LEX LOCALIS-JOURNAL OF LOCAL SELF-GOVERNMENT
ISSN:1581-5374 E-ISSN:1855-363X
VOL. 23, NO. S6(2025)

GENERATIVE AI FOR SECURE RELEASE ENGINEERING IN GLOBAL PAYMENT NETWORK

Avinash Reddy Segireddy[1]

[1]*Lead DevOps Engineer*
ORCID ID : 0009-0002-9912-0629

***Abstract*—**Generative AI offers great promise in secure release engineering for global payment networks but introduces major risks that must be managed. The fastest and safest way to embed release security into Generative AI-assisted development pro- cesses is through structured prompting that generates safe con- figurations, compliance-aware implementations, and AI-specific failure modes; constraints that enforce safe and secure code; and support for model integration within release pipelines that can be independently audited, automatically tested, and operationally assured. Generative AI-backed secret management, access con- trol, and secret rotation can enhance security but introduce fresh threats that need careful mitigation. Security should be baked into every AI-assisted release engineering process, not added as an afterthought. Examples from payments illustrate these issues, incorporating lessons learned from incidents involving Generative AI assistance.

***Index Terms*—**Generative AI, secure DevOps, CI/CD, PCI DSS, threat modeling, data leakage, secrets management,Generative Artificial Intelligence (Generative AI),Secure Release Engineer- ing,Autonomous DevSecOps Pipelines,AIOps for Financial Sys- tems,Payment Network Security,Compliance-Aware CI/CD,AI- Augmented Software Assurance,Resilient FinTech Infrastruc- ture,Intelligent Risk Mitigation in Software Delivery,Trustworthy AI in Financial Services.

I. INTRODUCTION

The global payment network landscape is characterised by a diverse collection of interconnected payment solu- tions—payment rails, payment gateways, payment proces- sors—across multiple countries and regions. The orchestration of these components satisfies the needs of billions of users, involving many thousands of releases every year. Security, compliance, and risk are therefore paramount, especially with regard to any elevated responsibilities for facilitation or cus- todianship. Generative AI would be a transformative addition for secure release engineering, but it could create new security holes, particularly in data leakage. Secure DevOps practices in continuous integration–continuous delivery are blamed for many of the most serious breaches; their inherent risk is now compounded by reliance on AI. An overview shows how the capabilities of generative AI can be harnessed for good, not evil. The European Commission acknowledges that Bold and daring as the vision may be, regulatory and supervisory frame- works must guarantee the reliability, safety, and compliance of AI systems for their users. At a minimum, AI use in banks must be subject to the same risk management standards in place today for other categories of banks' exposures, including integrated firewalls to separate generative AI systems from customer data that could be misused. Payment services are wrapped in additional layers of rules and regulations that impose data protection measures such as encryption, masking, filtering, access control, secure storage, secure release engi- neering, and incident management.

Fig. 1. Generative AI in Payments

A. *Overview of Global Payment Networks*

Supporting all card-based transactions, global card-based payment networks comprise the key clearing-and-settlement services (payment rails), gateways that link merchants and their acquirers to the payments infrastructure (payment gate- ways), and processors that ensure transaction controls and data handling (payment processors). A simplified transaction example illustrates these roles. A customer making a purchase at a merchant using a card issued by an issuer belonging to the USD global network has the following experience. The card is presented to the merchant, who submits the transaction for authorization via their acquirer to the network switch. The issuer receives the authorization request, checks that the card is in good standing, as well as the account balance/currency (if applicable), and either Authorizes or Declines the transaction. If an Authorize response is returned from the issuer, the user presents their card for payment; contacted after the fact, the issuer then completes the accounting (e.g., debiting funds). At this stage of the process, the payment gateway sets up a Card Not Present (CNP) transaction and calls the payment network switch; in turn, the network switch routes the transaction to the issuer. The issuer verifies the authorization against fraud prevention requests and returns an Authorize or Decline to the payment network switch. If Auth, the issuer debit the cardholder's selected funding source.

