International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V6I3P111
Eureka Vision Publication | Volume 6, Issue 3, 61-70, 2025

Original Article

Securing Digital Transformation: A Framework for Mainframe and Cloud Ape Governance

Arun K Gangula
Independent Researcher.

Received On: 06/07/2025 ***Revised On: 27/07/2025*** ***Accepted On: 26/08/2025*** ***Published On: 23/09/2025***

Abstract - The implementation of APIs to connect legacy mainframe systems with modern cloud platforms creates paramount security and governance problems. Organizations that implement hybrid models for digital transformation purposes face growing complexity in cyber threats because of their increased exposure. The current security controls, which operate in separate silos, fail to provide sufficient protection for this environment. The paper establishes a complete four-layered hybrid API governance framework to handle these security challenges. The framework consists of four main components which include (1) Policy and Governance Foundation with Policy-as-Code (PaC) for centralized rule enforcement and (2) Unified Identity Fabric which merges cloud and on-premises Identity and Access Management (IAM) under Zero Trust principles and (3) Secure Development and Operations Lifecycle (DevSecOps) which incorporates security into CI/CD pipelines for mainframe and cloud artifacts and (4) Unified Observability and Response plane which provides real-time threat detection and correlation across platforms. The paper explains the necessary architecture, implementation methods, and technological requirements to achieve secure hybrid modernization.

Keywords - API Security, Mainframe Modernization, Hybrid Cloud, API Governance, DevSecOps, Zero Trust, Identity and Access Management (IAM), SIEM, Policy as Code (PaC), Regulatory Compliance.

1. Introduction

1.1. The Hybrid Enterprise Reality: Mainframe Persistence and Cloud Adoption

The modern Enterprise IT sector operates through a hybrid system that unites traditional mainframes with contemporary cloud infrastructure. The financial services sector, together with healthcare and government, maintains mainframes as essential infrastructure because they deliver reliable operations and secure processing of vital workloads. Digital transformation requires cloud adoption because it enables organizations to achieve market competitiveness through agility, innovation, and scalability [1]. The permanent hybrid state requires mainframe and cloud technologies to operate together in perfect harmony.

1.2. APIs: The Essential but Vulnerable Bridge

APIs function as the essential connection points that allow cloud-native and mobile applications to access mainframe business logic and data in this hybrid environment [2]. Research indicates that 93% of organizations treat APIs as fundamental operational components because they enable digital product development and service delivery beyond technical integration [3]. The primary entry point for attackers has shifted to APIs. Organizations that expose mainframe capabilities through APIs unintentionally create a vulnerable interface that compromises the protection of sensitive assets [4].

1.3. Problem Statement: The Governance Gap in Hybrid Security

API-based integration between cloud and mainframe systems produces a significant security and governance gap because these systems operate with opposing security models. Mainframe security operates with perimeter-based controls at the host level, but cloud security depends on distributed trust systems and identity-based access and short-lived infrastructure. The different security approaches between mainframe and cloud systems create operational and organizational challenges. Mainframe teams often remain unaware of contemporary API security threats, while cloud security teams attempt to enforce controls that do not match mainframe systems [5]. The skills gap worsens this problem because experienced mainframe specialists retire while newer developers lack knowledge about legacy languages, such as COBOL, and mainframe architectures [6].

The situation produces a modernization paradox. The most popular modernization pattern, which involves keeping mainframe systems intact while exposing them through APIs (encapsulation), presents the most significant security risk [6]. The business disruption reduction of encapsulation enables direct public interface access to the organization's most secure systems, thus bypassing perimeter-based security that has developed over decades. The conversion of CICS transactions into RESTful APIs generates a new security risk through web attacks, which mainframes were not designed to handle [4]. The governance vacuum emerges because mainframe security

teams and cloud security teams lack complete visibility and control, thus making misconfiguration and exploitation more likely.

