DASANAYAKE, S.D.L.V., SENANAYAKE, J. and WIJAYANAYAKE, W.M.J.I. 2024. Devsecops for continuous security in trading software application development: a systematic literature review. *Journal of desk research review and analysis* [online], 2(2), pages 215-232. Available from: https://doi.org/10.4038/jdrra.v2i2.52

Devsecops for continuous security in trading software application development: a systematic literature review.

DASANAYAKE, S.D.L.V., SENANAYAKE, J. and WIJAYANAYAKE, W.M.J.I.

2024

DOI: https://doi.org/10.4038/jdrra.v2i2.52

The Journal of Desk Research Review and Analysis, Vol. 2, Issue 2, 2024, 215-231

DEVSECOPS FOR CONTINUOUS SECURITY IN TRADING SOFTWARE APPLICATION DEVELOPMENT: A SYSTEMATIC LITERATURE REVIEW

SDLV Dasanayake[1], J Senanayake[2] and WMJI Wijayanayake[3]

Abstract

This systematic literature review examined the implementation of DevSecOps for continuous security in financial trading software application development. This review identifies key strategies and security frameworks, analyses cybersecurity threats specific to trading applications, explores secure coding practices, and discusses the transition from DevOps to DevSecOps, focusing on security. A comprehensive search was conducted across multiple databases up to July 9, 2024. The study aimed to identify best practices for integrating security into every phase of the software development process, from initial design to deployment and maintenance. This included automated security testing, continuous monitoring, and incident response strategies tailored for financial trading platforms. The review also delved into the challenges faced by developers in the financial sector, such as compliance with stringent regulatory requirements and the need to protect highly sensitive financial data. Furthermore, it evaluated the effectiveness of current security frameworks in mitigating risks associated with trading software, including common vulnerabilities and attack vectors. The study had limitations, including the exclusive consideration of the most recent threats, potentially overlooking relevant historical data. Additionally, the focus on financial trading applications may limit the generalizability of the findings to other domains. Despite these limitations, the results highlighted the critical importance of incorporating DevSecOps concepts into software development processes to enhance the security and resilience of financial trading systems in an increasingly hostile cyber environment. This research underscores the need for continuous adaptation and improvement in security practices to keep up with evolving threats.

Keywords: DevSecOps, frameworks, security, threats, trading

[1] Department of Industrial Management, University of Kelaniya, Sri Lanka.

Email: lashadyavidumini@gmail.com https://orcid.org/0009-0007-3725-6658

[2] Robert Gordon University, UK. & University of Kelaniya, Sri Lanka.

Email: janakas@kln.ac.lk https://orcid.org/0000-0003-2278-8671

[3] Department of Industrial Management, University of Kelaniya, Sri Lanka.

Email: janaka@kln.ac.lk https://orcid.org/0000-0002-9523-5384

Received date: 28.08.2024

Accepted date: 22.10.2024

Introduction

DevOps is defined as the convergence of Development and Operations in the technology industry, where collaboration and trust between traditionally siloed domains are promoted. This approach emphasised streamlining development and deployment processes, with agility, velocity, and automation being the central focus. By fostering greater efficiency, reliability, and collaboration, the overall synergy between development and operations teams was intended to be enhanced (Pakalapati et al., 2023).

Due to security challenges such as manual security testing, inconsistent security policies, developer resistance to integrating security protocols, a lack of secure coding standards, and the neglect of static security testing within DevOps practices (Rafi et al., 2020), the concept of DevSecOps was introduced (Abiona et al., 2024). Security was integrated into the DevOps process through DevSecOps by embedding security practices throughout the development lifecycle. This approach involved identifying project objectives and security needs during the planning phase, threat modelling to understand potential vulnerabilities, and performing software impact analysis to assess risks before modifying. Key security tasks, including vulnerability detection, automated vulnerability repair, and infrastructure scanning, were conducted throughout the DevOps workflow to ensure robust security measures were consistently implemented (Fu et al., 2024).

Initially, trading applications were created to facilitate the exchange of digital assets or services. Trading is defined as the act of purchasing, selling, or exchanging goods, services, or financial instruments among different parties to take advantage of financial market conditions for profit (Kanevche et al., 2021).

In traditional financial trading application development, the focus on agility and speed often led to deprioritising data security and privacy assurance, as they were viewed as time-consuming tasks requiring specialised personnel and technology. Security in software development was traditionally considered a post-development task, leading to vulnerabilities being discovered late in the process, potentially resulting in security breaches, financial losses, and reputational damage (David et al., 2024). Researchers implied that a gap existed between assumed knowledge and the actual practices of secure coding among developers. The diversity and lack of standardisation in secure coding guidelines resulted in challenges such as a lack of guidelines, an excess of guidelines, or conflicting recommendations (Gasiba & Lechner, 2019). Furthermore, integrating DevSecOps principles into the development of financial trading software applications was seen as receiving insufficient focus.

The following research questions will be addressed in this study:

A. Research Question 01
How could security practices be refined to minimise resource and time overhead in customising security measures for trading applications instead of standard financial security methods?

B. Research Question 02
What effective strategies could be implemented to educate developers on secure coding practices, ensuring adherence to robust security protocols in developing trading applications?

C. Research Question 03
How could collaboration between development and security teams be optimised to facilitate a smooth transition from DevOps to DevSecOps methodologies, enhancing security measures in the development lifecycle of trading applications?

Methodology

To systematically explore how DevSecOps implementation ensured continuous security in financial and trading application development, the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) framework (Page et al., 2021) was employed. The inclusion criteria for the literature review consisted of papers focusing on security or frameworks in DevSecOps, DevOps, SDLC, or SSDLC, studies related to security threats in trading, studies related to security in financial applications, and articles discussing secure coding practices. The exclusion criteria comprised papers unrelated to security emphasis, articles not addressing security and non-peer-reviewed articles.

The following keywords were used in the search strategy: "DevSecOps," "DevOps," "security," "financial," "trading," "threats," "cybersecurity," "SDLC," "SSDLC," "resource," "coding," and "framework." The Publish or Perish software was also utilised to retrieve relevant literature from Google Scholar and Scopus. Included papers had at least two or more of these keywords and were published within the last five years, which assisted in paving the way for achieving the objectives of the study.