B. *Security, Compliance, and Risk in Release Engineering*

The operational and architectural requirements that regula- tory regulators are putting on global payment networks are clearly motivating investment in governance and compliance, an investment that has in turn resulted in the formalization of the Secure Software Development Lifecycle (SSDLC), a high-level threat model and associated design patterns. Sev- eral recent incidents have however also highlighted the risk of introducing artificial intelligence influences into release engineering workflows without appropriate controls being in place. Cyber Kill Chain and MITRE ATT&CK threat modeling of payment networks point to significant attack exposure in Cloud and DevOps actors and tactics. A focus on the CI/CD aspect of release engineering therefore highlights detection, control and architectural patterns, in particular within the pipeline execution and prompt generation areas, that can help mitigate the risk of data leakage, accelerate incident response and ensure such influences are secure by design. A review of the shortest paths and spiders through the CI/CD kill chain and MITRE ATT&CK traversals identifies key areas for emphasis, development and operational enablement. Control actions associated with the relevant actors in the pipelines include Integrate, Transfer, Detect, Control, Monitor, Protect and Analyze, whilst the Kill Chain phases encompass Recon- naissance, Exploit, Install, Command & Control and Execute. Mistakes made by payment organizations are therefore being replicated, or accentuated, in the use of GenAI tools and services by the public at large and by criminals, with growing concerns about data leakage and privacy impacts (mis)using such tools, highlighting the need for similar considerations and controls in DevOps.

II. Foundations of Secure Release Engineering

Core concepts that underpin secure releases illuminate the perils of AI-driven tooling and processes while providing guidance for establishing adequate safeguards. Strongly regulated industries such as payments, where contractual obligations require compliance with PCI DSS, PSD2 and local legislation, mandate release engineering practices capable of resisting tomorrow's threats, as invisible weaknesses can be magnified by the next incident. A functional release pipeline consists of six stages—code, build, test, release, deploy and monitor—across which software changes are validated and made available to users. In payment environments, stringent security controls must be integrated into each stage of the pipeline to protect the sensitive data handled and ensure that security-related functions such as key rotation, data masking, data minimisation and safe data-sharing are correctly implemented. Threat modelling identifies and prioritises the high-value attacker goals, primary attack surfaces and data flows that should be considered when performing security verification on code artefacts and workflows. These security gates

complement the security verifications catalogued in Section 4.1. By addressing all these considerations, it becomes possible to accurately define the threat scenarios that can inform incident response planning and tooling. Four generic Generative AI capabilities have particular bearing on security in release engineering: prompting, code synthesis, software-based safety constraints and secured release pipeline integration. The potential of these capabilities is intrinsically limited by the governance and risk factors previously outlined, and the quality of their application is determined by the thoughtfulness of the prompts. The following sections explore how specific prompting techniques can elicit response decision-making that respects the security, compliance and risk concerns detailed earlier.

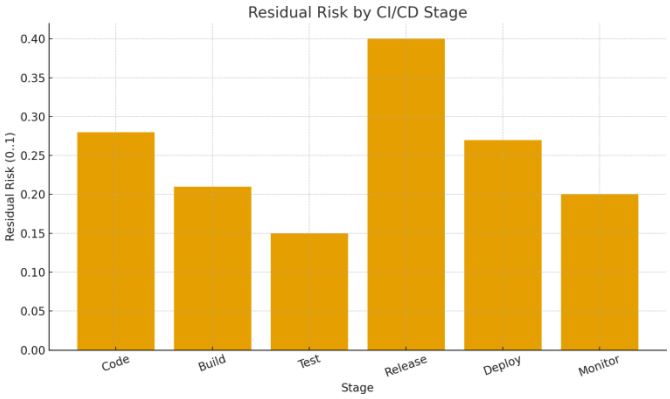


Fig. 2. Residual Risk by CI/CD Stage

TABLE I
RESIDUAL RISK PER CI/CD STAGE

Stage	Inherent Risk	Control Effectiveness	Residual Risk
Code	0.7	0.6	0.28
Build	0.6	0.65	0.21
Test	0.5	0.7	0.15
Release	0.8	0.5	0.4
Deploy	0.6	0.55	0.27
Monitor	0.5	0.6	0.2