2. The Strategic Context of Mainframe Modernization and Integration

2.1. An Analysis of Modernization Drivers

Mainframe environment modernization requires strategic pressure from multiple directions that surpasses basic cost reduction goals. The main drivers for modernization have shifted from cost reduction, benefits of licensing, and infrastructure savings to business agility and competitive differentiation. [7] Enterprises recognize that their dependable legacy systems hinder innovation. [1] Organizations pursue modernization because it enables them to innovate faster while maintaining technological parity with competitors. [7] The main driver for development velocity stems from the adoption of DevOps practices together with CI/CD pipelines. The deployment speeds of modernized API-based architectures reach two to three times faster than traditional methods, according to reported data. [1] The ability to rapidly deploy new products and services through agile workflows becomes essential for companies to adapt to changing market requirements.

Organizations face a severe talent shortage due to the decline in the number of experienced COBOL and mainframe systems programmers as their retired workforce ages. [6] Through modernization, organizations gain access to a larger talent pool, including developers who specialize in Java, Python, Node.js, and cloud-native development. [1] The process of modernization provides organizations with a dual benefit of upskilling the workforce and creating a more competitive workforce that can support essential operations. [7]

The main driving force stems from the need to access the vast amount of data stored on mainframe systems. The valuable customer, transactional, and workflow data accumulated in legacy systems for decades has remained inaccessible to users. The data stored on mainframes becomes accessible through API enablement during modernization, which enables real-time analytics and ML and AI applications for deeper strategic insights and data-driven decision-making. [7] The expenses associated with inaction now exceed the costs of modernization because organizations face rising downtime risks, increasing compliance and security liabilities, and integration challenges with modern applications. Industry surveys show this understanding through a report, which states that 71% of organizations intend to modernize or migrate away from mainframes during the upcoming period. [1]

2.2. Key Technologies and Integration Patterns

A hybrid coexistence strategy requires enabling technologies and architectural patterns to achieve successful implementation. The main tools for connecting mainframe systems to cloud environments stand as the primary components. The middleware solutions IBM z/OS Connect and Microsoft Host Integration Server (HIS) enable standard RESTful API calls from cloud applications to become native requests that mainframe programs (such as CICS transactions or IMS programs) can process. [9] These tools perform intricate data transformations (e.g., from JSON to COBOL copybook format) and protocol conversions to create an API facade for legacy systems.

2.3. Critical Review of Modernization Patterns for Hybrid Coexistence

The selection of appropriate modernization strategies determines future costs, risks, and organizational agility. The "7 Rs" framework establishes a complete classification system that includes Rehost, Re-platform, Refactor, Rearchitect, Replace, Retire, and Retain (or encapsulate). [6] The Gartner TIME framework enables organizations to develop application modernization roadmaps through strategic evaluation of these options. [8] Most large enterprises face insurmountable challenges when attempting to replace their core mainframe systems entirely because of high risks, substantial costs, and complex implementation processes. The extensive business logic within these systems exists in poorly documented form while remaining essential to operations. The most practical and widespread solution for modernization has shifted toward maintaining multiple systems simultaneously.

The most widely used approach among these is the Retain/Encapsulate pattern, which also goes by the name Extend/Augment. [6] The mainframe application core remains intact while modern APIs (RESTful services) encase its functions and data access procedures. The approach enables cloud-based applications to securely connect with the legacy system through APIs without needing immediate modifications to COBOL or PL/I code. [9] The API-first integration method extends the mainframe's operational period while making it a full participant in modern distributed systems. [8]

Table I provides a comparative analysis of these key modernization strategies to guide strategic decision-making.

Table 1. Comparative Analysis of Mainframe Modernization Strategies [1]

Strategy	Description	Key Tools/Tech	Benefits	Risks/Challenges	Best Use Case
Rehost (Lift-and-Shift)	Move applications "as-is" to cloud	Micro Focus, Heirloom, AWS	Fastest migration, low upfront cost,	Still a monolith; limited DevOps;	Quick ROI with minimal changes.