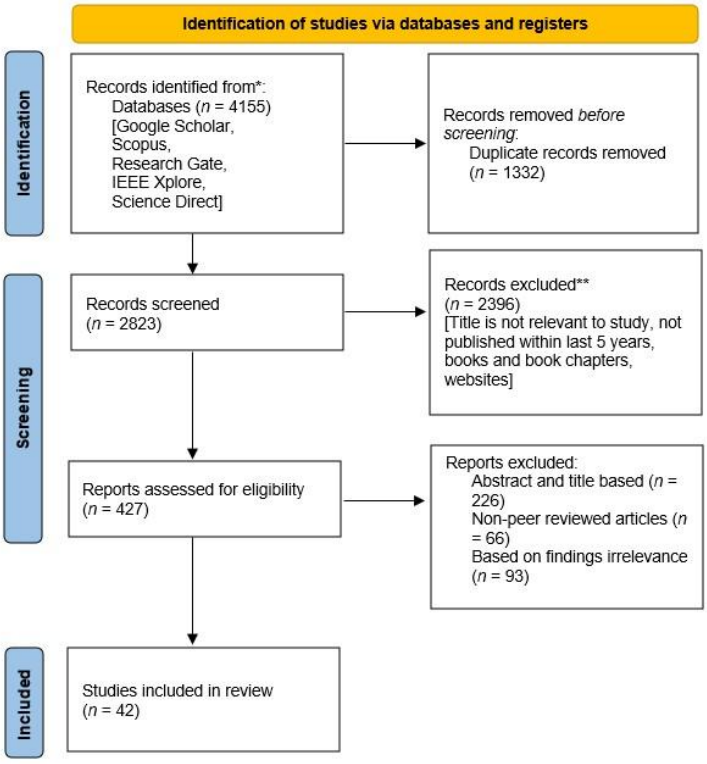

Fig. 1. Research Methodology

Several potential biases were acknowledged while conducting this systematic literature review. Biases could have arisen from choosing studies based on trading and financial areas, potentially excluding necessary research and leading to an incomplete view of the topic. Additionally, using a limited set of keywords or databases might have resulted in missing relevant studies published under different terms or in other sources. Language bias might have resulted from including only studies published in the English language. Finally, a bias may have emerged from considering only studies published within the last five years, potentially overlooking older but still relevant research. Data collection for the study was performed through the systematic literature review, and potential biases were addressed by considering both the limitations and challenges within DevSecOps and DevOps rather than solely focusing on the positive aspects of DevSecOps.

Literature Review

A. Refining Security Practices

Since 2016, financial institutions have been targeted by advanced cyberattacks, leading to a continuous evolution in threats to the global financial system (Shalabi et al., 2023). Concurrently, managing these risks in digital trade became a crucial responsibility for corporations to protect their trading domains (Huang et al., 2021). Therefore, researchers introduced various security refinements for diversified financial domains as solutions to this concern.

1) Financial Application Security:

To analyse financial application security refinements comprehensively, several advanced frameworks and strategies were proposed in recent research, including social recommendation frameworks (Zhao et al., 2024), unified frameworks for automating software security (Aljohani & Alqahtani, 2023), big data encryption algorithms (Xiao & Metawa, 2022), CNN (Convolutional Neural Networks) based IDS (Intrusion Detection System) frameworks (Dahiya et al., 2023), and SWIFT customer security frameworks (Shalabi et al., 2023), along with other regulations or guidelines.

The Multiinterest and Social Interest-Field Framework (MISIF) enhanced financial security through social recommendations (Zhao et al., 2024), while a unified framework affiliated with automating software security analysis within DevSecOps paradigms served as middleware between CI/CD pipelines and application security services. This framework consisted of the components of an agent and an engine. The agent, embedded within a project's CI pipeline, collected and forwarded project details to the engine. The engine, built on a microservice architecture, handled tasks such as security vulnerability scanning and offered modular deployments for features like vulnerability management and reporting (Aljohani & Alqahtani, 2023).

Applying blockchain technology (Fernandez-Morin et al., 2023) and big data encryption algorithms (Xiao & Metawa, 2022) was also recommended to ensure information security in the financial sector. Incorporating blockchain technology into cybersecurity frameworks offered key benefits like decentralisation, peer-to-peer connections, intelligent contracts, and enhanced security measures, thereby improving privacy protection (Dahiya et al., 2023). In finance and financial services, blockchain technology enhanced security through robust measures, decentralisation, and immutability, ensuring data integrity and protecting against cyber threats. Techniques like Proof of Work (PoW), Proof of Stake (PoS), encryption, and smart contracts further secured transactions and automated agreement enforcement while ensuring regulatory compliance to mitigate risks (Trivedi et al., 2021). Blockchain emphasises consensus mechanisms, which are vital for maintaining the integrity and security of information in financial markets. Additionally, the encryption and linking of information blocks made the data immutable and challenging to alter, further enhancing security. The distributed ledger system inherent in blockchain technology ensured data integrity by making it extremely challenging to modify data, as each block was securely linked to the previous one (Fernandez-Morin et al., 2023).

Other big data algorithms were examined in terms of data encryption. These algorithms included classifying data based on security levels, applying appropriate encryption methods, securing data processing to prevent unauthorised access, and implementing data management and protection strategies such as power separation systems and emergency backups. Furthermore, privacy considerations necessitated defining the nature of data for effective utilisation and maintenance, classifying data into different security levels, and establishing a data destruction strategy for unused data to prevent loss (Xiao & Metawa, 2022).

Applying knowledge graphs in financial information security strategies highlighted several key refinements. It enabled the analysis and modelling of financial business risks, facilitating the systematic sorting of financial risk knowledge and the extraction of secure entities, semantic relationships, and attribute information from diverse data sources. Entities related to network threats were classified into risk and strategy information, with enhanced word vector sequences aiding in effectively identifying financial security entities. Integrating the Unified Cybersecurity Ontology (UCO) and D3FEND network security countermeasures improved threat intelligence sharing and the understanding of offensive and defensive technologies. This approach established a comprehensive financial information security strategy system encompassing threat modelling, risk analysis, and attack reasoning (Ye et al., 2023). The SWIFT Customer Security Framework enhanced resilience against cyber threats by assembling specialist teams, evaluating existing controls, assessing system performance, and conducting in-depth studies of new security controls. This framework ensured compliance with international standards and industry best practices (Shalabi et al., 2023).

