmed, Z., & Francis, S. C. (2019, November). Integrating security with devsecops: Techniques and challenges. In 2019 International Conference on Digitization (ICD) (pp. 178-182). IEEE.

[174] Yaacoub, J. P. A., Noura, H. N., Salman, O., & Chehab, A. (2022). Robotics cyber security: Vulnerabilities, attacks, countermeasures, and recommendations. International Journal of Information Security, 1-44.

[175] Khan, S., & Parkinson, S. (2018). Review into state of the art of vulnerability assessment using artificial intelligence. Guide to Vulnerability Analysis for Computer Networks and Systems: An Artificial Intelligence Approach, 3-32.

[176] Abiodun, O. I., Alawida, M., Omolara, A. E., & Alabdulatif, A. (2022). Data provenance for cloud forensic investigations, security, challenges, solutions and future perspectives: A survey. Journal of King Saud University-Computer and Information Sciences.

[177] Nyangaresi, V. O. (2022). Terminal independent security token derivation scheme for ultra-dense IoT networks. Array, 15, 100210.

[178] Tauqeer, O. B., Jan, S., Khadidos, A. O., Khadidos, A. O., Khan, F. Q., & Khattak, S. (2021). Analysis of security testing techniques. Intelligent Automation & Soft Computing, 29(1), 291-306.

[179] di Biase, M., Bruntink, M., & Bacchelli, A. (2016, October). A security perspective on code review: The case of chromium. In 2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM) (pp. 21-30). IEEE.

[180] Casola, V., De Benedictis, A., Rak, M., & Villano, U. (2020). A novel Security-by-Design methodology: Modeling and assessing security by SLAs with a quantitative approach. Journal of Systems and Software, 163, 110537.

[181] Tan, V., Cheh, C., & Chen, B. (2021, October). From Application Security Verification Standard (ASVS) to Regulation Compliance: A Case Study in Financial Services Sector. In 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) (pp. 69-76). IEEE.

[182] Thomas, T. W., Tabassum, M., Chu, B., & Lipford, H. (2018, April). Security during application development: An application security expert perspective. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (pp. 1-12).

[183] Mutlaq, K. A. A., Nyangaresi, V. O., Omar, M. A., & Abduljabbar, Z. A. (2022, October). Symmetric Key Based Scheme for Verification Token Generation in Internet of Things Communication Environment. In Applied Cryptography in Computer and Communications: Second EAI International Conference, AC3 2022, Virtual Event, May 14-15, 2022, Proceedings (pp. 46-64). Cham: Springer Nature Switzerland.

[184] Mughal, A. A. (2020). Cyber Attacks on OSI Layers: Understanding the Threat Landscape. Journal of Humanities and Applied Science Research, 3(1), 1-18.

[185] Acar, Y., Backes, M., Bugiel, S., Fahl, S., McDaniel, P., & Smith, M. (2016, May). Sok: Lessons learned from android security research for appified software platforms. In 2016 IEEE Symposium on Security and Privacy (SP) (pp. 433-451). IEEE.

[186] Meng, N., Nagy, S., Yao, D., Zhuang, W., & Argoty, G. A. (2018, May). Secure coding practices in java: Challenges and vulnerabilities. In Proceedings of the 40th International Conference on Software Engineering (pp. 372-383).

[187] Gasiba, T. E., Lechner, U., Pinto-Albuquerque, M., & Mendez, D. (2021, May). Is secure coding education in the industry needed? An investigation through a large scale survey. In 2021 IEEE/ACM 43rd International

Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET) (pp. 241-252). IEEE.

[188] Nyangaresi, V. O. (2022). A Formally Validated Authentication Algorithm for Secure Message Forwarding in Smart Home Networks. SN Computer Science, 3(5), 364.

[189] de Almeida, M. G., & Canedo, E. D. (2022). Authentication and authorization in microservices architecture: A systematic literature review. Applied Sciences, 12(6), 3023.

