International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 5, Issue 1, 104-116, 2024
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V5I1P110

Original Article

Secure Software Development Life Cycle (SSDLC) for AI-Based Applications

Sunil Anasuri
Independent Researcher, USA.

Abstract *- The high rate of incorporation of Artificial Intelligence (AI) in software systems transformed industries; however, they brought new and complicated issues about security. The data poisoning, adversarial inputs, model inversion, and biased decision-making vulnerabilities to AI are not completely covered in traditional Software Development Life Cycles (SDLCs). This article proposes an end-to-end Secure Software Development Life Cycle (SSDLC) with special consideration to AI-based platforms and software. It also discusses the need to incorporate security principles at every level of the process, starting with the requirements and preprocessing of secure data, through model development and deployment, and up to post-deployment monitoring. The paper also tests threat modeling approaches adapted to applying to AI pipelines, ripe attack paths and suggestions to defenses. Using industry frameworks such as NIST AI RMF and Google SAIF, along with examples of similar frameworks, the paper demonstrates best practices for integrating privacy, fairness, and accountability into the lifecycle of an AI system. A practical example of a case study of a global financial institution shows the practical value of the adoption of SSDLC of AI, such as decreased security alerts, enhanced developer productivity, and increased regulatory compliance. The paper finishes with a conclusion of the problems that are currently being addressed in relation to the security of AI workflows and a look at the future, including DevSecOps integration and automated threat identification as part of MLOps. The work aims to present a systematic, security-aware model for developing resilient and trustworthy AI applications.*

Keywords *- Threat Modeling, MLOps Security, Adversarial Attacks, Secure Deployment, Compliance, DevSecOps, SSDLC Framework.*

1. Introduction

Artificial Intelligence (AI) has developed fast into an emerging research field and an innovative technology that changes the world by transforming the most significant industries such as healthcare, finance, transportation, and defense. With the growing integration of AI technologies into decision-making and real-time operations, the security, reliability, and ethical conduct of AI systems have become a priority to address. The current methods of Software Development Life Cycle (SDLC) are, however, not entirely capable of ensuring the kind of security required by AI models. [1-3] The reason behind these adverse issues is the fact that AI is both data-driven and probabilistic, which means that its vulnerabilities could also originate in training data, model architecture, and deployment environment rather than merely the source code. A Secure Software Development Life Cycle (SSDLC) created specifically to contend with these new threats is necessary when it comes to AI-based applications.

A SSDLC focused on AI should account for a wider threat model, including the presence of adversary-based attacks that alter the behavior of models, training data poisoning, untoward access to model parameters, and bias in the predictions that could lead to discriminatory results. Moreover, the use of AI on cloud platforms and on edge devices introduces new points of attack, necessitating full-fledged security solutions across the life cycle of development and deployment in addition to after-deployment watching. This paper proposes an enhanced SSDLC framework that formally integrates security and trustworthiness throughout all phases of AI software development. The framework addresses the need to facilitate the creation of AI systems that are not only functional and precise but robust against threats and that follow ethical principles by using good design principles (including security), constant monitoring of risks, and alignment with AI governance standards. This method indicates the developing understanding that the safety and responsible behavior of AI should be ingrained as part (not an addition) of the AI development cycle.

2. Background and Related Work

Artificial intelligence has altered the access orthodoxy of software and has brought about the necessity of distinctive security routines. In this section, the researcher [4-6] examines the premises of traditional software development, the emergence of AI-

related risks, and how security-related frameworks for AI are currently operating to address them, given the emerging nature of these risks.

2.1. Traditional Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) offers an organized approach towards creating high-quality, cost-effective, and secure software systems. It normally comprises 6 main stages. The planning stage entails the task of developers and interested parties to identify what the project scope should be, the project objectives, resources, and project time frame. There is then the design phase, in which the design of the architecture, technologies, and data structures is chosen and aligned with both the functional and non-functional requirements. The implementation phase will entail programming and assimilating the programs. The system is tested during this step, and it goes through intensive quality verification, functional, security and performance testing, to ascertain that it is up to specifications. The deployment stage involves testing the software in the production environment. The maintenance phase, the last of all phases, ensures long-term operability due to repairs based on user feedback, patching of vulnerabilities, and the introduction of updates. Although this linear or iterative model has been adequate for conventional use, it is no longer adequate in terms of complexity and dynamism of AI-based systems.

2.2. AI-Specific Threats and Vulnerabilities

In comparison to conventional systems, AI models leverage data and statistical tendencies instead of deterministic reasoning. This data dependency creates its security issues. Poisoning attacks imply the injection of malicious samples into training sets to target the model behavior or incur bias. Adversarial assaults exploit AI inference by adding small changes to input data to deceive the model and frequently do not affect human perception. Attackers may employ techniques such as model inversion and exfiltration to infer training data or other sensitive information about a black-box model. There is an injection of prompts in generative AI programs, which can result in the manipulation of the output response. Additionally, there is exposure to the supply chain, as it relies on third-party pretrained models, or access to the Application Program Interface (APIs) that could be compromised. The advent of deepfakes and synthetic media has also exacerbated the problem of trust, giving adversaries a chance to create realistic multimedia fakes with the assistance of AI, thereby undermining authenticity and eroding the moral sense of ethical responsibility.

2.3. Security in Machine Learning Pipelines

As organizations embrace Machine Learning (ML) pipelines, it has become more important to secure the various phases involved, such as data ingestion as well as inference. Exposure of sensitive machine learning data during collection and preprocessing. Preprocessing can leak sensitive information to the model or introduce prejudiced or malicious inputs, which can compromise the model's integrity. The training stage is vulnerable to theft of intellectual property and unreasonable data usage of secret algorithms. Model tampering, endpoint exploitation, and revealing inference data are among the risks that the deployment and inference stages encounter. This requires a comprehensive end-to-end security solution, including encryption, identity access management, web-based secure access, and logging and real-time monitoring. Hardware-based safeguards such as technologies like Trusted Execution Environments (TEEs) and confidential computing can be used to protect data and model integrity. These defences must be integrated into the broader machine learning operations (MLOps) stream to establish secure and compliant AI systems.