Equation 01: Data-leakage probability for GenAI usage (Data-leakage probability across controls)

$$pleak = psens(1 - m)(1 - s)pretainproute \quad (1)$$

A. Release Pipeline Architecture

The release pipeline is a model for understanding and visually communicating the multiple stages involved in taking code changes from development through testing and into production. The typical stages are code, build, test, release, deploy, and monitor. Security is inherently of paramount importance for production releases, and the pipeline model serves as a useful guide to the security checks and controls that are traditionally integrated into such releases. The release architecture applicable to any given CI/CD process should carefully document all security gates and verifications, in- cluding where, when, and how they are performed. Having a clear picture of such safeguarding will ultimately assist when the processes are being planned with generative AI. The threat model for a release pipeline therefore identifies architectural elements that are of direct relevance to security, integrated, and risk considerations. These elements serve as an aid to understanding how a typical payment release may be exploited from within or outside the organisation. Doing so allows the payment industry-specific insights into release- security threats to be mapped back to realistic attack scenarios and detection- , containment- , and recovery-related playbooks. Assuredly, integrating where possible generative AI does not remove these critical security checks but can have positive implications through developers making fewer mistakes and careful AI prompting and code testing covering security holes when releases are ensured.

B. Threat Modeling and Attack Surfaces

Attackers seek money or operational disruption. They may expose secret keys and certificates, thereby compromising transaction integrity and authenticity, or take control of GPoPs, masquerading as legitimate payment networks and siphoning off funds. The main data flow travels from merchants, through payment gateways and processors, to card issuers. Payments can be located anywhere in the world: Europe, North America, Asia, Oceania, Africa, or any combination thereof. Therefore, data might be subject to EU GDPR, UK DPA, PCI DSS, and/or similar regulations. Agents within payment networks should use production machines that comply with least-privilege principles. Moreover, environment secrets (API keys, database passwords, encryption keys) should be stored in vaults and have limited lifetimes. The following sets of assets have the highest business value and should be properly protected. Business secrets that authorize card issuance and payments: issuing banks, acquirers, payment service providers, sensitive GPoPs, corporate buyers of goods or services, cardholder data (payment card number, expiration date, service code, cardholder name). Security secrets such as CA private keys, bank signing keys, certificates, private keys of sensitive GPoPs conducting interbank transactions, card brands' private keys. Transaction integrity secrets that enable redacted version of cardholder data: merchant MCC, production GPoP secret keys, sensitive GPoP credentials for acting as issuer or acquirer for non-existing BP, transaction type (normal, reversal or refund), transaction amount, currency code, ISO 4217 code. Transaction authenticity secrets and controls that enable funds transfer without being abused: sensitive GPoPs, card brands, participating networks for normal PCI DSS transaction type in non-prod environments.

III. Generative AI Techniques and Tooling

Generative AI introduces diverse capabilities that can en- hance the security of software release processes. However, warnings about hallucination and associated reliability con- cerns urge caution. The effects of such issues depend on the situational context where ML systems are applied, and a generous understanding of safety allows greater practical applicability. Securing DevOps—especially CI/CD pipelines fueling production services—adds another layer of risk to those applying AI systems. AI

inclusion raises rapid devel- opment and automated operations. Security and risk controls are paramount at these levels, integrating traditional ICIO CNCID and security gate coverage within secure AI De- vOps architecture governed by secure-by-design principles. Examples of general-purpose prompting that assist secure DevOps amplify obstacles in these pipelines, encouraging ap- propriate safety validation. Generative synthesis of code must incorporate safety checks, addressing sensitive data handling, provenance tracking, effective validation, execution and results verification, and secrets exposure and management. Scopes of AI model integration within CI/CD products, access treatment and secrets utilization contribute to foundational architecture patterns shaping secure AI GenAI DevOps.