The Application Security Verification Standard (ASVS) for Regulation Compliance mapping addressed regulatory compliance compliance. This included aligning ASVS security controls with the Monetary Authority of Singapore (MAS) regulations, identifying crucial controls, and leveraging international standards like the NIST Cybersecurity Framework (Tan et al., 2021). Furthermore, a practical illustration in the Indonesian banking sector demonstrated refining financial application security. Regulatory authorities enhanced financial security by increasing risk awareness, optimising supervisory procedures, and improving coordination mechanisms. Clear and specific regulatory mandates, a robust enforcement framework, and alignment with international best practices ensured compliance and effectively addressed emerging financial challenges (Nasution, 2023).

2) Trading Application Security:

Refining security measures for trading applications involved integrating multiple frameworks (Huang et al., 2021; Liu et al., 2022; Cali et al., 2024; Pahlevan & Ionita, 2022) and practices (Oosthoek & Doerr, 2020) to address the complexity of cybersecurity risks in various trading sectors.

A systematic framework defined for understanding the transnational governance of cybersecurity risks in digital trade involved several key steps. Data was collected from the Technical Barriers to Trade Information Management System (TBT IMS) and the ECIPE Digital Trade Estimates (DTE) database, focusing on keywords like "cyber security" and "information security." Cases were identified and categorised, with additional cases found through reviews and expert workshops. Each case was analysed for related events and strategies by governments or corporations, resulting in 75 cases with 228 events involving 31 nations. Finally, cases were annotated using a detailed framework (Huang et al., 2021), with a collaborative review ensuring unbiased results. This approach allowed for the comprehensive assessment of threats, enabling informed decision-making for effective security management. Through meticulous documentation and multi-perspective analysis, this method enhanced the accuracy and reliability of threat identification and response strategies (Huang et al., 2021).

In Bitcoin exchange trading platforms, proper configuration of HTTP security headers, timely patching of vulnerabilities, minimising attack surfaces, improving server security, and complying with Know Your Customer (KYC) and Anti-Money Laundering (AML) regulations were critical practices to mitigate adversary exploitation and laundering techniques (Oosthoek & Doerr, 2020). In e-commerce, applying theoretical frameworks like Cyber-attack Theory (CAT) and Information Security Theory (IST), along with steps such as employee training, establishing organisational security protocols, and investing in secure technology, helped manage cybersecurity challenges effectively (Liu et al., 2022).

Blockchain-enabled token-based renewable energy certificates (RECs) were introduced for trading platforms where blockchain integration ensured a decentralised and immutable ledger for transparent and tamper-proof record-keeping, preventing fraud and double-counting. Tokenization converted RECs into digital tokens, enhancing liquidity and accessibility for easier trading while advanced security measures protected against blockchain-related threats using cryptographic standards and secure consensus mechanisms. The framework was designed for scalability, accommodating a growing number of participants and transactions without compromising performance. Smart contracts automate the trading process, reducing intermediaries and ensuring reliable execution based on predefined conditions (Cali et al., 2024).

Secure and Efficient Exchange of Threat Information was performed with the SETS framework, which enabled the secure and efficient exchange of Cyber Threat Intelligence (CTI) among organisations using blockchain technology. Organisations with valid credentials connected to the framework, where permissions were role-based. Threat information was stored in a database, while its hash was stored on a private blockchain to ensure integrity and prevent tampering. The framework automated CTI distribution through a publish-subscribe mechanism. When an organisation identifies a cyber incident, it generates a hash and publishes the threat information to the framework. The hash was stored on the blockchain, and the framework verified the CTI's integrity, rejecting duplicated or altered submissions. The CTI was immediately forwarded to subscribed organisations, which verified it against the blockchain hash. Organisations could also request specific CTI, which the framework retrieved, verified, and sent (Pahlevan & Ionita, 2022).

The literature on refining security frameworks, practices, and regulations in the financial and trading sectors suggested notable differences in addressing security threats. Financial application security focused on comprehensive frameworks, advanced encryption, neural networks for intrusion detection, and knowledge graphs for risk modelling.

Although existing frameworks contributed valuable insights into cybersecurity, several limitations prevented them from fully applicable to financial trading applications. For instance, Multiinterest and Social Interest-Field Frameworks (MISIF) were limited by their focus on social recommendations and static user interests (Zhao et al., 2024), making them less suitable for trading contexts that demanded real-time adaptability to market conditions—additionally, blockchain-based and CNN-based IDS frameworks presented implementation challenges. Issues include scalability, resource consumption, and the complexity of smart contracts. These frameworks were also primarily designed to address general cybersecurity concerns (Dahiya et al., 2023) rather than the rapid changes and unique risks associated with trading systems.

The Secure and Efficient Threat Sharing (SETS) framework faced performance trade-offs, as blockchain-based platforms often sacrificed efficiency and speed to ensure trust and data privacy. Furthermore, the framework's reliance on authorised participants and the need to address GDPR compliance complicated its seamless integration (Pahlevan & Ionita, 2022). The slow pace of its implementation suggested that the framework might not have met the performance demands of dynamic trading platforms. Cybersecurity frameworks designed for e-commerce also faced limitations when applied to trading systems. These frameworks often focused on common threats like social engineering, malware, and denial-of-service attacks while neglecting threats specific to trading environments. The lack of quantitative analysis in these frameworks also limited their ability to provide comprehensive insights into the full scope of cybersecurity threats (Liu et al., 2022). Moreover, Renewable Energy Certificate (REC) systems using distributed ledger technology emphasised the need for organisational-

level policies and a balance between decentralisation and performance, challenges that were misaligned with the high-speed, resource-efficient demands (Cali et al., 2024) of financial trading applications.

Overall, trading application security emphasises systematic frameworks for governance, theoretical models for cybersecurity, and secure threat information exchange. Financial applications use blockchain for decentralisation and big data encryption, while trading platforms use blockchain for transparent record-keeping and secure intelligence exchange. These differences highlighted how financial applications used a broader range of advanced frameworks and strategies while trading applications focused on practical implementations of theoretical models tailored to their specific needs.

B. Identification of Threats

In recent years, the financial services sector saw the rise of new technologies like digital banking, blockchain, and data analytics, allowing banks, insurers, and other financial institutions to serve their clients innovatively. Consequently, data breaches have become increasingly common due to this rapid digitalisation (Tan et al., 2021). While utilising technology for financial activities became standard practice, it was crucial to minimise exposure to threats to ensure that traders could conduct transactions securely (Kariuki et al., 2023).

1) Threats to Trading Applications:

Threats documented in the literature over the past five years were analysed to understand the differences and variations in threats between the financial and trading sectors. This analysis highlighted the unique challenges faced by trading systems, which differed from those in the broader financial services industry.