2.4. Existing Secure Development Frameworks for AI

In response to such mounting concerns, various organizations and institutions have proposed frameworks of secure development as applicable to AI. The NIST AI risk management framework (AI RMF) gives more systematic advice on how to establish, assess, and mitigate AI-specific risks within its entire lifecycle. It highlights openness, responsibility and conformance to societal principles. MITRE ATLAS is a threat matrix of adversarial machine learning tactics that helps security professionals foresee and thwart attacks. The Microsoft AI Security Framework groups the best practices into five transformational principles: security, privacy, fairness, accountability, and transparency and promotes governance approaches to the trustworthy deployment of AI. In the same spirit, Google Secure AI Framework (SAIF) is working on ensuring strong security measures along the AI development process emphasizing supply chain integrity and shared risk reduction. All these frameworks can be explicitly described as industry transformation towards more holistic, secure, and ethical AI development paradigms, the basis of a standardised Secure Software Development Life Cycle in AI application direction.

3. Foundations of SSDLC for AI-Based Applications

The further rise of AI systems in both vital infrastructure and -level applications demands that security and ethical considerations mustntegrated throughout the stages of the ware implimplementation [7-10] The Secure Software Development Life Cycle (SSDLC) of AI-based applications represents a prescribed framework that helps guarantee that security, privacy and trustworthiness are inherent elements of AI system development. The AI-adapted SSDLC method, in contrast to conventional

SSDLC methods, provides insights into the special features of AI in the form of dependence on substantial data stores, model generalization, non-determinism, and ongoing learning by incorporating controls, governance and compliance checks at the domain-specific level in every stage.

3.1. Principles of SSDLC

The graphic known as the SSDLC Process Wheel summarizes the key concepts of a Secure Software Development Life Cycle by depicting the basic component processes of the life cycle as a continuous cycle, defining each process in relation to each other, indicating how secure software engineering is an iterative process, which considers all aspects of software engineering. At the core of the model are the conventional development phases: Requirements, Design, Development, Testing and Deployment. This core will be surrounded by an outer ring that will denote the integrated security processes, which are Risk Assessment, Threat Modeling, Static Analysis, Security Testing and Security Assessment. This graphic illustration supports the point that security is not a secondary consideration that is applied after completion of various developmental phases, but rather an in-built provision.

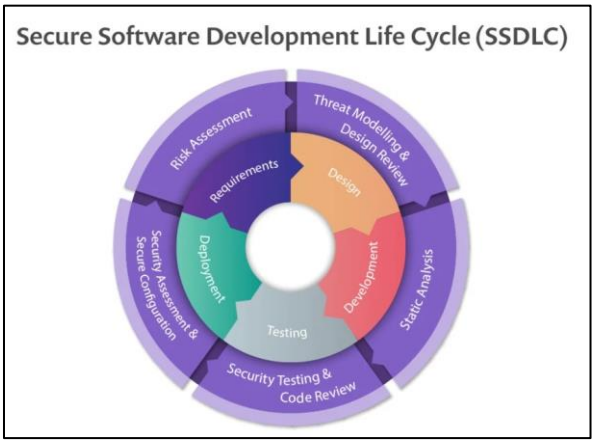

Fig 1: Standard Secure Software Development Life Cycle (SSDLC) with integrated security checkpoints

The outer ring continues to show the nature of security as a layer and is integrated into every development stage. With every development stage, there is a corresponding security activity that promotes the identification, mitigation, and management of vulnerabilities. The threat modeling, as well as design reviews, can be used as an example during the design phase; it will prevent architectural vulnerabilities. Static code analysis and security testing cause code quality and robustness testing to be conducted during development and testing. The wheel in this manner speaks about a recurrent circle of enhancement and alertness, which brings an understanding of the proactive character of SSDLC. In applications of AI, these principles of foundations remain highly applicable but must be extended to address AI-specific risks, including model poisoning, adversarial attacks, and data governance.

3.2. Key Phases of SSDLC Adapted for AI

To implement these principles, every step of the SSDLC would need to be modified to address the issues of AI. This methodology will enable safe AI conception, deployment, and maintenance.

3.2.1. Requirement and Risk Assessment

The initial stage involves identifying security, privacy, and ethical requirements, as well as functional needs. This comprises familiarity with the use case, data sources, scenarios of abuse and regulatory requirements (e.g., GDPR, HIPAA, EU AI Act). The threats or potential attack vectors and vulnerabilities specific to AI, such as adversarial machine learning risks, data leakage, and misuse of model outputs, can be discovered with risk assessment tools and threat modeling techniques like STRIDE or MITRE ATLAS. This phase establishes the security threshold of the rest of the phases.

3.2.2. Secure Data Collection and Preprocessing

The training set of AI systems cannot be more reliable than the data on which they were trained. This stage focuses on safe and responsible data sourcing, where datasets do not include intentional malicious data, bias and Personally Identifiable Information (PII). Data sanitization, data validation, and logging solutions are applied to avoid data poisoning and ensure provenance. Data in storage and transfer are encrypted and access controlled to avoid unauthorized access. The preprocessing algorithms are also examined to ensure that they do not present weak points or leak sensitive patterns.

3.2.3. Secure Model Development

At this step, AI is created and trained securely and robustly. The codebase of machine learning will enforce secure coding practices. Adversarial training, differential privacy, and regularization methods are some of the defensive measures that can be used to enhance resilience and lower the prospect of overfitting or model inversion. Developers also use explainable tools (e.g., SHAP, LIME) to track the behaviour of the model and raise an alarm in case of strange correlations. Model development is made transparent and is traceable through security peer reviews and version control audits.

3.2.4. Model Testing and Validation

In addition to traditional functional testing, the AI systems must be thoroughly validated to demonstrate that they perform and are robust in a variety of and potentially unfavorable operating environments. These are adversarial robustness, fairness across demographic groups, privacy leakage, and interpretability. Artificial testing data, fuzzing, and red-team activities also imitate attacks and challenge the limits of the model. Validation metrics should contain not only accuracy, but also indicators that refer to trust, e.g., model confidence, fairness measurements, and robustness measures.

3.2.5. Deployment with Security Controls

The hardened infrastructure and stringent controls should be in place when deployed in real-world conditions. Inference pipelines and model endpoints are secured using secure APIs, encrypted communication channels, authentication protocols and Role-Based Access Control (RBAC). Containerization and sandboxing can segregate AI services in order to minimize the persistent damage of any possible hack. Also, they can impose model watermarking, model integrity tests to detect unauthorized modifications or distribution.

3.2.6. Monitoring and Incident Response

AI systems, once implemented, should be continuously monitored to identify anomalies, drifts, and indications of compromise. The model inputs, outputs and decision rationale can be tracked using logging mechanisms, with the ability to do traceability and post-incident auditing. The anomaly identification models and Intrusion Detection Systems (IDS) are capable of raising suspicious activity in real time. A proper incident response strategy specific to AI use cases, i.e., adversarial input detection or managing data breaches, is necessary. Lifecycle management practices, such as frequent retraining, patching and maintenance, and retirement planning, during this phase can also ensure the long-term enforcement of system integrity and compliance.