Fig. 3. Generative AI Techniques and Tooling

A. Prompting Strategies for Secure DevOps

Attention and intent in prompts strongly influence the nature and behavior of Generative AI responses. To align outcomes with security and risk goals, prompts should: - Request a safe default configuration. - Elicit responses that consider security and regulation throughout the incident's life cycle. - Explore failure modes and potential for abuse of the response capability while polishing and demonstrating the model. To ensure that privileged information does not leak, prompting should reflect security principles that govern code synthesis. The following general techniques can help in this regard: - Saying "no" again and again: Most Generative AI cannot generate anything unless requested. Prompting the model not to generate at times when it should not, preventing private information exposure, or even minimizing the overall infor- mation in the final production are rich techniques. - Input- output direction orientation: Generative AI is endowed with great sorting capacity. Input-only prompts should be designed to inhibit generation, whereas Output-only responses should encourage generation. - Kill maybe: Using maybe generally gives unexpected results. Therefore, any threat based on an expected behavior should explore the answer not being gen- erated, instead forcing quantitative restrictions on the result. The following examples demonstrate how this method can help with secure DevOps.

B. Code Synthesis with Safety Constraints

Four types of safety constraints augment a prompt's utility in a secure DevOps context: handling of secrets, mitigating security-critical input validation vulnerabilities, embedding provenance information into generated artifacts, and ensuring appropriate reproducibility characteristics. These constraints are relevant for generated code, scripts used within release pipelines, and commands issued to hosted services or LLMs. Management of secrets, such as passwords and API keys, is a critical component of secure software engineering and release workflows. It is essential to avoid any hard-coded secrets or leakage of vault service credentials in any generated code, and that secrets are kept encrypted. A prompt constraint in natural language might, therefore, specify that "code must manage secrets via a vault solution" or "do not hard-code secrets." A parallel reinforcement learning (RL) system could be trained to de-classify secrets in both generated and test outputs.

IV. SECURITY BY DESIGN IN AI-DRIVEN RELEASE WORKFLOWS

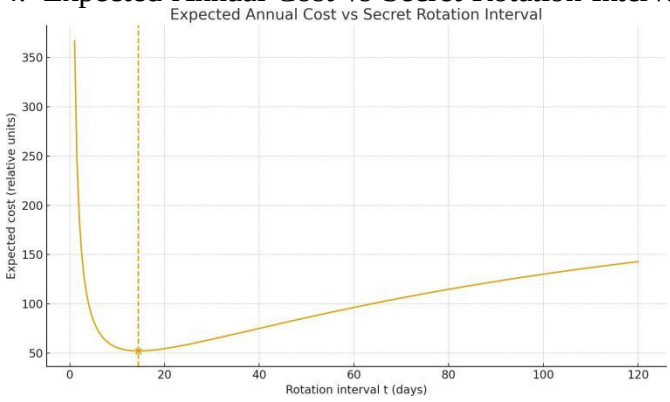
Security must be baked into automation processes, rather than included as an afterthought. Such embedding ensures that the outcome of an AI-influenced workflow remains free of security flaws, data leaks, or risks to personal privacy. Attention to the aspects of architecture integration discussed in Section 4.1 enables security considerations to be an integral part of every phase of release and deployment, rather than an extra layer tacked on at some later stage. Integrating AI tooling into CI/CD pipelines, especially within the earlier stages, opens additional risk avenues that must be addressed for the overall security framework to hold. New application or service code introduced to production reflecting AI influences is no different from traditional code in requiring safeguards against potential dangers such as mishandling sensitive information or providing attack surfaces for malicious entities; indeed, such attacks may be more likely because AI-generated code is unfamiliar. These considerations and more must therefore be factored deeply into code prompting and generation, with attention to secret handling, input validation, provenance support, and the principle of reproducibility.

Equation 02: Minimization (closed-form condition)

Differentiate and set to zero unique risks and demands verifiable deployments that address the specific challenges of each model category. Detecting and controlling these risks ensures that the model's safety and security characteristics remain intact. Four integration categories are defined: (a) models hosted by a third party (public or multi-tenant), (b) models obtained from a model repository and deployed on cloud or on-premise infrastructure,

(c) models packaged and distributed within products or as part of continuous delivery, and (d) custom models supporting internal tooling (e.g., prompt refining or syntactic sugar for domain-specific development).