Table I
Threats In Trading

Threats	**Paper References**
Supply Chain Attacks	(Huang et al., 2021; Hogan et al., 2023; Oosthoek & Doerr, 2020)
Intellectual Property and Data Theft, Unauthorised Access Due to Weak Practices, Spam Emails	(Huang et al., 2021)
Vishing, personalisation, Smishing	(Liu et al., 2022)
Malicious Domains, COVID-19 Dis-information Used as a Weapon, Fake News and Disinformation Lures, Unprotected Networks	(Kariuki et al., 2023)
Phishing Attacks	(Hogan et al., 2023; Kayode-Ajala, 2023; Liu et al., 2022)
Ransomware Attacks, Data Breaches	(Hogan et al., 2023; Kayode-Ajala, 2023)

Unauthorised Malicious Breaches	(Hogan et al., 2023; Kariuki et al., 2023)
Use of Stolen Credentials, Abuse of Functionality, Advanced Techniques, Hot Wallet Breaches, Cold Storage Breaches, Cryptsy Breach Gate.io Breach, and Bitstamp Breach	(Oosthoek & Doerr, 2020)
Hacking	(Bianchi & Tosun, 2019; Kariuki et al., 2023)
Vulnerable Networks, Exploitation of Vulnerabilities, Failure to Install Security Software	(Kariuki et al., 2023; Gagliani, 2020)
Manipulation of Transactions	(Kayode-Ajala, 2023; Kariuki et al., 2023)
Distributed Denial of Service (DDoS)	(Kayode-Ajala, 2023)
Spyware and Trojans	(Liu et al., 2022; Kariuki et al., 2023)
Industrial Security Systems (ICS) Attacks	(Gagliani, 2020)
Systemic Disruptions Aiming for Widespread Chaos	(Kayode-Ajala, 2023)

A study on Bitcoin exchanges analysed 36 cyber breaches, cumulatively resulting in the theft of at least 1,156,399 BTC from legitimate owners. Although the amount of BTC stolen per breach has decreased in recent years, the increasing BTC-USD exchange rate has elevated the financial impact, with the USD yield now significantly higher. These breaches have collectively reduced Bitcoin's market value by billions, affecting all Bitcoin owners. At the same time, the reporting of technical details remains inadequate, as seen in the still unclear specifics of the Mt. Gox breach (Oosthoek & Doerr, 2020). Furthermore, the 2016 Bangladesh Bank cyber heist, which exploited vulnerabilities in the SWIFT network to steal $81 million, illustrates the critical risks posed by supply chain weaknesses in digital trade. These incidents highlight the need for robust security measures and governance strategies to address the diverse cyber threats in financial systems and digital trade platforms (Huang et al., 2021).

After analysing the data above, it became evident that specific threats, such as the Cryptsy Breach, Gate.io Breach, and Bitstamp Breach, were unique to trading systems. These breaches highlighted vulnerabilities in the trading environment not typically seen in the broader financial services industry. In traditional financial services, the primary target of cyber threats was often personal information rather than the direct theft of funds. This distinction underscored the unique security challenges faced by

trading systems compared to the broader financial sector, where breaches involving the actual theft of funds were relatively rare (Oosthoek & Doerr, 2020).

C. Security Training for Developers

To address the identified issues of insufficient attention to secure coding practices among students and new developers (Kotey, 2023), the need to define a new security measure called Individual Security Threshold (IST) for linear secret sharing schemes (Kurihara et al., 2024), and the lack of a comprehensive and systematic secure coding training program for software developers in a case organisation within the financial sector (Niinivirta, 2023), it was essential to examine secure coding guidelines introduced in recent literature. These guidelines significantly enhanced software security education and reduced the need for extensive training resources, cost, and effort for new industry entrants.

1) Secure Coding Strategies:

- Enforcing secure programming guidelines and conventions in introductory programming courses to build students' skills in secure coding (Kotey, 2023).
- Providing security education at the academic level to equip new developers with the knowledge and skills needed to develop secure systems (Kotey, 2023; Niinivirta, 2023).
- Exposing students to static analysis tools and automated code review techniques to help detect vulnerable code and develop analytical skills (Kotey, 2023).
- Prioritising software security at fundamental stages of programming education to instil the importance of security in students and new developers (Kotey, 2023).
- Creating patterns and rules from typical coding mistakes that can be easily applied to machine learning to aid students in understanding and addressing vulnerabilities (Kotey, 2023).
- Following good programming conventions to reduce the likelihood of making mistakes and vulnerabilities in software systems (Kotey, 2023).
- Contributing to software security and training by emphasising the need for software security education at academic levels and preparing students for industry standards (Kotey, 2023).
- Establishing security guidelines in organisations to improve secure coding practices (Niinivirta, 2023).
- Addressing non-compliance with secure coding guidelines (Niinivirta, 2023).
- Analysing the influence of programming languages and expertise level on compliance with secure coding guidelines (Niinivirta, 2023).
- Adopting standards such as OWASP (Singleton et al., 2020), CERT, and NIST in relevant domains (Pikulin et al., 2023).
- Using gamification and open-source project contributions for training (Pikulin et al., 2023), (Pruemmer et al., 2023)
- Utilising platforms like Secure Code Warrior for interactive and gamified secure coding training (Pikulin et al., 2023).
- Employ Secure Coding Practices in Networks
  - Securely transmit a message via a noiseless network when certain edges or nodes are eavesdropped ’Secure Network Coding (NC) Protocol’ (Hayashi, 2021).
  - Combine coding operations on nodes for secure message transmission over noisy channels ’Secure Physical Layer Network Coding (PLNC)’ (Hayashi, 2021).
  - Consider a cross-layer protocol because it integrates secure NC and error correction 'PLNC' (Hayashi, 2021).

- Combine signals at the physical layer to enhance security and efficiency in the up-link phase to prevent easy interception and decoding by unauthorised parties (Liu et al., 2020).
- SNs (Sensor Nodes) can transmit data simultaneously, making it difficult for eavesdroppers to isolate and decode individual signals (Liu et al., 2020).
- Utilise SNs to preprocess the data to create equivalent parallelised subchannels for the home router (HR) in order to enhance security by ensuring that the data is not in its original form (Liu et al., 2020)
- Equivalent parallelised subchannels (Liu et al., 2020)
- Ensure that the data transmitted from the HR to the SNs is protected (Liu et al., 2020)

• The lossy secure and private source coding region is characterised by one private key available (Günlü et al., 2022).