	IaaS using emulation.	M2, Azure Logic Apps	and minimal disruption.	retains technical debt.	
Re-platform	Upgrade some components (e.g., database) for PaaS benefits.	Cloud databases (AWS RDS, Azure SQL), Linux	Gains some cloud efficiencies; moderate complexity.	Compatibility issues; retains some technical debt.	Tactical cloud optimization without a complete rewrite.
Refactor	Convert legacy code to modern languages and improve its structure.	COBOL-to-Java tools, modern IDEs	Better maintainability, closing skills gap, and enabling DevOps.	High cost, extensive testing, and risk of errors.	Long-term agility for critical apps.
Rearchitect	Redesign into microservices; expose APIs.	Microservices frameworks, Docker, Kubernetes, API gateways	Maximum agility, scalability, and cloud native.	Highest complexity and cost; requires expertise.	Complete digital transformation focus.
Retain/Encapsulate	Keep the legacy system but expose functions via APIs.	z/OS Connect, Azure API Management	Low risk, fast integration, and preserves investment.	Hybrid complexity, security, and latency concerns.	Quick integration with digital channels.
Replace	Fully retire and adopt COTS or SaaS solutions.	SaaS, custom cloud-native apps	Eliminates technical debt; modern capabilities.	High cost, migration risks, and potential lock-in.	When a suitable commercial solution exists.

The adoption of containerization as a key pattern helps speed up development while reducing risks that stem from insufficient mainframe expertise. The combination of Red Hat OpenShift with IBM Z and Cloud Modernization Stack enables developers to build sandboxed z/OS development and test environments that operate in containers on standard x86 hardware. Cloud-native developers can access mainframe-like environments through self-service using Git and Jenkins tools, which decreases their need for mainframe-specific skills and hardware. [10]

The modernization challenge has received significant recent development through generative AI applications. The main challenge of any mainframe project stems from the absence of modern documentation for code systems that date back decades. Google's Gemini models have developed AI tools that analyze legacy COBOL codebases to automatically explain business logic and detect application dependencies while generating initial test cases. [11] Tools now assist developers in converting legacy code into the modern Java programming language through automated refactoring processes. The AI-based method solves the "black box" issue by giving developers essential knowledge to work safely with legacy systems, thus speeding up modernization projects and lowering their risks. [8]

3. The Hybrid Ape Attack Surface: A New Frontier for Threats

3.1. From Fortified Perimeters to Pervasive Endpoints

API connections between mainframes and cloud services establish a complete security model transformation. The mainframe security model operates as a fortress that maintains strong host-level access control managers (ACMs) such as RACF, ACF2, or Top Secret, and a secure network boundary. [12] The system grants access through restricted channels, which are thoroughly monitored. API integration shatters this model. The perimeter effectively dissolves, replaced by a distributed and pervasive collection of API endpoints that can be accessed from anywhere on the internet. The new reality establishes identity as the primary security boundary. [4] Each API request requires separate authentication and authorization because network location-based trust assumptions no longer apply. The transition exposes the mainframe core operations to new security threats that its original design did not anticipate.

The existing risk has increased because enterprise network architecture continues to have persistent weaknesses. Security assessments show that mainframe LPARs and general corporate environments are not adequately segregated in most cases. The absence of segmentation between corporate network segments allows an attacker to use a single compromised location to access the mainframe system. The continued use of insecure legacy protocols such as unencrypted FTP for data transfers remains a common and easily addressable gap that exposes credentials and data in transit. [5]

3.2. Emerging Threats: AI-Driven Attacks and Unsafe Consumption

The threat landscape shows continuous evolution through two major trends, which need immediate focus: artificial Intelligence functions as a weapon that brings both advantages

and disadvantages to the table. Security analytics powered by AI helps improve threat detection, but adversaries use AI to create complex polymorphic malware and automate vulnerability discovery and exploitation at an unprecedented speed and scale. [12] The primary threat to API governance emerges from autonomous AI agents, which now function as the primary API consumers. [4] These agents, which use Large Language Models (LLMs), operate to execute complex tasks through automated interactions with multiple tools and APIs. This emerging trend establishes a potent new security threat. An AI agent operating at machine speed and scale can become vulnerable to manipulation or hijacking, which enables devastating attacks against Unrestricted Resource Consumption (API4) or Unrestricted Access to Sensitive Business Flows (API6). [13] An airline reservation system running on a mainframe would face catastrophic failure when an agent designed for travel booking makes millions of API requests through deception.

