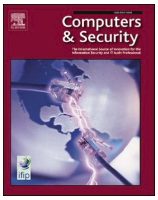
Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose



Full Length Article

Assessing of software security reliability: Dimensional security assurance techniques

Mohammad Ali [*], Ahsan Ullah, Md. Rashedul Islam, Rifat Hossain

Department of Computer Science & Engineering, World University of Bangladesh, Dhaka 1230, Bangladesh

ARTICLE INFO

Keywords:
Reliability
SSRM
STRIDE
DREAD
CVSS
Dynamic analysis
Security terms

ABSTRACT

Security plays a crucial role in ensuring the reliability of software systems, making a secure and dependable security framework vital for safeguarding software integrity. However, evaluating the dynamic and multifaceted aspects of security presents significant challenges, as various security metrics and factors complicate the assessment of reliability. This study advances the core concepts of software security through the application of security assurance techniques, including vulnerability scanning, code review, penetration testing, threat assessment, control evaluation, mitigation, risk assessment, and configuration review. In the context of the Software Security Reliability Model (SSRM), a framework was developed to enhance software security assurance across different stages. A comprehensive systematic literature review was conducted to identify security challenges, and the STRIDE and DREAD methodologies were applied to model security threats effectively. Additionally, a mathematical CVSS scoring method was utilized for risk assessment. The synthesis of diverse security methods, tools, attack patterns, and systems was analyzed, identifying 15 critical software security terms: authentication, authorization, encryption, access control, network security, application security, data security, incident response, compliance, threat intelligence, privacy protection, third-party risk, cloud security, endpoint security, and identity management. The findings highlight these terms as key contributors to improving software security reliability.

1. Introduction

Assessing software reliability involves determining how likely the software is to perform its tasks without errors within specific parameters and timeframes. Software security reliability denotes the stable and trustworthy performance of a software system in defending against security threats and sustaining its security features over an extended period. It entails confirming that the software operates reliably, securely, and devoid of unforeseen errors or weaknesses, thereby ensuring continuous protection against potential security breaches. Furthermore, heightened security correlates with enhanced user benefits, including improved usability and increased trustworthiness (Ahmad et al., 2015). In the majority of instances, the findings of software security assessments have illustrated an increasingly deteriorating landscape of vulnerabilities. The State of Software Security report Findings (2016-2023) 70 % contain flaws in third-party code and almost 71 % of organizations have security debt (Chris Eng, 2024). These figures reflect the persistent challenge organizations encounter in maintaining software security across increasingly complex ecosystems. Over 60 % of open-source codebases have security vulnerabilities, with many organizations failing to update these libraries regularly, resulting in heightened security risks and operational liabilities (SYNOPSYS, 2023). The efficacy of security approaches utilized in the software raises concerns regarding the integrity of the entire security system. Consequently, the entire software security system has reached a highly vulnerable state. Upon scrutinizing the security tasks employed, it becomes apparent that previous methods aimed to address specific threats as priorities. To ensure accurate documentation of security cases and rectify failed processes, attention is directed towards fault tolerance and coding defects (Weinstock Howard, 2007, Huang and Kintala, 1993). In the realm of security management, addressing individual issues diverts attention from others, thereby hindering comprehensive problem-solving efforts. The real-world implementation of CVSS is a critical mechanism in the ongoing endeavors of companies such as Adobe, Microsoft, and Cisco to effectively manage vulnerabilities and assess risks. CVSS enables these organizations to objectively assess the

[*] Corresponding author.
E-mail addresses: 0321563379@student.wub.edu.bd (M. Ali), ahsan.ullah@cse.wub.edu.bd (A. Ullah), 0321563380@student.wub.edu.bd (Md.R. Islam), 0321563395@student.wub.edu.bd (R. Hossain).

https://doi.org/10.1016/j.cose.2024.104230
Received 23 September 2024; Received in revised form 3 November 2024; Accepted 25 November 2024
Available online 26 November 2024

severity and exploitability of software vulnerabilities by offering a standardized scoring system. In the same vein, Microsoft integrates CVSS scores into its security bulletins, enabling customers to promptly evaluate the consequences of identified vulnerabilities and implement the requisite remedies with a well-informed sense of urgency.

Evaluating software quality entails achieving all assurance objectives, and if deemed satisfactory, the software is categorized under the Level 'A software' stage (Rushby, 2015). A comprehensive analysis of various software issues and the strategic utilization of factors help mitigate security risks. Although solutions for software security have been proposed using various methodologies, weakness still persists to a significant extent and continues to proliferate constantly (Kudriavtseva and Gadyatskaya, 2022). This research paper presents a systematic analysis by employing a methodological framework and prior approaches to security reports published over the past seven years, offering an overview of recent advancements in software security research, with a primary focus on dynamic analysis techniques. We provide an initial comprehensive overview of Dimensional Security Assurance Techniques and methodologies for Software through the review of 330 publications. Following this, we conduct an in-depth analysis of a selected subset, focusing on 49 publications from leading software engineering and security venues, where we thoroughly enumerate the objectives, techniques, and tools discussed. In previous studies, vague modeling has been used for software system assurance cases, and general methods have been used for complex assurance models (Chechik et al., 2019, Rhodes et al., 2009). A resilient framework is essential to enhance the reliability of software security, enabling effective mitigation of evolving threats and ensuring sustained service delivery (Sahu and Srivastava, 2021, Kumar et al., 2023). Dimensional security assurance techniques are used as an effective and structural approach to secure the software system against all types of threats, with an emphasis on requirements. The SSRM's implementation of real-time configuration reviews and continuous vulnerability monitoring guarantees that security measures are dynamically adjusted to address emerging risks in cloud-based environments, where resource sharing and dynamic scaling are common. Our main focus is to present software security as a more secure, advanced, and reliable model to everyone. The comprehensive process of software security threat analysis framework will be provided using the dimensional security assurance approach, considering all aspects of software security.

The article is divided into six parts. The following section provides a literature review. Methodology is provided in section 3. Results, Evaluation, Discussion and conclusion drawn are in sections 4 to 6, respectively.