Fig. 4. Expected Annual Cost vs Secret Rotation Interval



Control	PCI DSS Areas (indicative)
Secrets Management (vault, no hard-coding)	Req 3, 7, 8, 10
Data Masking & Minimisation	Req 3, 6, 12
Provenance & SBOM in Artifacts	Req 6.3, 10
Reproducible Builds	Req 6.4, 6.5
Access Control (least privilege)	Req 7, 8
Automated Security Testing in CI	Req 6.2, 6.5, 11

TABLE II

CONTROL-TO-COMPLIANCE COVERAGE

1) Access Control, Secrets Management, and Secret Rota- tion: Access control underpins all secure release processes. Enforcing least privilege and need-to-know principles miti- gates risk across the pipeline. User permissions for all tasks, especially those involved in model training, are strictly lim- ited. Automated open-source scanning tools regularly check builds against hard-coded secrets and credentials; successful operations of these tools are themselves audited and mon- itored. Servers and services such as vaults containing API codes are actively rotated, with either a break-glass policy or real-time alerts governing non-compliance. Centralized secret vaults with easy-to-use integration libraries and extensive audit

$$\frac{dC}{dt} = -\frac{c_r T}{t^2} + c_b \lambda e^{-\lambda t} = 0 \Rightarrow c_b \lambda e^{-\lambda t} = \frac{c_r T}{t^2}$$

No simple closed-form for t^*; solve numerically

keys, database passwords, TLS termination, and cloud access

A. Secure Model Integration in CI/CD (2)

Four integration patterns are presented for the introduction of AI models in CI/CD pipelines, along with the relevant risk controls and verifiable deployment procedures surrounding these models. The guidance directly supports the previous discussion on secure-by-design prompting and the application of release automation and generation techniques in Section 3.2. The integration of AI models into release pipelines brings capabilities support secret collection, storage, and controlled release across development and production systems. The vault maintains a proper audit trail, with various layers of access controls in place to rule out abuse. Wherever feasible, secrets are kept out of the code repository and automatically rotated whenever possible. Events are scheduled in advance to avoid failures and downtime. Proper visibility into the stacks that the vault is managing gives security teams the ability to detect patterns related to rotation intervals and usage. These not only help with secret audits but also protect against insider threats and negligence.

Fig. 5. Case Scenario PowerPoint Presentation

V. Case Studies and Threat Scenarios

Concrete examples demonstrate how generative AI plays a role in secure release engineering. Consider incident response in the banking industry, where an AI-assisted pipeline has been compromised. Playbooks detail the detection, contain- ment, and recovery steps, and—given the potential for data leakage—address other actors whose sensitive information was exposed. Further, payment releases typically involve sensitive data handling, including customer account details, which AI may inadvertently leak to subsequent users. Both AI usage and access governance, therefore, must take centre stage. Incident Response in Payment Release Pipelines: Payment processing is heavily regulated, providing no shortage of incentive to minimise risk. Nevertheless, there is no guarantee that a pay- ment release pipeline will remain uncompromised—especially with the continuous pressure to release faster. The banking sector thus adopts a ”plan for the worst” mindset, maintaining well-rehearsed playbooks for incident response that reflect all response phases: preparation, detection, assessment, contain- ment, eradication, recovery, and lessons learned. AI-assisted pipelines are no exception, with their own Special Interest Group for AI Attack Responses that develops detection, containment, and recovery steps. The ability to generate code can be useful during recovery and may also assist detection when, for example, AI tooling is abused to create distinct but similar detection patterns. Containment methods might take broader principles into account, including segregation of access controls at the same level.

A. Incident Response in Payment Release Pipelines

Incident response preparation enables effective triage of detected anomalies and supports rapid containment, recovery, and remediation efforts. Playbooks should suit the specific functions embedded into the release pipeline. Actions com- prise detection, containment, and recovery from incidents affecting code, build, test, and monitor stages; external or- chestration of external deployment and AI model provisioning routines; and interactions with externally managed services. Attention to incident response on pipelines utilized for AI- influenced releases addresses a common attack vector. Since changes detected by monitoring systems connected to the payment release pipeline may arise from normal operational use as well as from malicious actors, appropriate response actions must be driven by incident response playbooks. Incident response playbooks tailored to AI-assisted components can cover detection, containment, and recovery tasks between the regular operations and playbook development teams, such as test data reviews and unexpected data access monitoring. Playbooks should also oversee external interactions such as service credential validity checks and the invocation of exter- nal deployment and AI model provisioning processes, as well as current activity detection in externally managed services.