4. Threat Modelling in AI Development

Threat modeling is an offensive security discipline that is used to detect, assess and alleviate threats to a system before they can be encountered. [11-14] Threat modeling can be modified to deal with the unusual properties of AI systems, including a dependence on large-scale data, complex models, and probabilistic results in AI development. Conventional threat modeling methods are insufficient when used in AI because new forms of attack surfaces expose data pipelines, training algorithms, and inference systems. This part discusses customized threat modeling techniques and pinpoints typical attack vectors in AI, as well as some safe design approaches to mitigate emerging threats.

4.1. Threat Modelling Techniques for AI Systems

Threat modeling applied to AI would entail listing the components of a system, appreciating how they interact, and determining where malicious entities are likely to violate security, privacy, or integrity. Among the most common methods is STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege), which can be modified to address the threats particular to AI, such as model tampering and data poisoning. Additionally, MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems) provides a concise matrix of adversarial techniques targeting machine learning, which can be useful in developing AI-aware threat models. Visualization tools, such as Diagram-based tools such as Data Flow Diagrams (DFDs), can be used to visualize AI data pipelines and model architecture and enable the security analyst to identify vulnerabilities through which threats may be injected into the system. The aim is to have proactive integration of the security controls into every component of AI, making it less likely to be exploitable.

4.2. Common Attack Vectors in AI Applications

Dynamic AI systems and their reliance on outside data create additional avenues of attack due to their dynamic nature, which is not seen in traditional software. These weaknesses may be manipulated during different phases of the AI lifecycle, all the way from training to inference.

4.2.1. Data Poisoning Attacks

Data poisoning refers to a severe threat in which hackers may introduce fraudulent or inaccurate data into the training dataset. Such manipulation can skew the model, lower its performance, and backdoor it with triggers that are activated under certain

conditions. For example, during an image classification task, adding incorrectly classified images to a training set may result in future misclassifications by the model. Poisoning may be availability (disrupting overall accuracy) or targeted (impacting specific predictions). Such attacks could be hard to detect in AI data pipelines where data is distributed and opaque since there is a lack of proper data validation, detection of anomalies, and monitoring provenance mechanisms.

4.2.2. Model Inversion and Extraction

Model inversion and extraction attacks are intended to rebuild necessary information or copy a proprietary model through repeated questions. Model inversion, in model inversion, an adversary translates the outputs of a model into a potentially sensitive training data item- restoring patient images to a medical AI system. Model extraction is the process of generating a working copy of a model, which reverses engineers ' decision boundaries of the model. These attacks pose a threat to intellectual property, user confidentiality, and compliance with regulations such as the GDPR. They are particularly widespread in Machine Learning As A Service (MLaaS) systems, where the model behavior can come out accidentally through publicly exposed Application Programming Interfaces (APIs).

4.2.3. Adversarial Examples

Adversarial examples are carefully designed inputs targeted at deceiving AI algorithms into incorrect predictions, typically with only small modifications that cannot be recognized as such to human eyes. As an example, an image classifier could easily label a tampered stop sign as a speed limit sign. The existence of these perturbations demonstrates that AI decision-making is fragile, as the mathematical structure of the model is exploited. Adversarial attacks could be white-box (the model is known to the attacker) and black-box (the attacker may only use the outputs of the model). They pose a constant menace, and their success with many AI platforms presents a significant problem, particularly in safety-sensitive areas such as autonomous vehicles and biometric security.

4.3. Secure Design Strategies

To address these risks, the process of building security into the AI process lifecycle should be incorporated. The most essential interventions are robust data curation, i.e., utilising reliable sources of data, employing data purification, and verifying data set integrity. To develop models, methods such as adversarial training (training on perturbed data) and differential privacy (adding output noise) improve both robustness and privacy. Unauthorized copies of the model or attacks on the models can be detected with the help of model watermarking and behavioral fingerprinting. Privacy APIs are secured through rate limitation and access control, protecting against inference and denial-of-service attacks. Besides, explainability tools can be used to uncover unexpected behavior of the model early, and this assists in debugging as well as threat detection. Ultimately, integrating secure design principles with systematic threat modelling will enable AI systems to be not only secure but also innovative and trustworthy by design.

5. Security Integration across SSDLC Phases

5.1. Architectural Overview of SSDLC in AI Systems

The Secure Software Development Life Cycle (SSDLC) is suitable for AI-based systems. The flowchart records the development pathway beginning with Requirement and Risk Assessment to Compliance and Governance, defining the security checkpoint incorporated in each phase. [15-18] It provides an overview that visually links important security pieces, for example threat modeling, data quality checks, secure model development, adversarial testing, and real-time monitoring into one process. These phases are not independent of each other. Still, they are connected through feedback loops, thereby maintaining a constant verification and modification of new issues and changing regulations related to compliance.

Remarkably, the figure highlights the repeated and intertwined qualities of a secure AI development. For example, observations made during Monitoring and Incident Response (Phase 6) can be applied to Requirement and Risk Assessment (Phase 1) to improve threat models and update compliance policies. Secure Deployment and MLOps is an important part of the role, as it relies on secure containerization, CI/CD pipelines, API protection and access to control, which are also significant to the preservation of integrity throughout the model inference. Testing and validation are also represented as encompassing security, robustness, and fairness domains, which, in addition to adversarial resilience, are tested. The various aspects of life are described using a comprehensive lifecycle perspective, which is what I mean by a complete lifecycle view.

5.2. Security in Requirements Gathering

Security needs to enter the stage of SSDLC as early as possible, specifically the requirements gathering phase. In the case of AI systems, this step extends beyond the usual specification of functionality and performance requirements to include privacy requirements, regulatory compliance (e.g., GDPR or HIPAA), threat detection, and the expectations of stakeholders. Security-related requirement elicitation gets done where data scientists, developers, legal teams, and end-users participate to define and

capture privacy requirements, possible attack surface, ethical considerations, and data sensitivity. It is here that threat modeling takes place and lists out possible threats, and how they can be addressed in advance. Moreover, security requirements should be iterative and change systematically as the system grows or when new risks continue to appear in downstream cycles, such as model deployment or user feedback loops.