• Characterisation of the rate region for lossless secure and private source coding problems (Günlü et al., 2022).

• Consideration of a Gaussian remote source and independent additive Gaussian noise measurement channels for establishing lossy rate region under squared error distortion (Günlü et al., 2022).

• Providing an achievable lossy secure and private source coding region for a binary remote source with additive Gaussian noise channels, including computable differential entropy terms (Günlü et al., 2022).

• Maintain Effective Code Review Practices

- Increase awareness of security issue management in code reviews (Charoenwet et al., 2024).
- Investigate coding weaknesses raised during code reviews (Charoenwet et al., 2024).
- Understand the alignment of raised security concerns and known vulnerabilities (Charoenwet et al., 2024).
- Develop secure code review policies (Charoenwet et al., 2024).
- Enhance awareness of less frequently identified coding weaknesses (Charoenwet et al., 2024).
- Respond effectively to raised security concerns (Charoenwet et al., 2024).
- Address coding weaknesses early in development (Charoenwet et al., 2024).
- Focus on coding weaknesses that lead to security issues (Charoenwet et al., 2024).

• Address insecure usage of APIs to prevent vulnerabilities (Zhang, 2023).

• Simplifying API documentation and improving cybersecurity training for developers (Zhang, 2023).

• SEED Workshop (Cryptography module with OpenSSL's crypto APIs) (Singleton et al., 2020).

• The Secure Web Development Teaching (SWEET) Workshop (Web security modules with GPG tool for encryption) (Singleton et al., 2020).

• Security Injections (Secure coding techniques in C++, Java, Python) (Singleton et al., 2020).

• JCA (Java Cryptography Architecture) exercises (Singleton et al., 2020).

• Input Validation (Ryan et al., 2023; Xing et al., 2024)

• Output Encoding (Ryan et al., 2023)

• Authentication and Password Management (Ryan et al., 2023)

• Session Management (Ryan et al., 2023)

• Access Control (Ryan et al., 2023)

• Cryptographic Practices (Ryan et al., 2023; Xing et al., 2024)

• Error Handling and Logging (Ryan et al., 2023)

• Data Protection (Ryan et al., 2023)

- Communication Security (Ryan et al., 2023)
- System Configuration (Ryan et al., 2023)
- Database Security (Ryan et al., 2023)
- Code Analysis (Ryan et al., 2023)
- Conduct cybersecurity training evaluations, effectiveness evaluation (Pruemmer et al., 2023)
- Integer Overflow (Xing et al., 2024)
- Variable Naming (Xing et al., 2024)
- Scope of Variables (Xing et al., 2024)
- Buffer Overflow/Index Out-of-Bounds (Xing et al., 2024)
- Encapsulation and Data Hiding (Xing et al., 2024)

The effectiveness of software was measured in several ways. Security provided critical guidance, among other factors, which implied that software could improve its effectiveness by choosing appropriate conventional correctness and security criteria (Saxena & Agarwal, 2019).

D. Transition from DevOps to DevSecOps

The transition from DevOps to DevSecOps involved integrating security measures at every software development lifecycle phase to maintain agility and speed. This change necessitated effective collaboration among developers, information security professionals, and operations teams. To address the challenges posed by traditional security testing methods, which were often incompatible with DevOps, as well as developers' typically low prioritisation of security and compliance issues within the DevOps culture, security tools were embedded at various points throughout the DevOps lifecycle. DevSecOps created a unified framework that systematically incorporated security practices across the entire software development process (Nisha & Khandebharad, 2022).

Before examining DevOps and DevSecOps, it was essential first to explore the concepts of the Software Development Life Cycle (SDLC) and the Secure Software Development Life Cycle (SDLC), related frameworks, and their dissimilarities.

1) Software Development and Secure Software Development:

SDLC stands for System Development Life Cycle, a structured framework for planning, designing, implementing, testing, deploying, and maintaining software systems. The stages included planning, analysis, design, implementation, testing, deployment, and maintenance (Chan et al., 2024). SSDLC was an approach that integrated security in every phase of the software development lifecycle to ensure the software remained secure and free from exploitable vulnerabilities (Umeugo, 2023). SSDLC guidelines ensured secure authentication and authorisation, validating and sanitising user input, and protecting sensitive data through encryption and secure storage. Adherence to secure coding standards, rigorous session management, and secure error handling were also crucial. Maintaining secure configuration management, regularly updating and patching software, and conducting thorough security testing were emphasised (Otieno et al., 2023). It emphasised detecting and preventing security defects and outlined responsive measures to potential exploits. By integrating security principles from the initial design phase through to release and beyond, SSDLC aimed to mitigate vulnerabilities and guard against attacks. Influential factors for adopting SSDLC included company domain, budget constraints, timeline, and developer awareness (Omar et al., 2022). The primary distinction between SDLC and SSDLC was

that SSDLC was tailored for cloud-based systems, focusing on security. At the same time, SDLC was a broader framework for all types of software development (Chan et al., 2024).

The Comprehensive Lightweight Application Security Process (CLASP) and the Security Assurance Maturity Model (SAMM) were key frameworks in the Secure Software Development Lifecycle (SSDLC). CLASP provided methods, practices, roles, and resources to integrate repeatable and measurable security into software development. SAMM offered a self-assessment model to guide organisations in applying, evaluating, and improving software security practices throughout the Software Development Lifecycle (SDLC) (Umeugo, 2023). The challenges associated with adopting SSDLC practices included the absence of clear guidelines, stringent project timelines, inadequate knowledge of secure development practices, and ambiguous security requirements. Additionally, top management often lacked a definitive vision and clear policies concerning security integration in system development processes. In contrast, developers often lacked familiarity with security attacks and vulnerabilities and did not possess sufficient knowledge of secure software development methodologies (Maher et al., 2020).

DevSecOps concepts were utilised to address the challenges mentioned above. DevSecOps emphasises using automated security testing tools and processes, reducing the need for time-consuming manual intervention by security experts. Furthermore, it was utilised to achieve compliance with industry regulations and standards, such as HIPAA, PCI DSS, or ISO 27001 (Casola et al., 2024).