[190] Zikratov, I., Kuzmin, A., Akimenko, V., Niculichev, V., & Yalansky, L. (2017, April). Ensuring data integrity using blockchain technology. In 2017 20th Conference of Open Innovations Association (FRUCT) (pp. 534-539). IEEE.

[191] Stach, C., Gritti, C., Bräcker, J., Behringer, M., & Mitschang, B. (2022). Protecting Sensitive Data in the Information Age: State of the Art and Future Prospects. Future Internet, 14(11), 302.

[192] Sodanil, M., Quirchmayr, G., Porrawatpreyakorn, N., & Tjoa, A. M. (2015, July). A knowledge transfer framework for secure coding practices. In 2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE) (pp. 120-125). IEEE.

[193] Abood, E. W., Hussien, Z. A., Kawi, H. A., Abduljabbar, Z. A., Nyangaresi, V. O., Ma, J., ... & Ahmad, S. (2023). Provably secure and efficient audio compression based on compressive sensing. International Journal of Electrical & Computer Engineering (2088-8708), 13(1).

[194] Cremers, C., Jacomme, C., & Naska, A. (2023, August). Formal Analysis of Session-Handling in Secure Messaging: Lifting Security from Sessions to Conversations. In Usenix Security.

[195] Reyes, A., Jimeno, M., & Villanueva-Polanco, R. (2023). Continuous and Secure Integration Framework for Smart Contracts. Sensors, 23(1), 541.

[196] Chatziamanetoglou, D., & Rantos, K. (2023). Blockchain-Based Security Configuration Management for ICT Systems. Electronics, 12(8), 1879.

[197] Riegler, M., Sametinger, J., Vierhauser, M., & Wimmer, M. (2023). A model-based mode-switching framework based on security vulnerability scores. Journal of Systems and Software, 200, 111633.

[198] Nyangaresi, V. O. (2021). Hardware assisted protocol for attacks prevention in ad hoc networks. In Emerging Technologies in Computing: 4th EAI/IAER International Conference, iCETiC 2021, Virtual Event, August 18–19, 2021, Proceedings 4 (pp. 3-20). Springer International Publishing.

[199] Rawal, B. S., Manogaran, G., & Peter, A. (2022). Conduct Security Control Testing. In Cybersecurity and Identity Access Management (pp. 181-191). Singapore: Springer Nature Singapore.

[200] Khripunov, I. (2023). Bringing Safety-Security Culture into Harmony. In Human Factor in Nuclear Security: Establishing and Optimizing Security Culture (pp. 83-97). Cham: Springer International Publishing.

[201] Humayun, M., Niazi, M., Assiri, M., & Haoues, M. (2023). Secure Global Software Development: A Practitioners' Perspective. Applied Sciences, 13(4), 2465.

[202] Reeder, R. W., Ion, I., & Consolvo, S. (2017). 152 simple steps to stay safe online: Security advice for non-tech-savvy users. IEEE Security & Privacy, 15(5), 55-64.

[203] Abood, E. W., Abdullah, A. M., Al Sibahe, M. A., Abduljabbar, Z. A., Nyangaresi, V. O., Kalafy, S. A. A., & Ghrabta, M. J. J. (2022). Audio steganography with enhanced LSB method for securing encrypted text with bit cycling. Bulletin of Electrical Engineering and Informatics, 11(1), 185-194.

[204] Mihelič, A., Hovelja, T., & Vrhovec, S. (2023). Identifying Key Activities, Artifacts and Roles in Agile Engineering of Secure Software with Hierarchical Clustering. Applied Sciences, 13(7), 4563.

[205] Tang, J., Li, R., Wang, K., Gu, X., & Xu, Z. (2020). A novel hybrid method to analyze security vulnerabilities in android applications. Tsinghua Science and Technology, 25(5), 589-603.