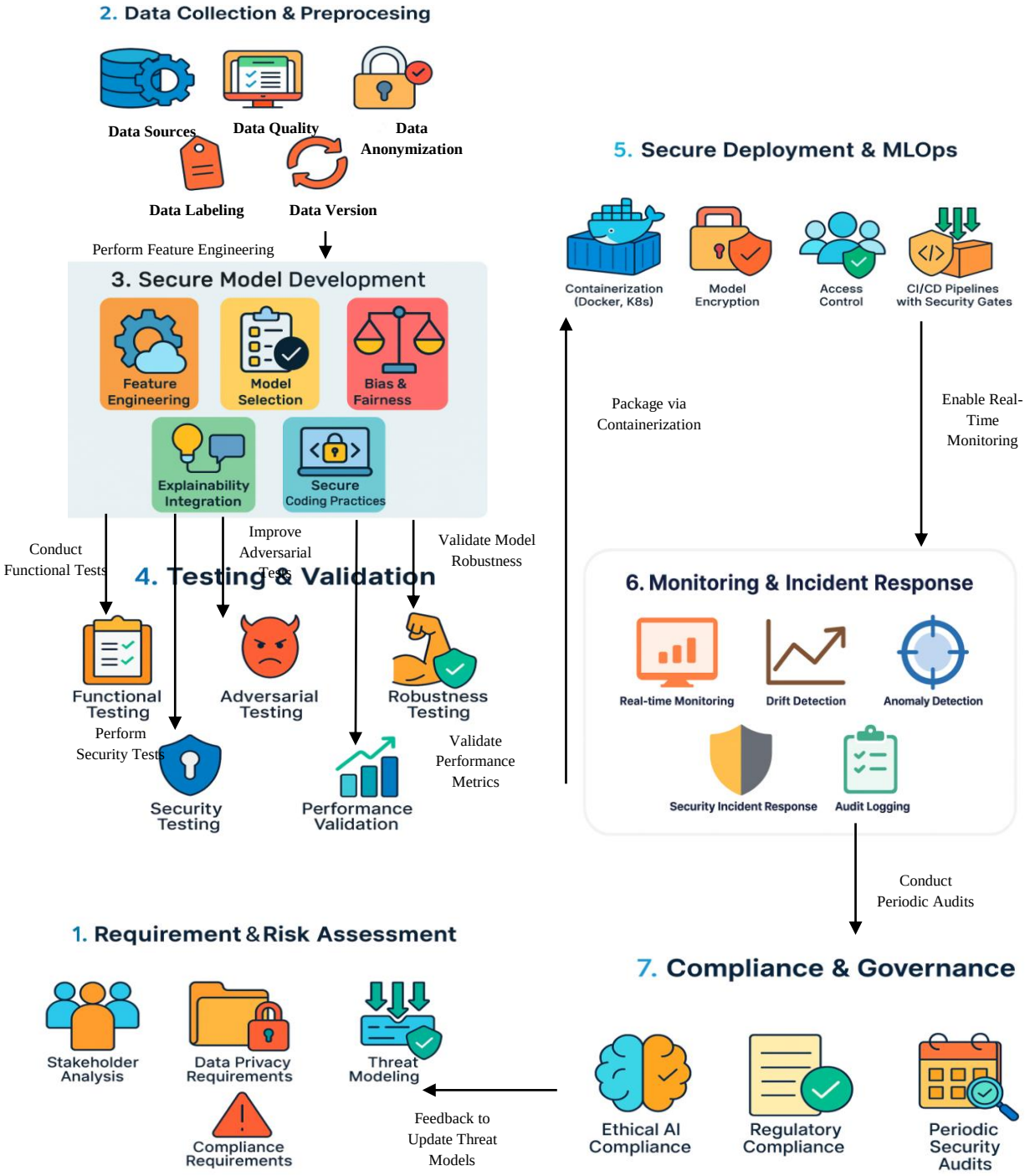

Fig 2: Secure Software Development Life Cycle (SSDLC) for AI-Based Applications

5.3. Privacy-Preserving Data Management

Application use depends greatly on large-scale datasets, many of which contain personally identifiable or sensitive information. Thus, strict data governance processes should be enacted so as to promote data confidentiality and safety. Privacy-preserving data management comprises secure data collection procedures, data anonymization, data-at-rest and data-in-motion encryption, and metadata which is used to follow data provenance and versioning. Labelling should also be secure to prevent information leakage at a stage of preprocessing. Model training without exposing raw data may also use methods such as differential privacy, federated learning, and homomorphic encryption, especially when there are distributed data sources or sensitive application areas, like in healthcare or finance.

5.4. Secure Model Training Protocols

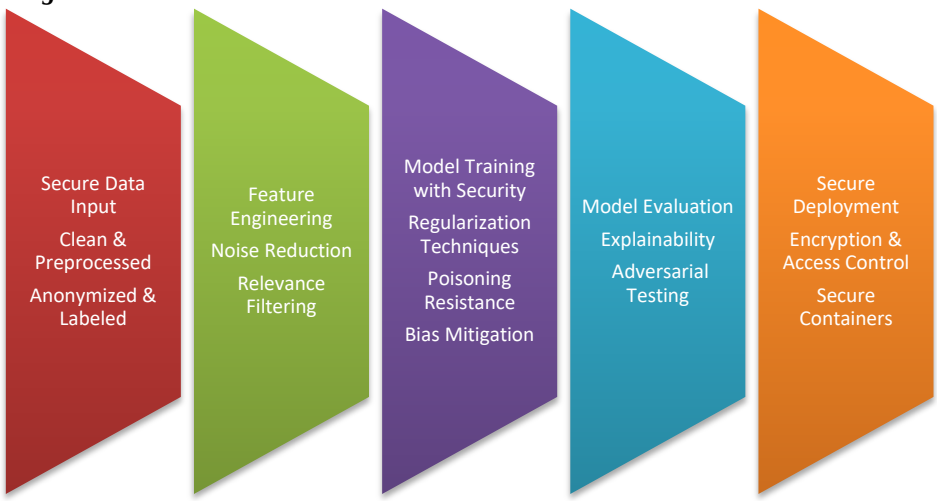

Fig 3: Secure Model Lifecycle Within SSDLC

The security of model training is very important, as this is where the process is susceptible to data poisoning, model theft, and a breach of integrity. [19-21] Training of models ought to take place in limited, access-restricted settings with validated datasets. Training protocols Secure model training may extend to cryptographically verifiable data pipelines, adversarial training to enhance model robustness, and fairness and transparency verification to demonstrate model ethical suitability. Reputational and legal risks can be reduced through bias and fairness audits and, especially where models are trained on heterogeneous data. Logging and monitoring model behavior that is used during training also ensures transparency and enables rollback in situations with suspicious patterns or anomalies.