2. Literature review

Through our paramount efforts, comprehensive investigation, and knowledge integration, a systematic literature review has been conducted to analyze the security assurance cases of ASR. Assurance cases are used as certifications to assessment attributes such as reliability, and security of software (Sklyar and Kharchenko, 2020). Ensuring that the software system's requirements are met and maintained in a balanced and reliable state guarantees the system's effectiveness. The security assurance system enhances confidence by providing robust security, serving as an effective defense against all types of threats (Shukla et al., 2022). Based on the level of reliability, users and organizations provide differing amounts of positive and negative feedback. The reliability of software is categorized into three parts, the primary and most important step of which is modelling (Kotaiah and Khan, 2012). Reliability assessment necessitates multiple testing methods, including penetration testing, static and dynamic analysis, and load testing. Moreover, A transformational approach is employed for the formal verification and statistical testing processes within the context of reliability assessment (Cukic and Bastani, 1997). The integration of security methods mitigates multifaceted threats and presents a consolidated overview of the security posture (Kumar and Mishra, 2024). The proposed methods, which concentrate on specific problem-centric tasks across various types of assurance, As a result fail to enhance the system's stability rate. Uncertainty quantified by statistical measures reduces overall confidence; if one individual possesses 70 % confidence, another exhibits 30 % uncertainty (Duan et al., 2014). Software assurance model will facilitate the quantification of security assurance levels and provide confidence to the pertinent organizations (Khan et al., 2022). Delivering reliable software services is regarded as a fundamental objective in the software development process. However, Despite the presence of ensemble security gaps within the software, if security is regarded as an integral component, the software is considered insecure (Khan and Khan, 2018). Any form of vulnerability within a security system can lead to significant threats. Risks associated with software assurance case data, as well as the system design and development phases, continue to demonstrate an inability to adequately address vulnerabilities (Muram and Javed, 2023, Von Solms and Futcher, 2020).Various applied patterns and assurance techniques have been used during the development phase of security to address vulnerabilities systematically through dialectical interactions. Developers improve the system by implementing secure design patterns for critical and sensitive functions and utilizing dialectical interactions and assurance techniques (Weir et al., 2021, Ahmad et al., 2015). Additionally, incorporating security practices within the SDLC establishes a comprehensive framework for the development of secure software applications (Humayun et al., 2022). The design and development phases of software applications and security systems are widely acknowledged as complex and demanding endeavors (Kumar et al., 2015, Baldassarre et al., 2020). Any slight negligence in the system's requirements, approach, and factors increases the likelihood of systems security threats. Establishing an effective and resilient development foundation is a key strategy for mitigating complex system issues (Kumar and Khan, 2024, Pandey et al., 2024). Measuring the security practices of this system creates a comprehensive picture of the entire security assurance process, allowing for the assessment of its suitability and maturity (Chandra et al., 2009). Among the critical challenges in software development, the rates of vulnerabilities are as follows: security issues (62 %), framework (49 %), and lack of confidentiality and trust (40 %) (2011–2020) (Khan et al., 2022). The rising rate of vulnerabilities in security development raises questions about the effectiveness of previously proposed solutions. To control software security, it is necessary to analyze development design principles, identify various threat agents, and detect vulnerabilities (M.Surakhi et al., 2017). To assess potential risks through the threat modelling process, tools such as sla-generator, SPARTA, and Threat Agile are used (Granata and Rak, 2024). Specifically, software specifications and mitigation trees are required for the design and implementation aimed at mitigating identified threats (Frydman et al., 2014). The threats are divided into various groups to carry out the actual incidents, processes, and activities of the attacks (Tatam et al., 2021). Proper security assessment is considered a key factor in enhancing software reliability. The structures and techniques of previous security models have been thoroughly analyzed. Within the SSDLC, the STRIDE process and risk management encompass extensive security procedures, including identifying, analysing, and evaluating risks, thus providing effective solutions to security challenges (Yuniar Banowosari and Abidzar Gifari, 2019, Tariq et al., 2023). Security Vulnerability Testing (SVT) and the meaningful measurement of the effectiveness of the security tools provided a robust security system (Parizi et al., 2018). The systematic mapping study reveals that the predominant focus of studies is on identifying, adapting, and mitigating vulnerabilities, comprising 41 % of the research (Mohammed et al., 2017). The data derived from prior studies underscore the necessity of assessing the reliability of software security. A comprehensive software security architecture recommendation is needed to explore and address various critical security issues affecting software usability (Kumar et al., 2020). The systematic literature review in our study involved a structured methodology to ensure rigorous selection of relevant security challenges. Key primary criteria were adherence to established security frameworks (e.g., ISO/IEC 27001, NIST), frequency of mention in high-impact journals, and recency. We prioritized literature that concentrated on sector-specific vulnerabilities, evolving threats, and advancements in security assurance techniques. Several trends were identified during this assessment, including the growing emphasis on cloud security vulnerabilities, AI-driven threat vectors, and the intricacies of third-party risk management in distributed software environments. Furthermore, a significant shift was observed in the integration of automation in threat response and the emphasis on adaptive security models. The proposed framework was developed with these insights in mind, ensuring that it was grounded in current security priorities and prepared for future challenges in software reliability.

3. Methodology

Evaluating software security reliability entails assessing the capability of a software system to operate correctly and securely under both normal and adverse conditions. Reliability metrics play a crucial role as a key factor in protecting software systems from overall harm. Furthermore, the metrics involved in reliability are to address security threats, maintain integrity, and provide optimal solutions to critical problems by quickly recovering from failures or attacks. The reliability of security is so robust in protecting against skilled hackers that it is considered at inter-expert levels, educational levels, and in research within the academic community and textbook authorship (Weir et al., 2021). Detailed analysis, verification, and specification of all security elements reflect a significant level of assurance. Embedding comprehensive security considerations into the SDLC enables the evaluation of system security, the correction of identified problems, and the maintenance of robust security assurance.

By integrating with the phases of the SDLC, Fig. 1. **The overall process of Software Security assurance Analysis Framework.** provides a well-structured and more secure framework for all processes of software security assurance.

The framework that is provided is designed to seamlessly integrate into the Software Development Life Cycle (SDLC) by meticulously aligning critical security assurance activities with specific SDLC phases, thereby guaranteeing comprehensive security coverage. It includes essential components such as asset identification, threat modeling, risk analysis, vulnerability assessment, mitigation planning, validation testing, and continuous monitoring. The SDLC phases to which these components are assigned are diverse. During the requirements gathering and design phases, developers and security analysts identify essential assets that require protection, as illustrated in the "Model" section. In the design and development phases, threat modeling and risk analysis are essential for the identification of potential attacks and the prioritization of risks, thereby ensuring a security-conscious system architecture. The vulnerability assessment, which is designated "Identify" in the framework, is essential for the identification of vulnerabilities during development and testing. This is achieved through the use of techniques such as static and dynamic code analysis. Mitigation planning and execution, which are denoted as "Mitigation Planning & Execution" in the framework, guarantee that vulnerabilities are resolved through security controls and upgrades during the testing and deployment phases. Validation testing, which is conducted during the testing phase, guarantees the efficacy of the security controls that have been implemented. This process is equivalent to "Validation Testing" in the framework. Security in the operational environment is guaranteed through ongoing monitoring and enhancement following deployment, which is consistent with the maintenance phase. This integration guarantees that security assurance is a continuous and iterative process throughout the SDLC, from the initial requirements stage to maintenance.

In the initial phase of the software security assurance analysis framework, a thorough and effective analysis of objectives and scope establishes the fundamental components of the system. In this Table 1 shows the goals Specification for identifying assets and dependencies:

A comprehensive list of all software components, including applications, libraries, modules, and APIs, has been created Fig. 2 with the aim of identifying assets and dependencies along with their intended outcomes and benefits.

The proposed framework provides a methodologically rigorous approach to the identification and administration of software components and their dependencies, which are essential for the maintenance of robust security throughout the software development lifecycle. The comprehensive analysis of system element interdependencies is facilitated by the emphasis on mapping interactions and data flows, which is central to this framework. This process enables security analysts to accurately identify potential vulnerabilities and implement appropriate mitigation strategies. The framework commences with the inventorying of software components and the identification of hardware and infrastructure, thereby guaranteeing a comprehensive documentation of all critical system elements. This step is crucial in the development of a comprehensive architectural comprehension, which prevents the

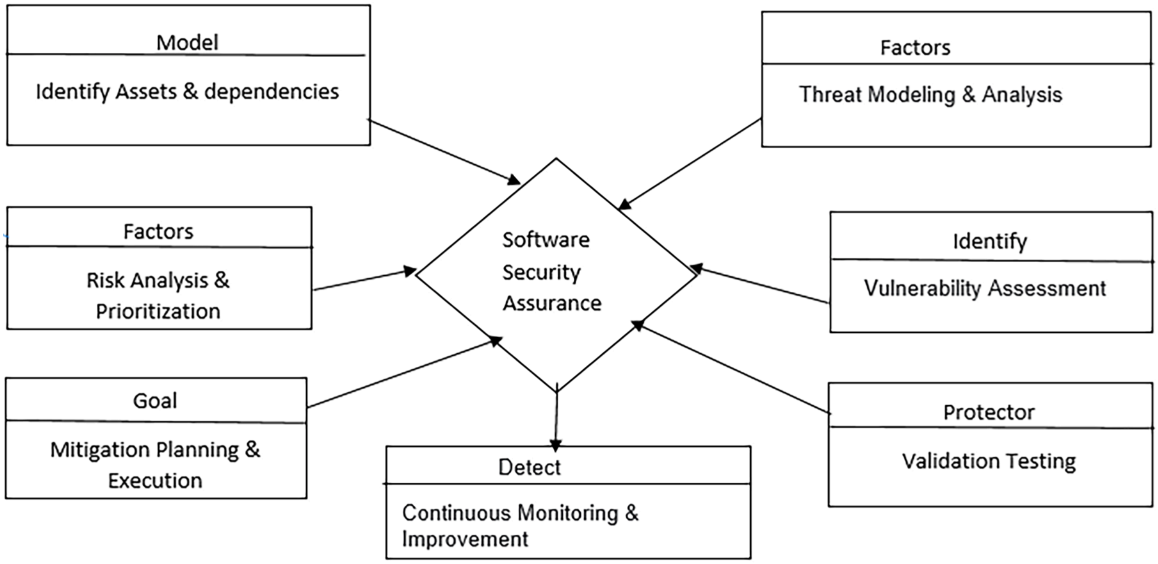

Model
Identify Assets & dependencies
Factors
Threat Modeling & Analysis
Factors
Risk Analysis & Prioritization
Software Security Assurance
Identify
Vulnerability Assessment
Goal
Mitigation Planning & Execution
Protector
Validation Testing
Detect
Continuous Monitoring & Improvement

Fig. 1. The overall process of Software Security assurance Analysis Framework.