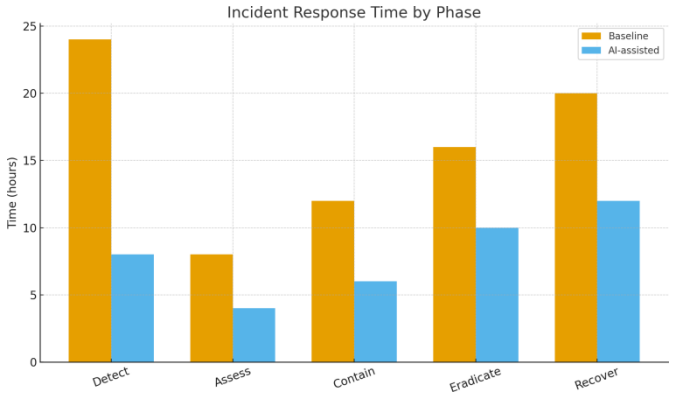
B. *Mitigating Data Leakage and Privacy Risks*

Emphasis on data privacy and protection extends beyond the need to prevent the unintentional release of secrets. Organizations must ensure that sensitive customer data are neither implicated in generative AI prompts and completions nor disclosed during work. Such scenarios present hazards in all workflows but highlight established controls in release engineering pipelines. By virtue of the highly regulated nature of payment services, organizations typically have contractual obligations for managing customer information — particularly production cardholder data — that affect all current use cases. Adequate safeguards during development and testing ensure that sensitive user data and transaction information are anonymized, masked, or otherwise rendered unrecognizable. Sampling sufficient volumes of representative data for testing purposes is a common control within PCI DSS. Frictionless and automatic data masking, together with obfuscation tech- niques that simultaneously eliminate identity while preserving data utility, facilitate the feeding of rich, anonymized data into developer, test, and R&D environments. Data-removal contracts are another suitable control for testing in dev and research areas.

VI. GOVERNANCE, COMPLIANCE, AND STANDARDS

PCI DSS, PSD2, and other regional regulations stipulate controls for the safe preparation of new software code and configuration changes, such as penetration testing, change failure analysis, or data masking to avoid pipeline data leaks. Payment institutions must extend these controls to AI-generated code since it executes in the computing environment of the institution and, like any other code, can introduce vulnerabilities if these controls are not respected. Additionally, releases are key stages for audits where the organizations are more exposed since any problem could leak data or money. As with any other corporate action performed by an employee, there has to be a governance process for these actions, providing an audit trail for the decisions made and creating resilience in functional continuity. Governance issues, especially the risk of using AI in ways that violate privacy regulations or not complying with the principles of safety, fairness, transparency, accountability, or explainability and ethical aspects such as racial bias or discrimination are also important. Even when there is no legal imposition, fairness and bias may be pertinent through the corporate vision that an institution wants to project. Other rules that assess those risks, provide recommendations, evaluation schemes, or tightening of regulations should be considered, especially those from supervisory organisms. To avoid problems, companies can use the recommendations of the EC White Paper on AI to implement trustworthy AI within their organizations.

Fig. 6. Incident Response Time by Phase



Phase	Baseline time (h)	AI-assisted time (h)	Improvement (%)	
Detect	24	8	66.66667	
Assess	8	4	50	
Contain	12	6	50	
Eradicate	16	10	37.5	
Recover	20	12	40	

TABLE III
INCIDENT RESPONSE TIME COMPARISON

Equation 03: Diminishing returns from masking

A simple conservative curve that reflects diminishing returns the CPE. Important areas of focus include controlled settings for training and fine-tuning data, ensuring model-generated content does not introduce undesirable gaps, or reducing the importance of CA control verifications. Additionally, access to both automatic and manual security testing tools is limited to users working to eliminate significant security vulnerabilities that may remain in the custom-built code. In agreements involving additional payment Transaction Data, such as for development assistance or AI/ML model training, Test Data Masking is specified to ensure data confidentiality and prevent the introduction of bias into the development, training, or preparation process.