5.5. Robust Validation and Testing Techniques

SSDLC validation and testing of AI go much beyond functionality testing. The security testing will be scalable, including adversarial robustness testing, penetration testing of the model API, and synthetic stress testing under high loads or extreme conditions. Validation should also involve fairness testing, measurements of performance on a variety of inputs, and conformance testing against systems of predefined regulators. Adversarial AI testing focused on areas where AI is vulnerable to unseen manipulation of inputs to trigger the failure of the model is vital in fields such as image recognition, fraud detection, and autonomous systems. Explainability tools, such as SHAP or LIME, must also be employed in the testing process to ensure that the decision-making logic is logically explicable and explainable.

5.6. Secure Deployment Practices (e.g., MLOps with Security)

The deployment of AI systems should be secure in a way that ensures it is tightly integrated with MLOps pipelines, thereby achieving security by design. Such measures enforce environmental consistency and isolation through containerization technologies (e.g., Docker, Kubernetes), encryption of model files, access controls, and protection against abuse or extraction attacks by defending inference APIs. The recommendation is to add security gates to the CI/CD pipeline and perform vulnerability scanning prior to deploying updates to production. Rollback options and logs should also be supported by the deployment process, so that any security issue or anomaly can be traced. Supporting security in MLOps is not only a time-saving feature at the update development stage, but it can also eliminate the attack surface in production.

5.7. Continuous Security Monitoring and Updates

Security in the AI system should not be considered as something that should only be applied once. When deployed, systems should be continually checked to ensure that there is no model drift, adversarial conduct, or API misuse, as well as infrastructural weaknesses. Real-time monitoring and anomaly picture posters, along with drifting detectors, assist in maintaining model performance and security integrity over time. Incidents must initiate a semi-automatic or manual incident response process, which includes a rollback, notifications to stakeholders, and revisions to threat models. Additionally, a regular audit and compliance plan should be implemented to ensure the continuous adherence of the enterprise to laws and ethics. Adaptive learning systems will include adaptive learning systems, so that with a change in threat, the AI system will be robust enough to respond.

6. Case Study: Implementing SSDLC in AI-Powered Applications

6.1. Case Overview and Context

Operating in an excessively regulated financial environment, one of the most renowned international banks had to cope with significant issues regarding maintaining security of applications powered by AI. Its digital ecosystem was too complex, resulting in the ability to generate millions of security alerts per day, with a vast majority of them turning out to be false positives. The Security Operations Center (SOC) was overwhelmed by the quantity of managing these alerts to the point where actual threats might be missed. Simultaneously, developers had to spend nearly 80% of their time patching vulnerabilities, thereby thwarting innovation and impeding the release of products. Security and engineering teams had to operate under tremendous pressure because of this contradiction between regulatory compliance and digital transformation.

6.2. Application of the SSDLC Framework

To address these issues, the bank implemented a Secure Software Development Life Cycle (SSDLC) framework installed with Generative AI (GenAI) functions. GenAI could be a breakthrough because it can help extend regulatory policies into operational security safeguards, which are incorporated into the development pipeline. The lifecycle, from requirements gathering to deployment and monitoring, was covered by the automation and distribution of preventive, detective, responsive, and corrective mechanisms. GenAI was also used to sort and rank warnings. The system applied risk-based heuristics and context to analyze possible notifications, and only notifications of vulnerabilities with the highest severity were escalated to dedicated response teams, which may handle DDoS attacks or malware, in order to eliminate the need to manually process millions of notifications every day. Additionally, regular security processes, such as account auditing, encryption key rotation, and real-time risk assessment, became automated, leading to a significant improvement in operational resilience.

6.3. Application of the SSDLC Framework

The effects of the SSDLC implementation using AI were tremendous. The level of alerts became significantly lower, as there were now only several tens of important alerts in a day (since millions of alerts were commonplace earlier). This change enabled developers to focus on where it needs to be: innovation and value delivery. Security personnel also shifted their focus to proactive approaches instead of managing reactive alerts, with more effort going into threat modeling and advanced vulnerability research. According to quantitative analysis, there is a significant Return On Investment (ROI), whereby the implementation of AI-augmented security is covered by a lack of risk exposure, compliance nonconformance, and inefficiency to some extent. There was also a significant decrease in repeat vulnerabilities in the bank, signifying the feasibility of proactive and automated remediation in the long run. The case focuses on the security benefits and organizational culture issues related to the embedded intelligent SSDLC processes that go beyond security posture and have a positive impact on team morale, development cycles, speed, and regulatory compliance.

Table 1: Security Impact of SSDLC Implementation

Aspect	Details	Impact
Problem	Volume of security alerts, false positives, inefficient processes, developer frustration	Millions of daily alerts; 80% of developer time spent on remediation
Solution	GenAI-automated controls, prioritization, protocol automation across SSDLC phases	Significant reduction in manual workload; faster, more secure development
Automated Security Controls	Preventive, detective, responsive, and corrective controls built into pipelines	Proactive vulnerability identification; continuous security monitoring
Threat Prioritization	GenAI filtered and routed only critical vulnerabilities to appropriate response teams.	Resources focused on high-risk threats; reduced false positives
Security Protocol Automation	Automated tasks like account disabling, key rotation, and live risk assessment	Streamlined operations; improved regulatory compliance
Result	SSDLC + GenAI improved alert handling, team efficiency, and return on investment	<10 critical alerts/day; improved developer productivity and risk mitigation

[1] https://www.nexgencloud.com/blog/case-studies/using-generative-ai-for-software-security-in-banking-a-case-study
[2] https://www.ijcesen.com/index.php/ijcesen/article/view/2384
[3] https://www.qualitestgroup.com/insights/blog/how-ai-powered-diagnostics-can-transform-your-sdlc-in-2024/
[4] https://www.v2solutions.com/whitepapers/ai-augmented-sdlc-whitepaper/
[5] https://www.palo-it.com/en/blog/applied-ai-sdlc
[6] https://www.akto.io/learn/ai-powered-secure-sdlc
[7] https://www.champsoft.com/2024/10/24/incorporating-ai-into-the-secure-software-development-life-cycle-ssdlc/
[8] https://www.linkedin.com/pulse/enhancing-secure-software-development-lifecycle-sdlc-ai-mohamed-riyas-wallf
[9] https://meridianjournal.in/index.php/IMJ/article/view/84
[10] https://www.securecodewarrior.com/article/2024-predictions
[11] https://www2.deloitte.com/content/dam/Deloitte/us/Documents/risk/us-ssdl-for-precision-ai-paper.pdf
[12] https://www.linkedin.com/pulse/integrating-ai-driven-security-sdlc-using-open-source-rai-ludif

7. Challenges and Future Directions

7.1. Evolving Threat Landscape for AI

With AI technologies moving toward becoming a common practice, an ever-evolving pursuit of cyber threats against AI systems is gaining ground. The structures of traditional security systems are usually inadequate when it comes to securing AI systems because their attack surfaces are unlike those of systems, which have parameters and training data as well as decision logic. The new methods are data poisoning attacks in training, evasion methods with adversarial examples, model inversion to extract sensitive data and prompt injection into generative AI models. These threats are continually changing, frequently responding to defenses put in place, thus prompting continuous threat intelligence, model hardening, and sound validation measures. Moreover, using third-party pre-trained models poses some vulnerabilities in the supply chain, which will require stringent provenance verification procedures and ensure integrity verification processes.