2) DevOps and DevSecOps:

DevOps was defined as a five-step workflow encompassing planning, development, code commit, build, test, deployment, and operation and monitoring. It aimed to enhance collaboration, automation, and agility in software development. However, it often neglected security considerations until later stages (Fu et al., 2024). It combines cultural philosophies, practices, and tools to deliver high-velocity applications and services (Rafi et al., 2020). DevSecOps extended these practices by integrating security into the DevOps process, addressing security needs from the planning phase through threat modelling and software impact analysis. Security tasks like vulnerability detection and automated repairs were embedded throughout the workflow to maintain security (Fu et al., 2024). DevSecOps cultivated a security-centric culture, involving developers in security processes. It was presented as a methodology that ensured security was embedded from the start, avoiding costly rework. It promoted collaboration among developers, security professionals, operations teams, and other stakeholders to create secure software with fewer vulnerabilities (Ashenden & Ollis, 2020).

DevOps encountered challenges, including rapid development and deployment, which often neglected security issues. This necessitated specialised tools to detect vulnerabilities in large and complex systems. Interaction among team members, especially with third-party dependencies and complicated security practices. Issues such as outdated security tool technology, functional limitations, integration difficulties, insufficient automation, incomplete data flow coverage, false positives, and disconnected tools were prevalent (Rajapakse et al., 2021). Additionally, challenges such as the lack of automated testing tools, manual security testing, coordination issues, threat modelling scalability, inconsistent security policies, untrusted inputs, compliance requirements, developer resistance, and immature deployment tools were also categorised under the challenges of DevOps (Rafi et al., 2020). Moreover, DevOps practices faced difficulties in ensuring pipeline security, managing the complexity of cloud or microservices environments, determining the timing of security team involvement, insider threats, and finding suitable security activities and tools (Leppänen et al., 2022).

Challenges in adopting DevSecOps included identifying pain points, resource assessment, progress measurement, and scope definition (Pakalapati et al., 2023). Further obstacles encompassed cultural shifts, tool integration, compliance alignment, and addressing skills gaps (Grigorieva et al., 2024; Sharma, 2024).

Furthermore, while the integration of DevSecOps with the Secure Software Development Lifecycle (SSDLC) provided significant security benefits, several challenges were encountered during their combined application. A primary issue was the trade-off between speed and security. DevSecOps emphasised rapid deployment, but many traditional security practices, such as compliance testing and architectural risk analysis, were manual and time-consuming, which slowed down the continuous integration/continuous deployment (CI/CD) process. This tension between maintaining the speed of software delivery and ensuring thorough security checks hindered the seamless integration of SSDLC into DevSecOps environments (Rajapakse et al., 2021). Additionally, the complexity of tool integration created challenges, as the security and development tools were not always well-aligned, resulting in inefficiencies and potential security gaps (Rajapakse et al., 2021). These challenges underscored the need for a balanced approach to merge these methodologies effectively.

The distinction between DevOps and DevSecOps was identified in their focus. DevOps emphasised collaboration, automation, and agility but tended to overlook security until the later stages. In contrast, DevSecOps integrated security practices from the outset, automating security gates to prevent slowing down the workflow. This approach ensured the development of secure software products without compromising agility (Fu et al., 2024).

Discussion

Financial infrastructures were critically vulnerable to cybercriminal activities, necessitating strong security measures. Financial institutions employed a multi-layered security approach, incorporating advanced technological solutions, employee training, and public awareness campaigns. Although comprehensive efforts were made, the continually evolving nature of cyber threats ensured that cybersecurity remained a moving target (Kayode-Ajala, 2023). Furthermore, the cybersecurity risks in digital trade posed an increasingly critical challenge for governments and corporations, who had to work hard to secure their cyber territories (Huang et al., 2021). Due to the complex challenges in information security, researchers recommended using a hybrid approach. This approach would integrate elements from multiple frameworks and standards to ensure comprehensive security (Abohatem et al., 2023).

Additionally, effective decision-making about the cost-effectiveness of these services was deemed crucial for managing applications (Aljohani & Alqahtani, 2023). The financial impact of security breaches could be substantial (Umeugo, 2023), and the costs associated with addressing vulnerabilities and implementing security measures could also be significant (Casola et al., 2024). For instance, the 2017 Equifax data breach led to punitive settlements, a drop in stock price, and significant damage to the company's reputation. Similarly, the SolarWinds hack, reported in December 2020, led to high remediation costs (Umeugo, 2023). DevSecOps integrated security into every phase of the software development lifecycle. This allowed vulnerabilities and security issues to be identified early in the process, reducing the cost and effort needed for remediation. This proactive approach ensured that security was embedded into the software from the start, preventing costly and time-consuming rework later (Abos, 2024).

Additionally, secure coding training was regarded as essential. It promoted software security education, which reduced the need for extensive training resources, costs, and effort for new developers in the

industry (Kotey, 2023). The Secure Software Development Lifecycle (SSDLC) was notably effective. It was estimated that the cost of fixing security bugs at the production stage of the SDLC could be up to 30 times higher than addressing the same bugs at the requirements stage (Umeugo, 2023).

Based on the security strategies reviewed in the literature, a hybrid method that combines secure coding strategies with DevSecOps within a framework designed explicitly for trading was considered a robust solution to evolving cybersecurity threats. This approach significantly reduced the heavy resource consumption associated with costs and security assurance teams. By leveraging DevSecOps and SSDLC methodologies, security costs and threat exposure were minimised through continuous security actions throughout the software development process. Additionally, utilising a customised framework tailored for trading software applications allowed for detecting threats unique to trading while aligning with common financial application threats. This comprehensive approach effectively addressed security concerns within the trading context, further reducing excessive resource usage.

Conclusion And Future Recommendations

Based on the literature review, it was evident that existing frameworks, although robust, did not fully address the unique security requirements and challenges faced by financial trading applications. Organisations organisations were required to adhere to software development frameworks such as the Software Development Lifecycle (SDLC) to develop software systems, including trading systems. SDLC is a structured framework organisations employ to systematically plan, design, implement, test, deploy, and maintain software systems. This process was generally divided into planning, analysis, design, implementation, testing, deployment, and maintenance (Chan et al., 2024).

In the financial sector, despite the advancements brought by Industry 4.0 technologies, a significant deficiency remained in sufficient regulations and effective countermeasures to ensure system security and protect data integrity from potential attacks (Dahiya et al., 2023). The finance sector is among the most rigorously regulated industries globally due to its responsibility for handling sensitive customer information and financial assets. As a result, financial institutions were required to implement stringent security measures to safeguard their systems, applications, and data from cyberattacks. Traditionally, security considerations in financial applications were addressed post-development (David et al., 2024).