Table 1
The components of assets & dependencies objective & scope.

Objectives	Scope
Comprehensive Threat Identification A comprehensive identification of all potential threats affecting software security has been systematically conducted.	**System Boundaries** Analysis has been conducted on every component, service, and data stream within the software system, as well as on third-party integration and external interfaces.
Security Posture Enhancement Enhancing the overall security of the software system involves prioritizing the implementation of essential security controls and mitigation strategies to effectively address the identified threats.	**Threat Landscape** In assessing the surrounding environment for potential threats, actor-based threats have been identified, such as cybercriminals and insider threats, as well as various types of threats like malware, phishing, and DDoS attacks, ensuring comprehensive analysis of both known and emerging threats.
Compliance Assurance To ensure the software's meets all relevant regulatory and industry standards, it is necessary to align security measures with specific compliance requirements such as GDPR and HIPAA, conduct regular audits, and maintain thorough documentation to demonstrate compliance efforts.	**Time Frame** A schedule has been defined for the initial analysis and regular updates of threats, and a periodic reassessment plan has been devised to incorporate new threats and changes in the environment.
Data Protection Utilizing robust encryption techniques for data in transit is essential to protect sensitive information from unauthorized access and breaches, guaranteeing secure storage and transmission infrastructures, and consistently updating access control measures to safeguard sensitive data.	**Risk Tolerance** Aiding stakeholders in decision-making regarding the identification of acceptable risk levels and the management of such risk's aids in prioritizing security measures to gain a security edge.

neglect of critical components during the security planning phase. Subsequently, the evaluation of security dependencies offers a structured evaluation of security-related external libraries and system integrations, which serves as the foundation for subsequent risk mitigation processes. The emphasis is redirected to the identification of critical data elements on the right side of the framework, which involves the identification of data assets and fluxes, as well as external dependencies. This guarantees the safeguarding of both internal and external system interactions from potential security breaches. Lastly, the documentation of assets and dependencies guarantees the establishment of a comprehensive record that facilitates ongoing security assessment and audit processes. This framework is a valuable instrument for systematic software security assurance due to its holistic and proactive approach, which incorporates both technical and operational dimensions of security management.

The documentation provides detailed information about the software system, including its versions, configurations, and roles. It identifies all data categories, including sensitive ones like PII and financial information, and the data movement process. The documentation also provides a catalog of physical hardware elements, network infrastructure, cloud services, and virtual environments that support the software system. It also identifies external systems like third-party services, APIs, and cloud services. The Fig. 3 Data Flow Diagrams (DFDs) are used to illustrate the interaction between security components and data flow dynamics. The proposed solutions address risks like supply chain attacks, dependency confusion, and typographical errors. The thorough risk analysis commences with the identification of assets and dependencies within the software system in order to evaluate the impact of dependencies on security posture. Potential vulnerabilities introduced by these dependencies are evaluated through threat modeling and analysis. Risk analysis and prioritization are instrumental in the assessment and ranking of the severity and probability of identified threats, which in turn informs the planning and execution of mitigation measures. Validation testing guarantees that the risks are effectively mitigated through the application of mitigations. The regular assessment of evolving dependencies is ensured by continuous monitoring, which maintains security assurance over time and adapts to new threats. Comprehensive security evaluation and administration are guaranteed by this structured approach. Regular updates to the documentation are part of the evaluation process. Software security threat modeling and analysis involve a systematic approach to identifying, assessing, and mitigating security risks within software systems. This structured approach helps identify potential security issues before they can be exploited by adversaries.

Threat modeling is a fundamental process that is used to identify and safeguard critical assets within software systems, including data, applications, and infrastructure. The objective of this process is to comprehend the potential consequences of a variety of hazards by classifying them into distinct categories, thereby facilitating a structured approach to risk assessment and mitigation. The STRIDE methodology is a widely used framework that classified threats into six categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. This methodical classification enables a thorough analysis of the vulnerabilities that could jeopardize the

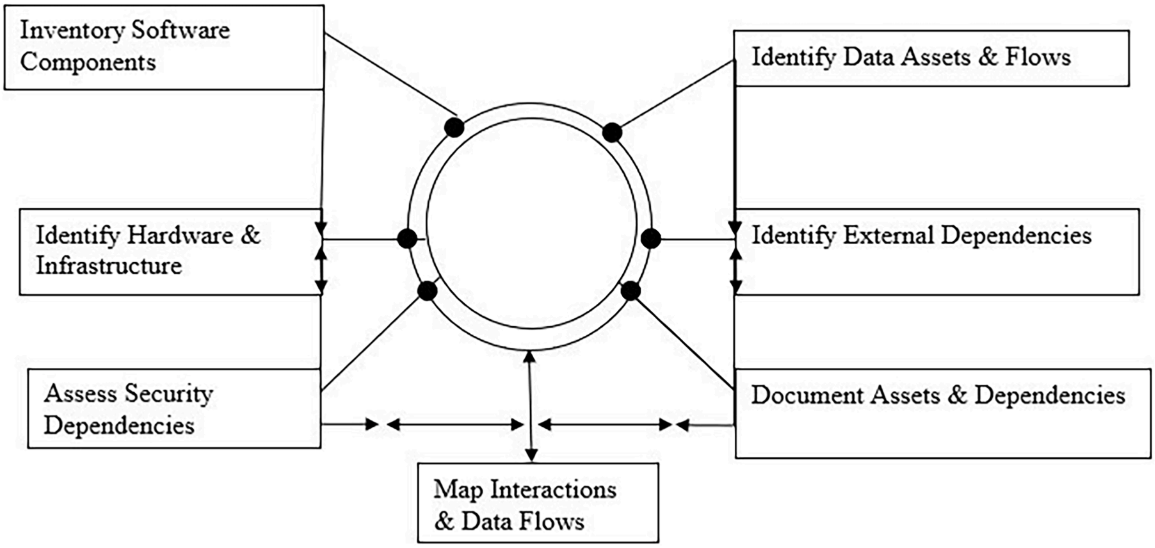

Inventory Software Components
Identify Data Assets & Flows
Identify Hardware & Infrastructure
Identify External Dependencies
Assess Security Dependencies
Document Assets & Dependencies
Map Interactions & Data Flows

Fig. 2. A comprehensive framework for identifying assets and dependencies.