7.2. Automation of Security in AI Pipelines

In order to safely scale AI, security tasks have to be automated in machine learning pipelines. The complexity and the pace of contemporary AI workflows can no longer be managed through manual workflows. Automation involves incorporating static and dynamic code checks, safe configuration management, automated bias detection, and continuous vulnerability scanning during the AI/ML life cycle. Also, real-time drift detection and anomaly alerting can now be provided with automated model monitoring tools. Nonetheless, there are still problems, including reducing false positive rates, delivering context-aware threat differentiation, and delivering automation logic that is not a new vulnerability. The application of AI-based security agents that can learn behavior patterns from previous threats is promising since such systems are always tested for their robustness and reliability.

7.3. Integration with DevSecOps for AI

Security principles of DevSecOps, that is, security as part of shared responsibility integrated throughout the development lifecycle, are being scaled to AI systems, constituting a new paradigm called AIOps-Sec or MLOps-Sec. This convergence is in line with agile development in AI, but in such a way that security is never forgotten along the way. This integration is important through the implementation of security gates of CI/CD pipelines, code and model scanning, and audit examples through logging. As may be expected, although it can be desirable to keep up with rapid model iteration cycles with intense security checks, doing so without automation and concurrent validation procedures can be delaying. Moreover, the teams must undergo cultural changes to foster collaboration among developers, data scientists, and security specialists. Training and education concerning specific threats related to AI are becoming essential to cross-functional security maturation.

7.4. Research Gaps and Opportunities

Even though recent progress has been made, a few research voids still exist towards the enhancement of security in AI applications. Standardized benchmarks are needed to test the robustness, transparency and explainability of models in attack scenarios. There are not many shared standards of how to prove the ethical consequences of AI implementation, as well as how to estimate the degree of privacy threats related to the use of models. Moreover, explainable AI (XAI) needs further consideration in order to help achieve regulatory compliance in the most critical areas such as the financial sector and the healthcare industry. There are possibilities to examine how the concepts of zero-trust architectures can be applied to AI, including watermarking IP stored in models and creating self-healing ML systems capable of responding to attackers independently of human effort. Research under these areas will require interdisciplinary research between cybersecurity, machine learning, software engineering, and regulatory sciences to bridge such gaps.

8. Conclusion

Secure Software Development Life Cycle (SSDLC) of AI-based software represents a historical breakthrough in the development of a paradigm of executing software security in the era of smart systems. The present situation, where artificial intelligence is deeply penetrating the spheres of high interest like finance, healthcare, national infrastructure, etc., has changed the demands of colleges to the traditional SDLC methods of developing software, which is no longer enough to deal with complexity, unpredictability, and the risks involved in AI technologies. In this paper, the authors have provided a detailed framework of SSDLC to ensure security, privacy, and compliance considerations during the development process of AI, starting with secure data collection and model training to sound testing, secure deployment, and ongoing monitoring.

Through threat modeling, embedded security control automation using AI, along with the incorporation of MLOps and DevSecOps paradigms, and preventing AI vulnerabilities like data poisoning and adversarial manipulation, it will be possible to ensure that organizations can preempt risks and establish confidence in AI systems. The case study example visible in the real world demonstrates the practical advantages of this approach, including a decrease in alert fatigue, an increase in operational efficiency, and improved returns on investment. The field is, however, dynamic, and research should continue, threat intelligence should evolve, and a multi-disciplinary approach must be taken in order to ensure that progress in AI development stays secure, ethical and resilient. SSDLC and AI are not merely a technical requirement, but a strategic one to the purposeful and sustainable application of intelligent technologies.

References

[1] Jason Ricol. “AI for Secure Software Development: Identifying and Fixing Vulnerabilities with Machine Learning.” December 2022.

[2] Tim Abdiukov. “Secure-by-design principles in Agile SDLC: Leveraging Formal Verification and AI-Enhanced Code Review in CI/CD Environments.” World Journal of Advanced Engineering Technology and Sciences, 2023, 09(01): 494–503.

[3] Berghoff, C., Neu, M., & von Twickel, A. (2020). Vulnerabilities of connectionist AI applications: evaluation and defense. Frontiers in Big Data, 3, 23.

[4] Shilpi Singh & Saurabh Sambhav. “Application of Artificial Intelligence in Software Development Life Cycle: A Systematic Mapping Study.” In Lecture Notes in Networks and Systems – Micro-Electronics and Telecommunication Engineering, Springer, 2023, pp. 655–665.

[5] Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, Stefan Wagner. *Software Engineering for AI-Based Systems: A Survey*. arXiv:2105.01984; submitted May 5, 2021; considers 248 studies from 2010-March 2020.

[6] Jing, H., Wei, W., Zhou, C., & He, X. (2021, June). An artificial intelligence security framework. In Journal of Physics: Conference Series (Vol. 1948, No. 1, p. 012004). IOP Publishing.

[7] Mockus, A., & D. Weiss. (2000). Global and local drivers of software quality: A study of the Apache server. ACM SIGSOFT Software Engineering Notes, 25(5), 1–12.

[8] de Vicente Mohino, J., Bermejo Higuera, J., Bermejo Higuera, J. R., & Sicilia Montalvo, J. A. (2019). The application of a new secure software development life cycle (S-SDLC) with agile methodologies. Electronics, 8(11), 1218.

[9] Faris Mohamed Ahmed Hassan, Shampa Rani Das & Manzoor Hussain. “Importance of Secure Software Development for the Software Development at Different SDLC Phases.” August 2023.

[10] Fujdiak, R., Mlynek, P., Mrnustik, P., Barabas, M., Blazek, P., Borcik, F., & Misurec, J. (2019, June). Managing secure software development. In 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS) (pp. 1-4). IEEE.

[11] Karim, N. S. A., Albuolayan, A., Saba, T., & Rehman, A. (2016). The practice of secure software development in SDLC: an investigation through existing model and a case study. Security and Communication Networks, 9(18), 5333-5345.