API governance needs to undergo a complete transformation to handle Non-Human Identities (NHIs) because of this change. [4] The majority of Identity and Access Management (IAM) systems operate with a human-oriented approach. The modern governance framework requires strong procedures for registering, authenticating, authorizing, and auditing these AI agents. The development of specialized "AI Gateways" has become necessary. These gateways function as a protective barrier in front of traditional API gateways to enforce AI-specific policies, which include consumption-based rate limiting and protection against prompt injection attacks and governance rule enforcement for AI agent interactions.

3.3. Applying the OWASP API Security Top 10 (2023 to Mainframe-Cloud Integrations

The complete understanding of the threat landscape requires analysis through established API security risks. The OWASP API Security Top 10 project establishes a definitive community-driven list of the most critical vulnerabilities that affect APIs. [13] These risks gain new importance when applied to mainframe-cloud hybrid architecture because they create a pathway between modern web-facing vulnerabilities and catastrophic breaches of legacy systems of record. These vulnerabilities have increased potential impact because they enable attackers to access organizations' most valuable and sensitive data assets. [7] The standard OWASP risks receive specific high-impact scenario interpretations in Table II, which apply to hybrid environments.

The framework enables practical threat modeling and security control prioritization. BOLA (API1) and BFLA (API5) represent the most dangerous threats when used in hybrid environments. A BOLA vulnerability enables attackers to extract millions of mainframe database records through API call ID manipulation. A BFLA flaw enables a low-privilege user on a modern cloud application to execute powerful high-privilege administrative functions on the mainframe, which could result in complete system compromise.

Table 2. Owasp Api Risks In A Mainframe-Cloud Context [13]

OWASP Risk	Description	Hybrid Example	Key Mitigation
API1: BOLA	Missing object-level access checks.	Users iterate through account numbers to access others' data via a cloud API linked to the mainframe DB2.	Enforce fine-grained authorization at the API layer; use Policy-as-Code.
API2: Broken Authentication	Flawed Auth lets attackers impersonate users.	Forged JWT allows an attacker to access CICS transactions.	Use OAuth2/OIDC; enforce MFA; secure token handling.
API3: Property-Level Authorization	Overexposed or uncontrolled object properties.	API returns full RACF user record (incl. sensitive fields).	Return only necessary data; server-side property validation.
API4: Resource Consumption	No limits on resource use can lead to DoS or cost spikes.	Excessive API calls trigger mainframe batch jobs, consuming MIPS.	Apply rate limiting and quotas; monitor consumption.
API5: Function-Level Authorization (BFLA)	Lack of role checks for specific functions.	A regular user calls the admin endpoint to reset passwords on the mainframe.	Enforce RBAC; separate regular/admin functions clearly.
API6: Abuse of Business Flows	No controls on business action frequency.	Bots reserve all inventory via API, blocking real customers.	Use CAPTCHA for BOT detection and behavioral monitoring.
API7: SSRF	Server accepts user-controlled URLs for internal calls.	User supplies the internal mainframe URL to import a profile picture.	Restrict URLs via allow-list; sanitize input.
API9: Inventory Management	Unknown or unmanaged APIs remain exposed.	The deprecated API version remains connected to the mainframe, making it vulnerable to attack.	Maintain automated API inventory and lifecycle management.