The Secure Software Development Lifecycle (SSDLC) is a methodology that integrates security considerations into each phase of the software development process. The primary objective of SSDLC was to ensure that the software remained secure and free from exploitable vulnerabilities throughout its entire lifecycle. Since DevSecOps cultivated a security-centric culture by involving developers in security processes (Ashenden & Ollis, 2020), DevSecOps and SSDLC could be utilised collaboratively to secure trading applications. Implementing a DevSecOps framework tailored explicitly to trading applications, based on the nature of these applications' threats, significantly reduced the heavy resource consumption associated with traditional security methods.

Furthermore, the recommendation of guidelines for developers on secure coding strategies enhanced the collaboration and effectiveness of the security assurance process.

In addition to recommending a hybrid approach that integrates secure coding strategies with DevSecOps, it is critical to outline detailed implementation strategies to enhance security in financial trading applications. First, organisations should implement continuous security assessments at every phase of the DevSecOps pipeline, including automated vulnerability scanning and penetration testing. This can be achieved by integrating security tools such as Static Application Security Testing (SAST)

and Dynamic Application Security Testing (DAST) into the continuous integration/continuous deployment (CI/CD) pipeline. Additionally, establishing clear communication and feedback loops between development and security teams will facilitate real-time responses to identified vulnerabilities. Another practical strategy involves incorporating Infrastructure as Code (IaC) practices, enabling automated infrastructure management that enhances both security and scalability. These steps, combined with regular security training for developers on the latest threat landscapes, ensure that the proposed framework effectively addresses the unique risks of financial trading platforms.

References

Pakalapati, N., Jeyaraman, J., & Sistla, S. M. K. (2023). Building resilient systems: Leveraging AI/ML within DevSecOps frameworks. *Journal of Knowledge Learning and Science Technology*, *2*(2), 213-230.2023.

Rafi, S., Yu, W., Akbar, M. A., Alsanad, A., & Gumaei, A. (2020). Prioritisation-based taxonomy of DevOps security challenges using PROMETHEE. IEEE Access, 8, 105426-105446.

Abiona, O. O., Oladapo, O. J., Modupe, O. T., Oyeniran, O. C., Adewusi, A. O., & Komolafe, A. M. (2024). The emergence and importance of DevSecOps: Integrating and reviewing security practices within the DevOps pipeline. *World Journal of Advanced Engineering Technology and Sciences*, 11(2), 127–133.

Fu, M., Pasuksmit, J., & Tantithamthavorn, C. (2024). AI for DevSecOps: A landscape and future opportunities. arXiv. https://arxiv.org/abs/2404.04839.

Kanevche, J., Karamachoski, J., Puncheva, M., & Marina, N. (2021). Trading application based on blockchain technology. Tehnički Glasnik, 15(2), 282-286.

David, P., Kushwaha, M. K., & Suseela, G. (2024). DevSecOps in finance: Strengthening the security model of applications. In *2024 4th International Conference on Data Engineering and Communication Systems (ICDECS)* (pp. 1–6). IEEE.

Gasiba, T. E., & Lechner, U. (2019). Raising secure coding awareness for software developers in the industry. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)* (pp. 141-143). IEEE.

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., et al. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ*, *372*. https://doi.org/10.1136/bmj.n71

Shalabi, K., Al-Fayoumi, M., & Al-Haija, Q. A. (2023). Enhancing financial system resilience against cyber threats via SWIFT customer security framework. In *2023 International Conference on Information Technology (ICIT)* (pp. 260–265). IEEE.

Huang, K., Madnick, S., Choucri, N., & Zhang, F. (2021). A systematic framework to understand transnational governance for cybersecurity risks from digital trade. *Global Policy*, *12*(5), 625–638.

Zhao, Q., Huang, J., Liu, G., Miao, Y., & Wang, P. (2024). A multi-interest and social interest field framework for financial security. *IEEE Transactions on Computational Social Systems, 11*(2), 1685-1695.

Aljohani, M. A., & Alqahtani, S. S. (2023). A unified framework for automating software security analysis in DevSecOps. In *2023 International Conference on Smart Computing and Application (ICSCA)* (pp. 1-6). IEEE.

Xiao, Y., & Metawa, S. (2022). Application of big data encryption algorithm in financial data security protection. In *The International Conference on Cyber Security Intelligence and Analytics* (pp. 866-870). Springer.

Dahiya, M., Mishra, N., Nagar, C., & Bhati, R. (2023). CNN-based IDS framework for financial cybersecurity. In *2023 4th International Conference on Intelligent Engineering and Management (ICIEM)* (pp. 1-6). IEEE.

Fernandez-Morin, J., Torrejon-Mundaca, K., & Meneses-Claudio, B. (2023). Application of blockchain technology for information security in the financial sector. *Salud, Ciencia y Tecnología - Serie de Conferencias, 2*, 432.

Trivedi, S., Mehta, K., & Sharma, R. (2021). Systematic literature review on application of blockchain technology in e-finance and financial services. *Journal of Technology Management & Innovation, 16*(3), 89-102.

Ye, X., Wang, S., Wang, H., Wei, Q., Yang, T., & Tao, Y. (2023). Application of knowledge graph in financial information security strategy. In *Proceedings of the 8th International Conference on Cyber Security and Information Engineering* (pp. 188-192).

Tan, V., Cheh, C., & Chen, B. (2021). 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.

Nasution, E. R. (2023). The role of regulatory authority in maintaining financial security in the Indonesian banking sector: A legal framework analysis. *The International Journal of Politics and Sociology Research, 11*(3), 386-397.

Liu, X., Ahmad, S. F., Anser, M. K., Ke, J., Irshad, M., Ul-Haq, J., & Abbas, S. (2022). Cybersecurity threats: A never-ending challenge for e-commerce. *Frontiers in Psychology, p. 13*, 927398. https://doi.org/10.3389/fpsyg.2022.927398

Cali, U., Kuzlu, M., Sebastian-Cardenas, D. J., Elma, O., Pipattana-somporn, M., & Reddi, R. (2024). Cybersecure and scalable, token-based renewable energy certificate framework using blockchain-enabled trading platform. *Electrical Engineering, 106*(2), 1841-1852.

Pahlevan, M., & Ionita, V. (2022). Secure and efficient exchange of threat information using blockchain technology. *Information, 13*(10), 463. https://doi.org/10.3390/info13100463.