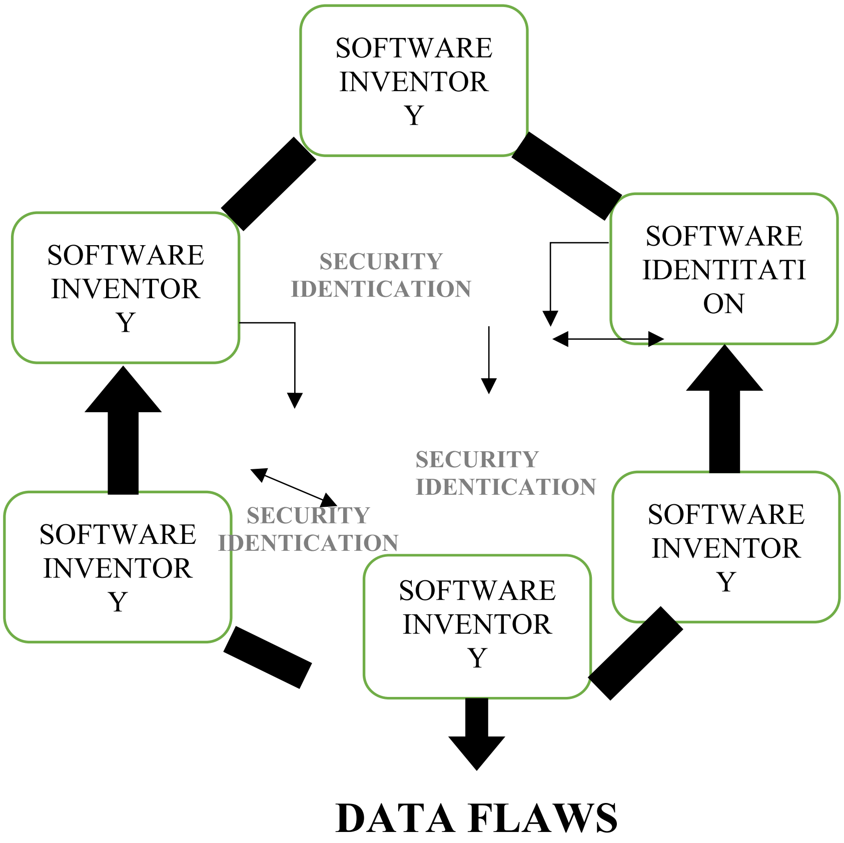


Fig. 3. Data Flow diagram (DFD).

security of these assets. Subsequently, quantitative and qualitative methodologies are implemented to assess the hazards identified by STRIDE. Quantitative methodologies, including the Common Vulnerability Scoring System (CVSS), offer a standardized scoring mechanism for evaluating the severity and probability of vulnerabilities. Simultaneously, qualitative methodologies, including expert judgment and risk matrices, are implemented to integrate contextual factors and human expertise into the evaluation of risks associated with these hazards. The final stage entails the development of mitigation strategies to mitigate these risks after the identification and assessment of threats. In order to monitor and guarantee the ongoing efficacy of these measures, these strategies typically involve the implementation of security controls, including encryption, access controls, and intrusion detection systems. Additionally, they involve regular security audits. Additionally, employee training programs are crucial for the organization to cultivate a culture of security awareness and mitigate human-related risks. In this regard, risk assessment models such as STRIDE and DREAD (which prioritize Damage, Reproducibility, Exploitability, Affected Users, and Discoverability) are essential for improving the overall security posture of a system. DREAD offers a multifaceted perspective on security risks by incorporating these dimensions, which facilitates a more effective and focused approach to risk mitigation. DREAD assists in the assessment of software security reliability by prioritizing threats according to their potential impact and probability within the broader framework of dimensional security assurance techniques. They enhance the software's resilience against prospective attacks by offering a structured framework for evaluating and responding to threats.

Attack Trees are visual diagrams that depict how an asset can be attacked, starting with a root node and branching out into various paths. They help break down complex attack scenarios into manageable parts, facilitating systematic analysis and addressing of each potential threat.

Attack Trees are a powerful tool for analyzing and visualizing potential cyber threats within software systems. Table 2 provide a structured framework for identifying and mitigating security threats. Each node within the tree delineates specific attack vectors or subdesires, demonstrating various methods through which property can be compromised. This hierarchical approach allows for a comprehensive examination of potential security breaches and aids in developing targeted mitigation techniques. For example, nodes like "Gain unauthorized access to the system," "Bypass authentication," "Exploit willing password," and "Use stolen credentials" define the technical and social engineering processes adversaries might employ. To effectively counter these threats, multi-hassle authentication and stringent password pointers are recommended. Each attack type is classified based on its nature, such as technical, human/social, network-based, or software-specific. The software risk evaluation criteria (Risk Level (High/Medium/Low), Likelihood (High/Medium/Low), and Impact (High/Medium/Low)) systematically guide the prioritization of mitigation efforts. This framework plays a crucial role in determining and focusing remediation efforts on high-chance situations where the risk and potential impact of security breaches are pronounced. High-chance situations require immediate attention and robust mitigation measures to effectively reduce primary security risks.

The middle stages of the software security threat analysis paradigm are crucial for risk analysis and prioritization. This process involves identifying and categorizing threats and vulnerabilities to maintain a secure software environment. The Identify Risks phase aims to identify and record every possible risk impacting the software system, using industry frameworks and standardized checklists. Key activities include brainstorming sessions with stakeholders, reviewing previous incident reports, and evaluating risks related to new technologies or processes. The Analyze Risks process is a systematic approach to understanding and managing identified risks, grouping hazards based on their origin and understanding their underlying reasons. The Prioritize Risks phase allocates resources to high-priority risks, while the Develop Mitigation Strategies phase develops plans for managing these risks. The Execution

Table 2
Attack tress.

Node ID	Parent Node ID	Description	Attack Type	Mitigation Strategy	Risk Level (High/Medium/ Low)	Likelihood (High/Medium/ Low)
1	-	Gain unauthorized access to the system	Root Goal	Implement strong access control	High	High
1.1	1	Bypass authentication	Technical	Multi-factor authentication	High	High
1.1.1	1.1	Exploit weak password	Technical	Enforce strong password policies	Medium	High
1.1.2	1.1	Use stolen credentials	Human/ Social	Use of account lockout machanisms	Medium	Medium
1.2	1	Exploit software vulnerability	Technical	Regular software patching	High	Medium
1.2.1	1.2	SQL Injection	Technical	Input validation	High	Medium
1.2.2	1.2	Buffer overflow	Technical	Bounds checking	Medium	Medium
2	-	Denial of Service (DoS) Attack	Root Goal	Implement redundancy and failover	High	Medium
2.1	2	Flooding attack	Network	Rate limiting and throttling	High	Medium
2.1.1	2.1	SYN Flood	Network	SYN cookies	Medium	Medium
2.1.2	2.1	UDP Flood	Network	Blocking unused ports	Medium	Medium
2.2	2	Application layer DoS	Application	Web application firewall	High	Medium
2.2.1	2.2	HTTP Flood	Application	Traffic analysis	Medium	Medium
2.2.2	2.2	Slowloris	Application	Timeouts and resource limits	Medium	Medium
3	-	Data Exfiltration	Root Goal	Data Loss Prevention (DLP)	High	High
3.1	3	Insider threat	Human	Employee training	High	Low
3.1.1	3.1	USB drive data theft	Human	Disable USB ports	Medium	Low
3.1.2	3.1	Email data leakage	Human	Email filtering	Medium	Medium
3.2	3	External attacker	Technical	Encryption of data at rest	High	Medium
3.2.1	3.2	Exploit network vulnerability	Network	Network segmentation	High	Medium
3.2.2	3.2	Man-in-the Middle attack	Network	TLS/SSL encryption	Medium	Medium

phase evaluates the viability, efficacy, and cost of each mitigation option, while the Implementation phase involves practical implementation of chosen risk mitigation techniques. The Monitor and Evaluation phase requires constant monitoring and evaluation, including security audits, IDS and SIEM systems, and a formal feedback loop. Identifying vulnerabilities is essential for software security and preventing potential exploits. Steps such as Asset Identification, Code Review and Static Analysis, Dynamic Analysis and Penetration Testing Table 3 are employed to identify security weaknesses in the source code before the software is run. This ensures the integrity, confidentiality, and availability of a security system.

Below is detailed table for advanced Dynamic Analysis and Penetration Testing:

Additionally, the improvement and usage of exploits inside managed surroundings serve to assess the efficacy of gadget defenses in opposition to recognized vulnerabilities. Automated Vulnerability Scanning additionally holds big importance, encompassing ordinary scans finished the use of automatic equipment to discover installed vulnerabilities inside software program and its related surroundings. This manner guarantees that the vulnerability scanner`s repository stays contemporary with the cutting-edge hazard intelligence. Moreover, it entails scrutinizing all aspects of the software system, encompassing third-party libraries, APIs, and configuration files.