4. Multi-Layered Framework for Hybrid Ape Governance

Enterprises need a unified governance framework to protect their hybrid mainframe-cloud attack surface adequately. The current complexity of hybrid environments makes it impossible to maintain separate security management for cloud and mainframe systems. The proposed framework consists of four interconnected layers, which provide unified governance throughout the entire API lifecycle, starting from development through production. The four security layers of Policy, Identity, Develops, and Observability work together to establish arobust security posture.

4.1. Layer 1: Policy and Governance Foundation

The fundamental operational standards for enterprise APIs are established through this foundational layer. The traditional governance approach, which relies on manual reviews and document-based policies, fails to meet the needs of agile and DevOps environments due to its slow and inconsistent operation [14]. This framework implements Policy-as-Code (PaC) automation for governance, which provides scalable and consistent results. The "API Council" or "Center for Enablement" functions as a centralized governance body to establish enterprise-wide standards that include API naming conventions and versioning strategies, resource structuring, and error handling formats. The council operates in a federated model, which enables teams to implement these standards independently.

Security and governance policies become machine-readable rules through PaC, which store them in version-controlled systems like Git using YAML or Rego syntax [15]. The policy demanding OAuth 2.0 with MFA for all PII-handling APIs gets transformed into enforceable code. Automated tools perform multiple stages of violation checks.

- IDE-level linting flags non-compliance during development.
- CI/CD pipelines prevent the deployment of non-compliant code.
- API gateways enforce policies at runtime [14].

The approach maintains uniform policy enforcement between cloud and mainframe APIs. Open Policy Agent (OPA) serves as a widely adopted open-source engine for implementing this model [15].

4.2. Layer 2: Unified Identity Fabric

API security depends on identity as its fundamental element, which determines both request origin and permitted actions. A hybrid environment requires Zero Trust implementation, which involves integrating cloud-native and mainframe identity systems. Cloud applications use Identity Providers (IDPs) like Microsoft Entra ID or Okta, leveraging open standards such as OAuth 2.0 and OIDC, issuing JWTs for authentication. Mainframes operate with External Security Managers (ESMs) such as RACF, ACF2, or Top Secret through proprietary communication protocols [5].

The Unified Identity Fabric operates as an abstraction layer that converts cloud identity information into mainframe-compatible formats [16]. The system integrates identity federation with reverse proxies, custom APIs, and middleware technologies [17]. The process involves:

- The API gateway accepts a JWT that comes from a cloud application.
- The fabric checks the JWT validity before accessing centralized policies (Layer 1).
- The fabric connects the cloud identity to a mainframe identity before accessing the ESM.
- The system generates a short-lived credential known as a RACF Pass Ticket for mainframe access.
- The mainframe executes transactions using security credentials that have the minimum required permissions.

The implementation of Zero Trust security across hybrid environments becomes unfeasible without identity mediation. The mainframe faces two main challenges because it lacks support for modern communication protocols, and poorly designed systems can introduce performance delays [17].

4.3. Layer 3: Secure Development and Operations Lifecycle (DevSecOps

The DevSecOps layer implements security within the Software Development Lifecycle (SDLC) by adopting the "shift left" strategy, which makes security a collective responsibility. The hybrid environment requires a single CI/CD pipeline that supports both mainframe and cloud components.

The build, test, and deployment automation for COBOL programs (mainframe) and containerized microservices (cloud) are performed by tools including Jenkins, GitLab CI, and Red Hat Ansible [18]. The primary automated security controls consist of:

- The source code analysis tools SonarQube, Checkmarx, and Codacy perform Static Application Security Testing (SAST) to detect vulnerabilities in COBOL, Java, and Python code during the code committing process [18].
- The SCA tool Snyk scans open-source libraries and dependencies to detect vulnerabilities, which ensure only secure components are used [18].
- The DAST tools OWASP ZAP and Acunetix use runtime probes in staging environments to identify active vulnerabilities in applications [18].
- The security tools Aqua Security and Trivy perform Docker image vulnerability scans before Kubernetes deployment [18].