[12] Mauri, L., & Damiani, E. (2022). Modeling threats to AI-ML systems using STRIDE. Sensors, 22(17), 6662.

[13] Barenkamp, M., Rebstadt, J., & Thomas, O. (2020). “Applications of AI in classical software engineering.” *AI Perspectives*, 2(1): 1–15.

[14] Arrey, D. A. (2019). Exploring the integration of security into software development life cycle (SDLC) methodology (Doctoral dissertation, Colorado Technical University).

[15] Khair, M. A. (2018). Security-centric software development: Integrating secure coding practices into the software development lifecycle. Technology & Management Review, 3(1), 12-26.

[16] Eian, I. C., Yong, L. K., Li, M. Y. X., & Hasmaddi, N. A. B. N. (2020). Integration of security modules in software development lifecycle phases. arXiv preprint arXiv:2012.05540.

[17] Shilpi Singh & Saurabh Sambhav. “Application of Artificial Intelligence in Software Development Life Cycle: A Systematic Mapping Study.” In Lecture Notes in Networks and Systems – Micro-Electronics and Telecommunication Engineering, Springer, 2023, pp. 655–665.
[18] Ahmed, S. R. (2007). Secure software development: Identification of security activities and their integration in software development lifecycle.
[19] Guo Ming, Zhang Lin. "Review of version tracking and determination technologies for open source components of package-free management files." Computer Application Research, 2019, 36(11): 3218-3225.
[20] Raj, G., Singh, D., & Bansal, A. (2014, September). Analysis for security implementation in SDLC. In 2014, 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence) (pp. 221-226). IEEE.
[21] Peng Zhang, and Haiyan Liu. "Summary of open source components for package-free management files." Computer Engineering, 2014, 40(8): 136-140.
[22] Kao, T. C., Mao, C. H., Chang, C. Y., & Chang, K. C. (2012, June). Cloud SSDLC: Cloud security governance deployment framework in secure system development life cycle. In 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (pp. 1143-1148). IEEE.
[23] Thirunagalingam, A. (2022). Enhancing Data Governance Through Explainable AI: Bridging Transparency and Automation. Available at SSRN 5047713.
[24] Pappula, K. K., & Rusum, G. P. (2020). Custom CAD Plugin Architecture for Enforcing Industry-Specific Design Standards. *International Journal of AI, BigData, Computational and Management Studies*, *1*(4), 19-28. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V1I4P103
[25] Rusum, G. P., Pappula, K. K., & Anasuri, S. (2020). Constraint Solving at Scale: Optimizing Performance in Complex Parametric Assemblies. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(2), 47-55. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I2P106
[26] Rahul, N. (2020). Optimizing Claims Reserves and Payments with AI: Predictive Models for Financial Accuracy. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(3), 46-55. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I3P106
[27] Enjam, G. R., & Tekale, K. M. (2020). Transitioning from Monolith to Microservices in Policy Administration. *International Journal of Emerging Research in Engineering and Technology*, *1*(3), 45-52. https://doi.org/10.63282/3050-922X.IJERETV1I3P106
[28] Pappula, K. K., & Rusum, G. P. (2021). Designing Developer-Centric Internal APIs for Rapid Full-Stack Development. *International Journal of AI, BigData, Computational and Management Studies*, *2*(4), 80-88. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I4P108
[29] Pedda Muntala, P. S. R., & Jangam, S. K. (2021). End-to-End Hyperautomation with Oracle ERP and Oracle Integration Cloud. *International Journal of Emerging Research in Engineering and Technology*, *2*(4), 59-67. https://doi.org/10.63282/3050-922X.IJERET-V2I4P107
[30] Rahul, N. (2021). Strengthening Fraud Prevention with AI in P&C Insurance: Enhancing Cyber Resilience. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(1), 43-53. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I1P106
[31] Enjam, G. R., & Chandragowda, S. C. (2021). RESTful API Design for Modular Insurance Platforms. *International Journal of Emerging Research in Engineering and Technology*, *2*(3), 71-78. https://doi.org/10.63282/3050-922X.IJERET-V2I3P108
[32] Rusum, G. P., & Pappula, kiran K. . (2022). Event-Driven Architecture Patterns for Real-Time, Reactive Systems. *International Journal of Emerging Research in Engineering and Technology*, *3*(3), 108-116. https://doi.org/10.63282/3050-922X.IJERET-V3I3P111
[33] Pappula, K. K. (2022). Containerized Zero-Downtime Deployments in Full-Stack Systems. International Journal of AI, BigData, Computational and Management Studies, 3(4), 60-69. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I4P107
[34] Jangam, S. K., & Karri, N. (2022). Potential of AI and ML to Enhance Error Detection, Prediction, and Automated Remediation in Batch Processing. *International Journal of AI, BigData, Computational and Management Studies*, *3*(4), 70-81. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I4P108
[35] Pedda Muntala, P. S. R. (2022). Natural Language Querying in Oracle Fusion Analytics: A Step toward Conversational BI. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(3), 81-89. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I3P109
[36] Rahul, N. (2022). Optimizing Rating Engines through AI and Machine Learning: Revolutionizing Pricing Precision. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(3), 93-101. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I3P110
[37] Enjam, G. R. (2022). Secure Data Masking Strategies for Cloud-Native Insurance Systems. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(2), 87-94. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I2P109

[38] Rusum, G. P., & Anasuri, S. (2023). Synthetic Test Data Generation Using Generative Models. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(4), 96-108. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I4P111

[39] Pappula, K. K. (2023). Edge-Deployed Computer Vision for Real-Time Defect Detection. *International Journal of AI, BigData, Computational and Management Studies*, *4*(3), 72-81. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P108

[40] Jangam, S. K. (2023). Data Architecture Models for Enterprise Applications and Their Implications for Data Integration and Analytics. International Journal of Emerging Trends in Computer Science and Information Technology, 4(3), 91-100. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I3P110

[41] Pedda Muntala, P. S. R., & Karri, N. (2023). Managing Machine Learning Lifecycle in Oracle Cloud Infrastructure for ERP-Related Use Cases. *International Journal of Emerging Research in Engineering and Technology*, *4*(3), 87-97. https://doi.org/10.63282/3050-922X.IJERET-V4I3P110

[42] Rahul, N. (2023). Transforming Underwriting with AI: Evolving Risk Assessment and Policy Pricing in P&C Insurance. *International Journal of AI, BigData, Computational and Management Studies*, *4*(3), 92-101. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P110

[43] Enjam, G. R., Tekale, K. M., & Chandragowda, S. C. (2023). Zero-Downtime CI/CD Production Deployments for Insurance SaaS Using Blue/Green Deployments. *International Journal of Emerging Research in Engineering and Technology*, *4*(3), 98-106. https://doi.org/10.63282/3050-922X.IJERET-V4I3P111