Furthermore, Third-Party Component Analysis is a crucial method for identifying vulnerabilities in third-party additives and libraries used in software frameworks. It involves testing dependencies, cross-referencing them with databases like the National Vulnerability Database, and ensuring continuous updates. Configuration Review is also essential for identifying vulnerabilities from misconfigurations in the software and its operational environment. Risk evaluation is performed using a vulnerability monitoring machine, tracking development in remediation efforts and ensuring accountability. Validation and review are essential to verify the accuracy of findings. Mitigation planning focuses on addressing identified security vulnerabilities, prioritizing risks, and implementing the latest security protocols. A feasibility evaluation is conducted to validate the effectiveness and sustainability of these techniques. System validation checking ensures the effectiveness of safety features, identifies vulnerabilities, and ensures compliance with security requirements. It goes beyond primary checks to assess resilience in simulated attacks or high-pressure conditions. Potential vulnerabilities are diagnosed through penetration testing, user acceptance testing (UAT), and security checks. Integrating security assessments into the CI/CD infrastructure, red teaming, and spoof testing enhances validation checking's robustness. Continuous tracking and improvement are

Table 3
Advanced dynamic analysis and penetration testing.

Activity	Description	Tools/Methods	Outcome	Challenges
Dynamic Testing	Perform dynamic testing to analyze how the application behaves during execution, identifying vulnerabilities that manifest only during runtime.	Dynamic Application Security Testing (DAST), Fuzz Testing, Interactive Application Security Testing (IAST)	Identification of runtime vulnerabilities, behavior analysis under various conditions	False positives, high complexity
Penetration Testing	Conduct penetration tests to simulate real- world attacks on the system, uncovering security weaknesses that could be exploited by attackers.	Manual Testing, Automated Pen Testing Tools (e.g., Metasploit, Burp Suite), Social Engineering	Discovery of exploitable vulnerabilities, real- world attack scenario simulation	Requires skilled testers, potential disruption
Exploit Development	Develop and use exploits in a controlled environment to test the system's defenses against known vulnerabilities.	Exploit Frameworks (e.g., Metasploit), Custom Exploit Development, Sandbox Environments	Validation of vulnerabilities exploitability, evaluation of defense mechanisms' effectiveness	High skill requirement, ethical/legal concerns

essential for maintaining a strong security posture.

Finally, the levels of non-stop tracking and development had been identified as critical for reinforcing device protection. Continuous tracking and development are essential for enhancing device protection, as they ensure robust security features and adapt to evolving threats. This involves continuously evaluating protection metrics, device behaviors, and risk landscapes to identify vulnerabilities and respond to incidents. Continuous tracking is an ongoing method incorporated into an organization's security strategy, providing real-time insights into the device's protection posture. Key activities include using computerized equipment and sensors for instant anomaly detection, implementing SIEM structures, incorporating risk intelligence feeds, organizing a strong incident response plan, conducting periodic security tests, and ensuring compliance with security requirements and regulatory requirements through regular audits and reviews.

In the process of assessing the reliability of software security, we have presented the results of our analysis of relevant cases clearly and succinctly across various sections.

Security Key Terms, Phrases, Techniques, and Total Case Results:

We conducted a detailed analysis Table 4 to identify key terms, phrases, and techniques across multiple attack vectors in our comprehensive study of cyber-attack techniques. Integrating attack vectors into the Software Development Life Cycle (SDLC) is essential for proactive security management. During the design and development phases, threat modeling and vulnerability assessments can be implemented to identify attack techniques such as phishing, malware, and SQL injection at an early stage. System resilience is enhanced by methods such as input validation for SQL injection and behavior monitoring for zero-day exploits. Methods such as code evaluations and penetration testing are employed during testing to identify vulnerabilities, including XSS and MitM attacks. To guarantee that evolving vulnerabilities are addressed throughout the software lifecycle, continuous monitoring in the deployment phase detects insider threats and APTs. Phishing, for instance, encompasses terms such as spear phishing and whaling, which are frequently described by terms such as "email scam" and "credential harvesting." Social engineering and deceptive communications are among the methods employed in phishing. This comprehensive investigation examined 324 documents and identified 67 instances of deception. In the same way, malware, which has been identified as a substantial threat, is linked to essential terms such as trojans, spyware, and ransomware. In discussions of malware, which employs techniques such as polymorphic malware, fileless attacks, and drive-by downloads, phrases such as "malicious software" and "system infection" are frequently used. 74 instances of malware infections were identified in the analysis of 289 documents.

Additionally, SQL injection attacks, also known as error-based, union-based, and blind SQL injections, are often linked to database exploitation and SQL attacks. These attacks can be effectively mitigated through strategies like input validation and parameterized queries. Cross-Site Scripting (XSS), also known as script injection and client-facet

Table 4
Security key terms, phrases, techniques, and total case results.

Attack Technique	Key Terms	Phrases	Techniques	Total Documents Analyzed	Total Hits (Instances)
Phishing	Spear phishing,whaling	"Credential harvesting", "Email scam"	Social engineering, deceptive emails	324	67
Malware	Ransomware, spyware, trojan	"Malicious software", "System infection"	Polymorphic malware, fileless attacks, drive-by downloads	289	74
SQL Injection	Error-based, union-based, blind	"Database exploitation", "SQL attack"	Input validation, parameterized	195	45
Cross-Site Scripting (XSS)	Stored XSS, reflected XSS, DOM-based XSS	"Script injection", "Client-side attack"	Input sanitization, content security policy (CSP)	243	58
Man-in-the-Middle (MitM)	HTTPS spoofing, SSL stripping, session hijacking	"Data interception", "Network attack"	HTTPS implementation, public key pinning	210	52
Distributed Denial-of-Service (DDoS)	Volumetric attacks, protocol attacks, application layer attacks	"Service disruption", "Traffic flood"	Rate limiting, traffic filtering, anti-DDoS services	178	40
Zero-Day Exploits	Exploit kits, targeted attacks, unpatched vulnerabilities	"Unknown vulnerabilities", "Immediate threat"	Patch management, threat intelligence, behavior monitoring	260	65
Credential Stuffing	Botnet-driven attacks, manual attacks, account takeover	"Password reuse", "Credential breach"	Multi-factor authentication (MFA), rate limiting, credential monitoring	225	80
Insider Threats	Data theft, sabotage, privilege misuse	"Internal breach", "Employee threat"	User behavior analytics, strict access controls	190	47
Advanced Persistent Threats (APTs)	Nation-state actors, cyber espionage, stealth attacks	"Long-term compromise", "Targeted attacks"	Endpoint detection and response (EDR), network segmentation, advanced threat protection	275	70

attack, can be prevented through enter sanitization and content protection policies (CSP). Man-in-the-Middle (MitM) attacks involve HTTPS spoofing, SSL stripping, and consultation hijacking. Countermeasures for MitM attacks include HTTPS implementation and public key pinning. Distributed Denial-of-Service (DDoS) attacks, classified into volumetric, protocol, and alertness layer attacks, can be mitigated through fee limiting, visitor filtering, and anti-DDoS services. A study of 178 files found forty instances of DDoS attacks. Overall, implementing effective strategies and implementing robust security measures can help protect against these attacks and protect against potential data breaches. Zero-Day Exploits involve exploiting kits and unpatched vulnerabilities, while Credential Stuffing attacks involve password reuse and breaches. Mitigation strategies include patch management, hazard intelligence, and conduct monitoring. Insider threats involve data theft and sabotage, while Advanced Persistent Threats (APTs) involve long-term compromise and centered attacks. Mitigation techniques include endpoint detection and response (EDR), community segmentation, and superior risk protection. 65 instances of zero-day exploits were observed across 260 files. Strategies for insider threats include multi-thing authentication, price limiting, and credential monitoring.