Oosthoek, K., & Doerr, C. (2020). Cybersecurity threats to bitcoin exchanges: Adversary exploitation and laundering techniques. *IEEE Transactions on Network and Service Management, 18*(2), 1616-1628.

Kariuki, P., Ofusori, L. O., & Subramaniam, P. R. (2023). Cybersecurity threats and vulnerabilities experienced by small-scale African migrant traders in Southern Africa. *Security Journal*, 1-30. https://doi.org/10.1057/s41284-023-00312-4.

Hogan, K. M., Olson, G. T., Mills, J. D., & Zaleski, P. A. (2023). An analysis of cyber breaches and effects on shareholder wealth. *International Journal of the Economics of Business, 30*(1), 51–78.

Kayode-Ajala, O. (2023). Applications of cyber threat intelligence (CTI) in financial institutions and challenges in its adoption. *Applied Research in Artificial Intelligence and Cloud Computing, 6*(8), 1-21.

Bianchi, D., & Tosun, O. K. (2019). Cyber attacks and stock market activity. *WBS Finance Group Research Paper*, (251), 1-50.

Gagliani, G. (2020). Cybersecurity, technological neutrality, and international trade law. *Journal of International Economic Law, 23*(3), 723–745.

Kotey, J. N. (2023). A functioning code may not be secure: A preliminary study on the students' complacency with secure coding.

Kurihara, I., Kurihara, J., & Tanaka, T. (2024). A new security measure in secret sharing schemes and secure network coding. *IEEE Access*.

Niinivirta, N. (2023). Software developers' secure coding needs in the financial sector: A case study.

Singleton, L., Zhao, R., Song, M., & Siy, H. (2020). Cryptotutor: Teaching secure coding practices through misuse pattern detection. *In Proceedings of the 21st Annual Conference on Information Technology Education* (pp. 403–408).

Pikulin, V., Kubo, D., Nissanka, K., Bandara, S., Shamsiemon, M. A., Yasmin, A., Jayatilaka, A., Madugalla, A., & Kanij, T. (2023). Towards developer-centered secure coding training. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)* (pp. 24-31). IEEE.

Pruemmer, J., van Steen, T., & van den Berg, B. (2023). A systematic review of current cybersecurity training methods. *Computers & Security*, 103585.

Hayashi, M. (2021). Secure physical layer network coding versus secure network coding. *Entropy, 24*(1), 47.

Liu, Q., Zhang, W., Ding, S., Li, H., & Wang, Y. (2020). Novel secure group data exchange protocol in smart home with physical layer network coding. *Sensors, 20*(4), 1138.

Günlü, O., Schaefer, R. F., Boche, H., & Poor, H. V. (2022). Private key and decoder side information for secure and private source coding. *Entropy*, *24*(12), 1716.

Charoenwet, W., Thongtanunam, P., Pham, V.-T., & Treude, C. (2024). Toward effective secure code reviews: An empirical study of security-related coding weaknesses. *Empirical Software Engineering*, *29*(4), 88.

Zhang, Y. (2023). Secure coding practice in Java: Automatic detection, repair, and vulnerability demonstration.

Ryan, I., Roedig, U., & Stol, K.-J. (2023). Measuring secure coding practice and culture: A finger pointing at the moon is not the moon. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)* (pp. 1622-1634). IEEE.

Xing, G., Liang, G., & Salem, T. (2024). Interactive learning modules for fostering secure coding proficiency in introductory programming courses. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2* (pp. 1859-1860).

Saxena, S., & Agarwal, D. (2019). A model to quantify effectiveness assessment model through security and correctness assessment for adoption of the e-procurement. In *Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM)*, Amity University Rajasthan, Jaipur, India.

Nisha, T., & Khandebharad, A. (2022). Migration from DevOps to DevSecOps: A complete migration framework, challenges, and evaluation. *International Journal of Cloud Applications and Computing (IJCAC)*, *12*(1), 1-15.

Chan, M. O., Yazid, S., et al. (2024). A novel framework for information security during the SDLC implementation stage: A systematic literature review. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, *8*(1), 88-99.

Umeugo, W. (2023). Secure software development lifecycle: A case for adoption in software SMEs. *International Journal of Advanced Research in Computer Science*, *14*(1).

Otieno, M., Odera, D., & Ounza, J. E. (2023). Theory and practice in secure software development lifecycle: A comprehensive survey. *World Journal of Advanced Research and Reviews*, *18*(3), 53-78.

Omar, A., Alsadeh, A., & Nawahdah, M. (2022). Adherence to secure software development lifecycle.

Maher, Z. A., Shah, A., Chandio, S., Mohadis, H. M., & Rahim, N. (2020). Challenges and limitations in secure software development adoption: A qualitative analysis in Malaysian software industry prospect. *Indian Journal of Science and Technology*, *13*(26), 2601-2608.

Casola, V., De Benedictis, A., Mazzocca, C., & Orbinato, V. (2024). Secure software development and testing: A model-based methodology. *Computers & Security*, *137*, 103639.

Ashenden, D., & Ollis, G. (2020). Putting the sec in develops: Using social practice theory to improve secure software development. In *Proceedings of the New Security Paradigms Workshop 2020* (pp. 34-44).

Rajapakse, R. N., Zahedi, M., & Babar, M. A. (2021). An empirical analysis of practitioners' perspectives on security tool integration into DevOps. In *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (pp. 1-12).

Leppänen, T., Honkaranta, A., & Costin, A. (2022). Trends for the DevOps security: A systematic literature review. In *International Symposium on Business Modeling and Software Design* (pp. 200-217). Springer.

Grigorieva, N. M., Petrenko, A. S., & Petrenko, S. A. (2024). Development of secure software based on the new DevSecOps technology. In *2024 Conference of Young Researchers in Electrical and Electronic Engineering (ElCon)* (pp. 158-161). IEEE.

Sharma, P. (2024). DevSecOps integration-security in the software delivery pipeline: Exploring the integration of security practices into the software delivery pipeline to ensure secure software development practices. *Australian Journal of Machine Learning Research & Applications*, *4*(1), 46–54.

Abohatem, A. Y., Ba-Alwi, F. M., & Al-Khulaidi, A. A. (2023). Suggestion cybersecurity framework (CSF) for reducing cyber-attacks on information systems. *Sana'a University Journal of Applied Sciences and Technology*, *1*(3).

Abos, P. (2024). DevSecOps for secure software development in the cloud.

Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2021). Challenges and solutions when adopting DevSecOps: A systematic review. *Journal of Information and Software Technology*.