[206] Bidgoly, A. J. (2020). Robustness verification of soft security systems. Journal of Information Security and Applications, 55, 102632.

[207] Rouland, Q., Hamid, B., & Jaskolka, J. (2021). Specification, detection, and treatment of STRIDE threats for software components: Modeling, formal methods, and tool support. Journal of Systems Architecture, 117, 102073.

[208] Nyangaresi, V. O., & Mohammad, Z. (2021, July). Privacy preservation protocol for smart grid networks. In 2021 International Telecommunications Conference (ITC-Egypt) (pp. 1-4). IEEE.

[209] Abrahamsson, P., Botterweck, G., Ghanbari, H., Jaatun, M. G., Kettunen, P., Mikkonen, T. J., ... & Wang, X. (2020). Towards a secure devops approach for cyber-physical systems: An industrial perspective. International Journal of Systems and Software Security and Protection (IJSSSP), 11(2), 38-57.

[210] Nifakos, S., Chandramouli, K., Nikolaou, C. K., Papachristou, P., Koch, S., Panaousis, E., & Bonacina, S. (2021). Influence of human factors on cyber security within healthcare organisations: A systematic review. Sensors, 21(15), 5119.

[211] Qamar, S., Anwar, Z., Rahman, M. A., Al-Shaer, E., & Chu, B. T. (2017). Data-driven analytics for cyber-threat intelligence and information sharing. Computers & Security, 67, 35-58.

[212] Honi, D. G., Ali, A. H., Abduljabbar, Z. A., Ma, J., Nyangaresi, V. O., Mutlaq, K. A. A., & Umran, S. M. (2022, December). Towards Fast Edge Detection Approach for Industrial Products. In 2022 IEEE 21st International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS) (pp. 239-244). IEEE.

[213] Dissanayake, N., Jayatilaka, A., Zahedi, M., & Babar, M. A. (2022). Software security patch management-A systematic literature review of challenges, approaches, tools and practices. Information and Software Technology, 144, 106771.

[214] Mugarza, I., Flores, J. L., & Montero, J. L. (2020). Security issues and software updates management in the industrial internet of things (iiot) era. Sensors, 20(24), 7160.

[215] Li, F., Rogers, L., Mathur, A., Malkin, N., & Chetty, M. (2019, November). Keepers of the machines: examining how system administrators manage software updates. In Proceedings of the Fifteenth USENIX Conference on Usable Privacy and Security (pp. 273-288). USENIX Association.

[216] 41Diéguez, M., Cares, C., Cachero, C., & Hochstetter, J. (2023). MASISCo—Methodological Approach for the Selection of Information Security Controls. Applied Sciences, 13(2), 1094.

[217] Nyangaresi, V. O. (2021, November). Provably secure protocol for 5G HetNets. In 2021 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS) (pp. 17-22). IEEE.

[218] Wen, S. F. (2018, January). Learning secure programming in open source software communities: a socio-technical view. In Proceedings of the 6th International Conference on Information and Education Technology (pp. 25-32).

[219] Malatji, M., Von Solms, S., & Marnewick, A. (2019). Socio-technical systems cybersecurity framework. Information & Computer Security.

[220] Paja, E., Dalpiaz, F., & Giorgini, P. (2015). Modelling and reasoning about security requirements in socio-technical systems. Data & Knowledge Engineering, 98, 123-143.

[221] Al Sabbagh, B., & Kowalski, S. (2015). A socio-technical framework for threat modeling a software supply chain. IEEE Security & Privacy, 13(4), 30-39.

[222] Li, T., Horkoff, J., & Mylopoulos, J. (2018). Holistic security requirements analysis for socio-technical systems. Software & Systems Modeling, 17, 1253-1285.

[223] Wu, P. P. Y., Fookes, C., Pitchforth, J., & Mengersen, K. (2015). A framework for model integration and holistic modelling of socio-technical systems. Decision Support Systems, 71, 14-27