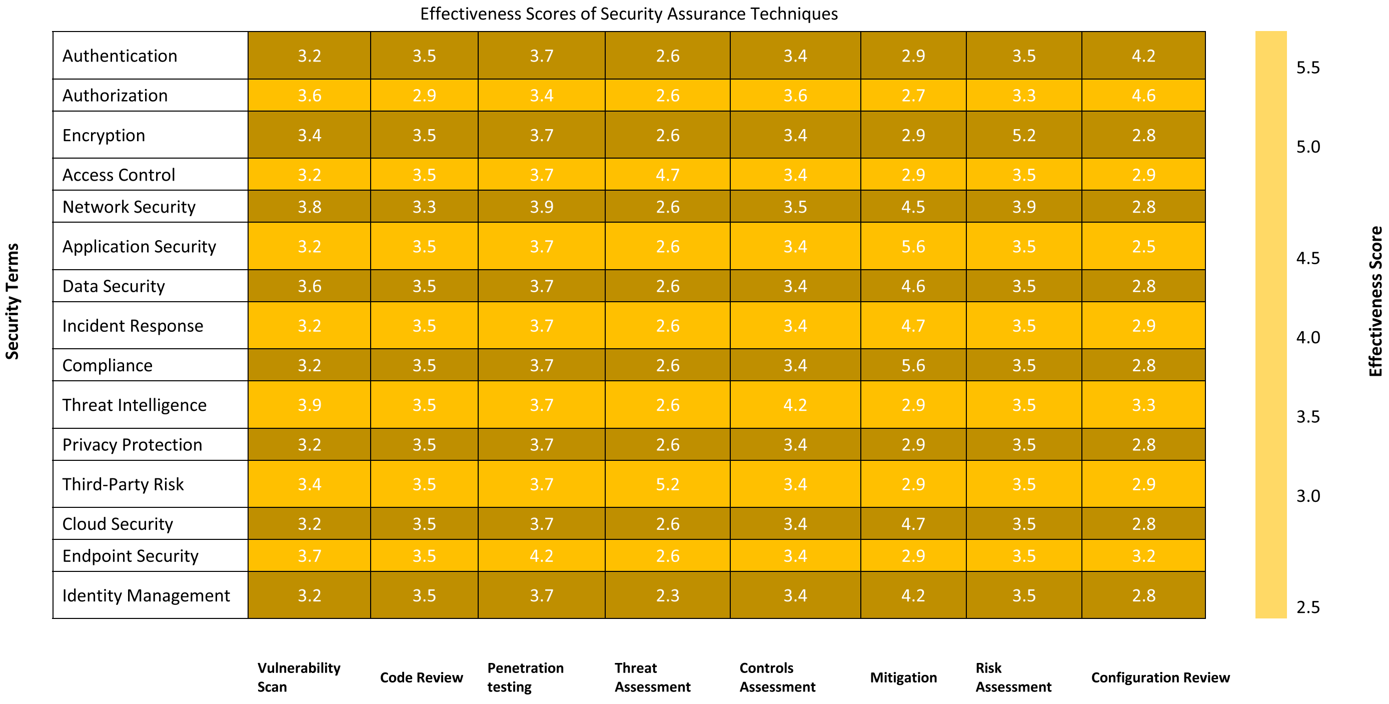
4. Results

The assessment of software security reliability was conducted through the review of 330 papers; the effectiveness of the security assurance techniques we applied is mathematically displayed in the table (Table 5).

The study evaluated 15 critical security terms using a multi-dimensional framework, prioritizing key aspects for software security reliability. The approach has guided by the Common Vulnerability Scoring System (CVSS) and industry-specific best practices. A risk-based approach was used to balance metrics, focusing on the impact and exploitability of vulnerabilities across various security domains. The effectiveness of security assurance techniques was categorized into high, medium, and low levels, reducing data complexity and allowing for comparative analysis across domains like authentication, encryption, and network security. A quantitative scoring system has been used to balance metrics, ensuring that all security domains were equally represented and no critical aspect was overlooked. The assesses the effectiveness of safety warranty strategies across various domains, with scores ranging from 0 to 10. Vulnerability Scan consistently ranks within the excessive effectiveness category, indicating robustness in identifying vulnerabilities in domain names and Authentication and Authorization. Code Review also shows strong performance, falling into the excessive effectiveness class throughout all areas. Authentication's critical role in verifying user identities is underscored by its efficacy score of 5.5, which is particularly evident when it is combined with techniques like risk assessment and penetration testing. Encryption, which receives a score of 5.0, emphasizes the significance of safeguarding data in transit and at rest, rendering it a critical element in the prevention of data intrusions. Application security (4.5) and compliance (4.5) also exhibit high efficacy in preventing vulnerabilities and complying with regulatory standards. The necessity of a robust infrastructure and rapid recovery mechanisms in order to mitigate ongoing assaults is underscored by the scores of 4.5 and 4.0 for network security and incident response, respectively. Cloud security and third-party risk are critical areas where security assurance is impacted by challenges such as configuration management and third-party vendor risks, as evidenced by moderate scores of 3.0 and below. The significance of continuous monitoring and intelligence collection in enhancing an organization's security posture is further emphasized by security controls such as identity management (2.5) and threat intelligence (3.5). In conclusion, these scores provide a comprehensive understanding of the efficacy of various security techniques, thereby assisting security professionals and software developers in customizing their security strategies to address specific risks and vulnerabilities within the software development life cycle (SDLC). These metrics guarantee that security measures are appropriately prioritized in accordance with their importance to the system's overall integrity (Fig. 4).

In this simulation of, the CVSS (Common Vulnerability Scoring System) Base Score of 4.0 is within the medium-effectiveness range (4.0 to 6.9), suggesting that the security technique evaluated is moderately

Table 5
The effectiveness scores of the security assurance techniques.



Effectiveness Scores of Security Assurance Techniques

Security Terms	Vulnerability Scan	Code Review	Penetration testing	Threat Assessment	Controls Assessment	Mitigation	Risk Assessment	Configuration Review
Authentication	3.2	3.5	3.7	2.6	3.4	2.9	3.5	4.2
Authorization	3.6	2.9	3.4	2.6	3.6	2.7	3.3	4.6
Encryption	3.4	3.5	3.7	2.6	3.4	2.9	5.2	2.8
Access Control	3.2	3.5	3.7	4.7	3.4	2.9	3.5	2.9
Network Security	3.8	3.3	3.9	2.6	3.5	4.5	3.9	2.8
Application Security	3.2	3.5	3.7	2.6	3.4	5.6	3.5	2.5
Data Security	3.6	3.5	3.7	2.6	3.4	4.6	3.5	2.8
Incident Response	3.2	3.5	3.7	2.6	3.4	4.7	3.5	2.9
Compliance	3.2	3.5	3.7	2.6	3.4	5.6	3.5	2.8
Threat Intelligence	3.9	3.5	3.7	2.6	4.2	2.9	3.5	3.3
Privacy Protection	3.2	3.5	3.7	2.6	3.4	2.9	3.5	2.8
Third-Party Risk	3.4	3.5	3.7	5.2	3.4	2.9	3.5	2.9
Cloud Security	3.2	3.5	3.7	2.6	3.4	4.7	3.5	2.8
Endpoint Security	3.7	3.5	4.2	2.6	3.4	2.9	3.5	3.2
Identity Management	3.2	3.5	3.7	2.3	3.4	4.2	3.5	2.8