[44] Pappula, K. K., & Anasuri, S. (2020). A Domain-Specific Language for Automating Feature-Based Part Creation in Parametric CAD. International Journal of Emerging Research in Engineering and Technology, 1(3), 35-44. https://doi.org/10.63282/3050-922X.IJERET-V1I3P105

[45] Rahul, N. (2020). Vehicle and Property Loss Assessment with AI: Automating Damage Estimations in Claims. *International Journal of Emerging Research in Engineering and Technology*, *1*(4), 38-46. https://doi.org/10.63282/3050-922X.IJERET-V1I4P105

[46] Enjam, G. R. (2020). Ransomware Resilience and Recovery Planning for Insurance Infrastructure. *International Journal of AI, BigData, Computational and Management Studies*, *1*(4), 29-37. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V1I4P104

[47] Pappula, K. K., Anasuri, S., & Rusum, G. P. (2021). Building Observability into Full-Stack Systems: Metrics That Matter. *International Journal of Emerging Research in Engineering and Technology*, *2*(4), 48-58. https://doi.org/10.63282/3050-922X.IJERET-V2I4P106

[48] Pedda Muntala, P. S. R., & Karri, N. (2021). Leveraging Oracle Fusion ERP's Embedded AI for Predictive Financial Forecasting. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(3), 74-82. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I3P108

[49] Rahul, N. (2021). AI-Enhanced API Integrations: Advancing Guidewire Ecosystems with Real-Time Data. *International Journal of Emerging Research in Engineering and Technology*, *2*(1), 57-66. https://doi.org/10.63282/3050-922X.IJERET-V2I1P107

[50] Enjam, G. R. (2021). Data Privacy & Encryption Practices in Cloud-Based Guidewire Deployments. *International Journal of AI, BigData, Computational and Management Studies*, *2*(3), 64-73. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I3P108

[51] Rusum, G. P. (2022). WebAssembly across Platforms: Running Native Apps in the Browser, Cloud, and Edge. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(1), 107-115. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I1P112

[52] Pappula, K. K. (2022). Architectural Evolution: Transitioning from Monoliths to Service-Oriented Systems. *International Journal of Emerging Research in Engineering and Technology*, *3*(4), 53-62. https://doi.org/10.63282/3050-922X.IJERET-V3I4P107

[53] Jangam, S. K. (2022). Self-Healing Autonomous Software Code Development. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 42-52. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P105

[54] Pedda Muntala, P. S. R. (2022). Anomaly Detection in Expense Management using Oracle AI Services. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(1), 87-94. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P109

[55] Rahul, N. (2022). Automating Claims, Policy, and Billing with AI in Guidewire: Streamlining Insurance Operations. *International Journal of Emerging Research in Engineering and Technology*, *3*(4), 75-83. https://doi.org/10.63282/3050-922X.IJERET-V3I4P109

[56] Enjam, G. R. (2022). Energy-Efficient Load Balancing in Distributed Insurance Systems Using AI-Optimized Switching Techniques. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(4), 68-76. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I4P108

[57] Rusum, G. P., & Anasuri, S. (2023). Composable Enterprise Architecture: A New Paradigm for Modular Software Design. *International Journal of Emerging Research in Engineering and Technology*, *4*(1), 99-111. https://doi.org/10.63282/3050-922X.IJERET-V4I1P111

[58] Pappula, K. K. (2023). Reinforcement Learning for Intelligent Batching in Production Pipelines. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(4), 76-86. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I4P109

[59] Jangam, S. K., & Pedda Muntala, P. S. R. (2023). Challenges and Solutions for Managing Errors in Distributed Batch Processing Systems and Data Pipelines. *International Journal of Emerging Research in Engineering and Technology*, *4*(4), 65-79. https://doi.org/10.63282/3050-922X.IJERET-V4I4P107

[60] Pedda Muntala, P. S. R., & Karri, N. (2023). Leveraging Oracle Digital Assistant (ODA) to Automate ERP Transactions and Improve User Productivity. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(4), 97-104. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I4P111

[61] Rahul, N. (2023). Personalizing Policies with AI: Improving Customer Experience and Risk Assessment. International Journal of Emerging Trends in Computer Science and Information Technology, 4(1), 85-94. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P110

[62] Enjam, G. R. (2023). Modernizing Legacy Insurance Systems with Microservices on Guidewire Cloud Platform. *International Journal of Emerging Research in Engineering and Technology*, *4*(4), 90-100. https://doi.org/10.63282/3050-922X.IJERET-V4I4P109

[63] Pappula, K. K. (2021). Modern CI/CD in Full-Stack Environments: Lessons from Source Control Migrations. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(4), 51-59. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I4P106

[64] Pedda Muntala, P. S. R. (2021). Integrating AI with Oracle Fusion ERP for Autonomous Financial Close. *International Journal of AI, BigData, Computational and Management Studies*, *2*(2), 76-86. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I2P109

[65] Rusum, G. P. (2022). Security-as-Code: Embedding Policy-Driven Security in CI/CD Workflows. *International Journal of AI, BigData, Computational and Management Studies*, *3*(2), 81-88. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I2P108

[66] Jangam, S. K., Karri, N., & Pedda Muntala, P. S. R. (2022). Advanced API Security Techniques and Service Management. *International Journal of Emerging Research in Engineering and Technology*, *3*(4), 63-74. https://doi.org/10.63282/3050-922X.IJERET-V3I4P108

[67] Pedda Muntala, P. S. R., & Karri, N. (2022). Using Oracle Fusion Analytics Warehouse (FAW) and ML to Improve KPI Visibility and Business Outcomes. International Journal of AI, BigData, Computational and Management Studies, *3*(1), 79-88. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I1P109

[68] Rusum, G. P. (2023). Large Language Models in IDEs: Context-Aware Coding, Refactoring, and Documentation. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(2), 101-110. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I2P110

[69] Jangam, S. K. (2023). Importance of Encrypting Data in Transit and at Rest Using TLS and Other Security Protocols and API Security Best Practices. *International Journal of AI, BigData, Computational and Management Studies*, *4*(3), 82-91. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P109

[70] Pedda Muntala, P. S. R. (2023). AI-Powered Chatbots and Digital Assistants in Oracle Fusion Applications. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(3), 101-111. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I3P111

[71] Enjam, G. R. (2023). Optimizing PostgreSQL for High-Volume Insurance Transactions & Secure Backup and Restore Strategies for Databases. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 104-111. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P112