B. AI Safety, Fairness, and Accountability in Finance

Generative AI could transform the way financial services, including payments, are delivered, exposing players to new op- portunities, markets, and clients. However, AI holds risks that could unleash unintended consequences and have a detrimental impact on individuals' the public's and businesses' well-being in society and economy, creating chaos and harm, and even undermining trust in AI or in finance. Consequently, every AI initiative should take into account potential harm that might result from the AI service to be deployed and, if apparent, such risks should be mitigated prior to widespread adoption. In this respect, secure models, trained with representationally suffi- cient populations and unveiled through explainable systems, would help in ensuring that Bank's values are represented as well as that the AI service is safeguarded, fairness-aware, and sufciently validated. Attacks, data leaks, and biases can cause considerable harm and facial recognition undisputable systems. Therefore, the AI services stay aligned with the credit institution's values, and, even if an impact assessment assess- ment is compliant playing with those detection can surface into unfairness, explainability, or a holding basis performance should function as controls. Such design patterns can soften first-level detection, and enable the arranged adjustment to if the risk mitigation process is followed overly, monitoring and continually adjusting of the system.

VII. CONCLUSION

Integration of Generative AI into production workflows

$LeakRisk(m) = (1-m)\alpha,\ \alpha > 1\alpha \approx 1.5--2$ works well to illuosfftreartse o'hrgiganeizarltyiowsinsp"otential benefits in speed and capability. However, embedding Generative AI into production systems, like payment release pipelines, represents a new attack surface. Adversaries can exploit any weaknesses to attack organizations or their customers—indeed, compromised release processes can be particularly dangerous because the

A. PCI DSS, PSD2, and Regional Regulations (3)

The evaluation of AI-enabled release engineering, as out- lined by PCI DSS with supporting guidance from PSD2, reflects secure practices for payments across all dimensions. Key components focus on human-and-model-generated con- tent in a safe manner. PCI DSS applies to any organization in the Card Payment Ecosystem (CPE), including Tools & Services Providers (TSPs), which develop, deploy, operate, or manage IT systems for any of the CPE entities that can affect the security of cardholder data and/or the security of attacker can insert malicious code within the organization's infrastructure and under its control. Organizations must there- fore design AI-influenced workflows with security in mind, actively looking for opportunities to build safeguards directly into their processes. Implementing the supporting activities and architectures described in this work can help make AI- influenced release processes secure by design. Furthermore, the use of structured prompting strategies can elicit AI outputs that comply with important external controls and internal mitigations—such as those related to access management and Secrets Management—or that highlight known failure modes.

A. Emerging Trends

Several emerging trends related to AI-assisted secure re- leases for global payments could be the focus of subsequent research or could impact policy and design decisions. Two key trends that may affect the safety of AI-augmented re- lease pipelines are the increasing interest in the alignment problem and concerns about possible biases influencing AI- assisted decisions and outputs. The deep-water horizon disas- ter—attributed in part to the failure of interrelated safety sys- tems—serves as a warning about the risks of becoming overly reliant on AI-assisted processes and the importance of main- taining diversified controls. Artificial-intelligence-influenced workflows require caution, and these insights aim to embed security considerations into the design. However, this caution should not extinguish the experimentation and exploration that have long supported safety in payments. Solutions that provide governance and auditability, support safe AI training and inference, produce satisfactory playbooks and incident examinations, and treat tampering and misuse—as well as the release of generated secrets—as major security concerns are essential. Explicitly binding the feedback loop and data pipeline, using extra AI systems for critical safety checks, and applying appropriate diversity checks during handling will limit the possibility of undetected bias impacting incident response and playbooks.

REFERENCES

[1] Szandała, T., et al. (2025). AIOps for Reliability: Evaluating Large Language Models with Chaos Experiments for Autonomous Root-Cause Analysis. Proceedings of ICCS 2025. ICCS
[2] (SSRN preprint) Agentic AI in Software Engineering: Applications and Implications Across the SDLC (2025). SSRN
[3] arXiv preprint) Agentic AI for Software (2025). arXiv
[4] NashTech. (2025). The Rise of Cognitive Pipelines: CI/CD Meets Au- tonomous Intelligence. Industry blog (useful for framing "autonomous CI/CD"). NashTech Blog
[5] Reuters. (2025, June 25). Gartner: Over 40