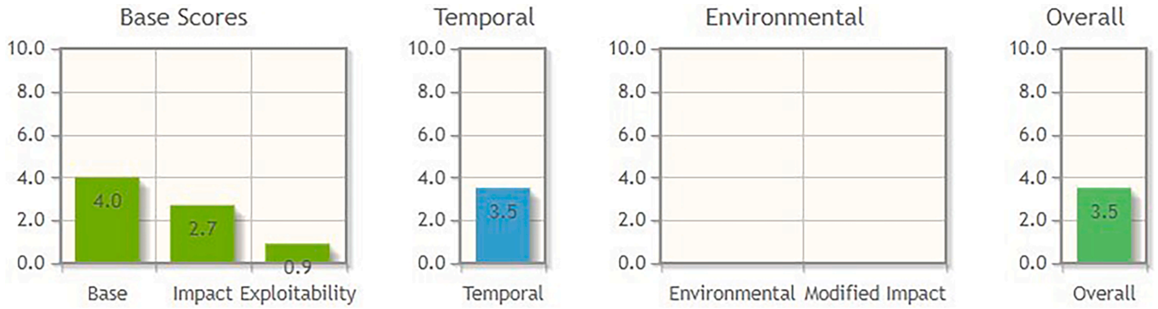


Fig. 4. Simulation of effectiveness scores of security assurance techniques.

effective in mitigating the vulnerability. The technique's strong protection in minimizing the overall damage to confidentiality, integrity, and availability caused by the vulnerability is indicated by the Impact Subscore of 2.7, which falls within the high-effectiveness range (0 to 3.9). Nevertheless, the Exploitability Subscore of 0.9, which is also within the high-effectiveness range, suggests that the vulnerability is relatively difficult to exploit, indicating that the security control in place is effective in limiting potential, exploitation. The CVSS Temporal Score of 3.5 is also within the high-effectiveness range, suggesting that the technique's capacity to safeguard the system is not substantially compromised by factors such as available mitigation and exploit code over time. The security technique in question provides robust protection against the assessed vulnerability under the current conditions, as evidenced by the final CVSS Score of 3.5, which reflects high efficacy. The proposed framework for evaluating software security reliability incorporates the Common Vulnerability Scoring System (CVSS) to provide a structured approach to vulnerability assessment by quantifying and prioritizing security risks. CVSS scores were modified within the framework by mathematical modeling, which weighted specific metrics to correspond with security assurance techniques. Nevertheless, CVSS is constrained in dynamic threat environments by its static scoring model, which is incapable of adapting to real-time fluctuations in threat levels. By integrating supplementary adaptive threat intelligence inputs, CVSS's applicability in evolving contexts was improved, and a more responsive risk assessment framework was established. This limitation was resolved.

5. Evaluation metrics

The efficacy scores for a variety of security assurance techniques that are essential for the maintenance of comprehensive cybersecurity measures are assessed in the table. Each technique is essential for the evaluation, mitigation, and management of risks in particular domains.

The mathematical calculations below illustrate the efficacy of our assessment of the reliability of software security using security assurance techniques:

Vulnerability Scanning: This method entails the systematic examination of systems and networks to identify known vulnerabilities. The efficacy score of the Vulnerability Scan in relation to various criteria is as follows:

Scores: 3.2, 3.6, 3.4, 3.2, 3.8, 3.2, 3.6, 3.2, 3.2, 3.9, 3.2, 3.4, 3.2, 3.7, 3.2

Mean Score ($\overline{X}$Vulnerability Scan) Vulnerability Scan): Calculate the average score:

$$\overline{X}\text{Vulnerability Scan } = \frac{\sum_{\text{i}=1}^{15} X_{i,\text{Vulnerability Scan}}}{15} = \frac{49.3}{15} = 3.287$$

Categorized Effectiveness: Falls within the "Medium" category (4.0-6.9).

Code Review: This technique involves reviewing software source code to identify security vulnerabilities. The effectiveness scores for Code Review are as follows:

Scores: 3.5 (consistently across all criteria).

Mean Score ($\overline{X}$Code Review) : The mean score is 3.5, categorized as "Medium" (4.0-6.9)

Penetration Testing: Simulates real-world attacks to evaluate system defenses. The effectiveness scores for Penetration Testing are:

Scores: 3.7, 3.4, 3.7, 3.7, 3.9, 3.7, 3.7, 3.7, 3.7, 3.7, 3.7, 3.7, 3.7, 4.2, 3.7

Mean Score ($\overline{X}$Penetration Testing):

$$\overline{X}\text{Penetration Testing} = \frac{\sum_{\text{i}=1}^{15} X_{i,\text{Penetration Testing}}}{15} = \frac{55.8}{15} = 3.72$$

Categorized Effectiveness: Falls within the "Medium" category (4.0-6.9).

Threat Assessment: Assesses potential threats and their impact on organizational assets. The effectiveness scores for Threat Assessment are:

Scores: 2.6 (consistently across all criteria).

Mean Score ($\overline{X}$Threat Assessment): The mean score is 2.6, categorized as "High" (0-3.9).

Controls Assessment: Evaluates the effectiveness of implemented security controls. The effectiveness scores for Controls Assessment are:

Scores: 3.4 (consistently across all criteria).

Mean Score ($\overline{X}$Controls Assessment): The mean score is 3.4, categorized as "Medium" (4.0-6.9).

Mitigation: Implements strategies to reduce risks and vulnerabilities. The effectiveness scores for Mitigation are:

Scores: 2.9, 2.7, 2.9, 2.9, 4.5, 5.6, 4.6, 4.7, 5.6, 2.9, 2.9, 2.9, 4.7, 2.9, 4.2

Mean Score ($\overline{X}$Mitigation) :

$$(\overline{X}\text{Mitigation}) = \frac{\sum_{\text{i}=1}^{15} X_{i,\text{Mitigation}}}{15} = \frac{56.1}{15} = 3.74$$

Categorized Effectiveness: Falls within the "Medium" category (4.0-6.9).

Risk Assessment: Systematically evaluates organizational risks. The effectiveness scores for Risk

Assessment are: Scores: 3.5 (consistently across all criteria).

Mean Score ($\overline{X}$Risk Assessment) :The mean score is 3.5, categorized as "Medium" (4.0-6.9).

Configuration Review: Reviews system configurations to ensure security. The effectiveness scores for Configuration Review are:

Scores: 4.2, 4.6, 2.8, 2.9, 2.8, 2.5, 2.8, 2.9, 2.8, 3.3, 2.8, 2.9, 2.8, 3.2, 2.8

Mean Score ($\overline{X}$Configuration Review) :

$$(\overline{X}\text{Configuration Review}) = \frac{\sum_{\text{i}=1}^{15} X_{i,\text{Configuration Review}}}{15} = \frac{42.4}{15} = 2.827$$

Categorized Effectiveness: Falls within the "High" category (0-3.9).

Calculate Weighted Scores for Each Technique:

Vulnerability Scan: Categorized as Medium (3.287):

Weighted Score$_{\text{Vulnerability Scan}} = 60$

Code Review: Categorized as Medium (3.5): Weighted Score$_{\text{Code Review}} = 60$

Penetration Testing: Categorized as Medium (3.72): Weighted Score$_{\text{Penetration Testing}} = 60$

Threat Assessment: Categorized as High (2.6): Weighted Score$_{\text{Threat Assessment}} = 90$

Controls Assessment: Categorized as Medium (3.4):

Weighted Score$_{\text{Controls Assessment}} = 60$

Mitigation: Categorized as Medium (3.74): Weighted Score$_{\text{Mitigation}} = 60$

Risk Assessment: Categorized as Medium (3.5): Weighted Score$_{\text{Risk Assessment}} = 60$

Configuration Review: Categorized as High (2.827):

Weighted Score$_{\text{Configuration Review}} = 60$

Sum Up Weighted Scores:

Total Weighted Score = 60 + 60 + 60 + 90 + 60 + 60 + 60 + 90 = 540

Calculate Overall Successful Score:

Maximum Possible Weighted Score (if all techniques scored in the High category): 8 x 90 = 720

$$\text{Overall Successful Score}: \ \text{Overall Successful Score} = \left(\frac{\text{Total Weighted Score}}{\text{Maximum Possible Weighted Score}}\right) \times 100$$

$$\text{Overall Successful Score} = \left(\frac{540}{720}\right) \times 100 = 75$$

This research provides a comprehensive assessment of cybersecurity effectiveness across critical security assurance techniques. The calculated overall successful score of 75 % indicates the robustness of implemented cybersecurity measures, highlighting strengths and areas for potential enhancement in organizational defense strater against evolving cyber threats.

6. Conclusions

The necessity for advanced, systematic, and effective evaluation is underscored by the escalating vulnerabilities in software security and the increase in attacks. Comprehensive security assessments offer detailed information regarding the behavior of unfamiliar assaults, uncover vulnerabilities, and provide detailed information about all types of robust attacks. The software security analysis framework has effectively elucidated the procedures for mitigating security threats and attacks. Attack trees can be employed to systematically analyze and mitigate complex attack scenarios. The prevention of their recurrence is facilitated by the evaluation of the comprehensive characteristics of attack varieties. Additionally, the overall efficacy of the security system is improved by a well-organized framework at each security level. The attack trees table is a critical component of comprehensive security measures, as it is responsible for the analysis of prospective attack scenarios. Detailed and effective methods for the development of advanced dynamic analysis and penetration testing are provided through systematic analysis to address complex problems. The reliability of the security measures is assessed in order to determine the system's effectiveness. The effectiveness of the analysis has been considerably improved by the evaluation of multidimensional conditions in the analysis of security cases. The reliability of security has been enhanced by the efficacy of security assurance techniques that have been implemented in software security. Resource constraints are a major obstacle in implementing advanced security techniques like dynamic penetration testing or threat modeling. These methods require extensive expertise, specialized tools, and financial resources, which may not be feasible for smaller organizations. To address this, future research should explore cost-effective, lightweight solutions like automated processes or open-source tools, which could alleviate the burden on organizations with limited resources and make security assurance techniques more accessible.

Funding statement

This research has no funding.

Data availability

The data that support the findings of this study are not openly available due to reasons of sensitivity and are available from the corresponding author upon reasonable request.

CRediT authorship contribution statement

Mohammad Ali: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Ahsan Ullah:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Investigation. **Md. Rashedul Islam:** Visualization, Validation, Resources, Investigation, Data curation. **Rifat Hossain:** Visualization, Resources, Investigation, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Ahmad Z, Asif M, Faisalabad U, Shahid PM, Pakistan F, Rauf A. Implementation of secure software design and their impact on application. 2015.

Ahmad Z, Asif M, Faisalabad U, Shahid PM, Pakistan F, Rauf A. Implementation of secure software design and their impact on application. 2015.

Baldassarre, MT, Barletta, VS, Caivano, D, Scalera, M., 2020. Integrating security and privacy in software development. Softw. Qual. J.. 28, 987–1018.

Chandra, P, Deleersnyder, S, De Win, B, Hinojosa, K, 2009. Software assurance maturity model.

Chechik, M, Salay, R, Viger, T, Kokaly, S, Rahimi, M., 2019. Software assurance in an uncertain world. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag, pp. 3–21.

Chris Eng, the state of software security 2024 Addressing the Threat of Security Debt.

Cukic, B, Bastani, FB, 1997. Attaining high confidence in software reliability assessment.

Duan, L, Rayadurgam, S, Heimdahl, MPE, Ayoub, A, Sokolsky, O, Lee, I, 2014. Reasoning about confidence and uncertainty in assurance cases a survey.

Frydman, M, Ruiz, G, Heymann, E, César, E, Miller, BP., 2014. Automating risk analysis of software design models. Sci. World J.. 2014.

Granata, D, Rak, M., 2024. Systematic analysis of automated threat modelling techniques: Comparison of open-source tools. Softw. Qual. J. 32, 125–161.

Huang Y, Kintala C. Software fault tolerance in the application layer. 1993.

Humayun, M, Jhanjhi, NZ, Almufareh, MF, Khalil, MI., 2022. Security threat and vulnerability assessment and measurement in secure software development. Comput. Mater. Contin. 71, 5039–5059.

Khan, RA, Khan, SU., 2018. A preliminary structure of software security assurance model. In: Proceedings - International Conference on Software Engineering. IEEE Computer Society, pp. 137–140.

Khan, RA, Khan, SU, Alzahrani, M, Ilyas, M., 2022. Security assurance model of software development for global software development vendors. IEEe Access. 10, 58458–58487.

Khan, AW, Zaib, S, Khan, F, Tarimer, I, Seo, JT, Shin, J., 2022. Analyzing and evaluating critical cyber security challenges faced by vendor organizations in software development: SLR based approach. IEEE Access. 10, 65044–65054.

Kotaiah, B, Khan, RA., 2012. A survey on software reliability assessment by using different machine learning techniques. Int. J. Sci. Eng. Res. 3.

Kudriavtseva A, Gadyatskaya O. Secure Software development methodologies: a multivocal literature review. 2022.

Kumar, R, Khan, RA., 2024. Securing communication protocols in military computing. Netw. Secur. 2024.

Kumar, R, Mishra, SK., 2024. Assessing the impact of heat vulnerability on urban public spaces using a fuzzy-based unified computational technique. AI. Soc.

Kumar, R, Khan, SA, Khan, RA., 2015. Revisiting Software Security: Durability Perspective. Int. J. Hybrid Inf. Technol. 8, 311–322.

Kumar, R, Baz, A, Alhakami, H, et al., 2020. A Hybrid model of hesitant fuzzy decision-making analysis for estimating usable-security of software. IEEE Access. 8, 72694–72712.

Kumar, R, Khan, SA, Khan, RA., 2023. Software Durability: Concepts and Practices. CRC Press.

Mohammed, NM, Niazi, M, Alshayeb, M, Mahmood, S., 2017. Exploring software security approaches in software development lifecycle: a systematic mapping study. Comput. Stand. Interfaces. 50, 107–115.

Muram, FU, Javed, MA., 2023. ATTEST: Automating the review and update of assurance case arguments. J. Syst. Arch. 134.

Pandey, AK, Das, AK, Kumar, R, Rodrigues, JJPC., 2024. Secure Cyber engineering for IoT-Enabled smart healthcare system. IEEE Internet of Things Magazine 7, 70–77.

Parizi, RM, Qian, K, Shahriar, H, Wu, F, Tao, L., 2018. Benchmark requirements for assessing software security vulnerability testing tools. In: Proceedings - International Computer Software and Applications Conference. IEEE Computer Society, pp. 825–826.

Rhodes T, Boland F, Fong E, Kass M, Galllagher PD. Software assurance using structured assurance case models. 2009.

Rushby J., (650) 326-6200 • Facsimile. 2015.

Sahu, K, Srivastava, RK., 2021. Predicting software bugs of newly and large datasets through a unified neuro-fuzzy approach: Reliability perspective. Adv. Math.: Sci. J. 10, 543–555.

Shukla, A, Katt, B, Nweke, LO, Yeng, PK, Weldehawaryat, GK., 2022. System security assurance: a systematic literature review. Comput. Sci. Rev. 45.

Sklyar V, Kharchenko V. Assurance case for safety and security implementation: a survey of applications. 2020.

Surakhi, O.M., Hudaib, A, AlShraideh, M, Khanafseh, M, 2017. A survey on design methods for secure software development. Int. J. Comput. Technol. 16, 7047–7064.

SYNOPSYS, open source security and risk analysis report 2023 FINAL.

Tariq, U, Ahmed, I, Bashir, AK, Shaukat, K., 2023. A Critical cybersecurity analysis and future research directions for the internet of things: a comprehensive review. Sensors 23.

Tatam, M, Shanmugam, B, Azam, S, Kannoorpatti, K., 2021. A review of threat modelling approaches for APT-style attacks. Heliyon 7.

Von Solms, S, Futcher, LA., 2020. Adaption of a secure software development methodology for secure engineering design. IEEE Access. 8, 125630–125637.

Arguing security-creating security assurance cases Charles B. Weinstock Howard F. Lipson John Goodenough January 2007.

Weir, C, Rashid, A, Noble, J., 2021. Challenging software developers: Dialectic as a foundation for security assurance techniques. J. Cybersecur. 6.

Yuniar Banowosari L, Abidzar Gifari B. XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE System Analysis and Design Using Secure Software Development Life Cycle Based On ISO 31000 and STRIDE 2019.