Supply chain security stands as a vital component in this

process. The process requires artifact integrity verification and uses dynamic secrets management to prevent the use of long-lived credentials [19]. These controls integrated into the CI/CD pipeline make security an essential element of mainframe and cloud development processes.

4.4. Layer 4: Unified Observability and Response

The final layer delivers complete visibility across cloud and mainframe environments. Hybrid architectures produce security data fragmentation because mainframe SMF records and RACF logs exist separately from cloud provider logs (e.g., AWS CloudTrail), API gateway logs, and application logs. Advanced attacks remain invisible to teams because they view security data in isolated fragments. A cloud-native Security Information and Event Management (SIEM) system solves this problem by collecting, standardizing, and connecting security log data from both environments [20]. This enables cross-platform attack chain reconstruction. A single correlation rule can establish a connection between API gateway request IDs and mainframe CICS transaction IDs to generate unified security alerts with complete context information [20].

The detection capabilities receive improvement through User and Entity Behavior Analytics (UEBA) models, which establish typical patterns for users, service accounts, and API clients. Machine learning enables UEBA to detect abnormal behavior through its analysis of typical patterns, which include both API call irregularities and unexplained data access [20]. The unified approach to observability enables proactive threat hunting and rapid incident response, which are crucial for securing hybrid cloud-mainframe ecosystems.

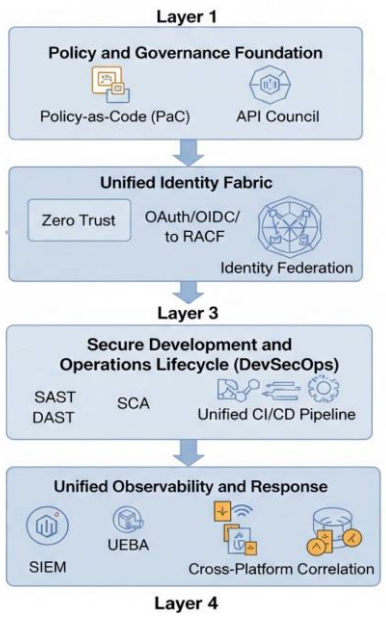

Figure 1. The Four-Layered Hybrid API Governance Framework

5. Framework Implementation and Operationalization

5.1. Phased Roadmap for Adoption

A governance framework of this magnitude needs a strategic, phased implementation approach. A "big bang" rollout is unrealistic and likely to fail due to the complexity and organizational change involved. A more pragmatic roadmap consists of four distinct phases, allowing an organization to build maturity incrementally, demonstrate value early, and adapt the framework to its specific needs.

- **Phase 1:** Assessment and Discovery. The journey starts by conducting a comprehensive evaluation of the present situation. A complete listing of all mainframe applications, together with their business importance and data connection forms part of this process. [8] The analysis of legacy codebases through automated tools reveals their logic structure and interconnectedness, while knowledge capture sessions with subject matter experts serve to document "tribal knowledge" before its disappearance. [6] The cross-functional API governance body or council forms during this phase to create the first set of policies and standards.
- **Phase 2:** Pilot Program. After defining the landscape, the following step involves choosing a single application with low risks that demonstrates meaningful value for framework implementation. The pilot process examines a complete end-to-end workflow through the integration of tool chains and validation of the CI/CD pipeline, as well as testing the identity fabric and initial SIEM correlation rules configuration. The team obtains practical experience by running this phase to enhance their procedures while showing concrete outcomes to stakeholders before undertaking extensive investment.
- **Phase 3:** Scaled Rollout. The framework implementation process advances through multiple stages that expand coverage to additional applications and services based on lessons learned from the pilot program. The expansion process requires prioritization according to both business value and risk factors. The governance policies, together with automation scripts, receive continuous improvement as new teams join the project. The fourth phase focuses on enhancing speed by implementing new processes and tools within the organization's standard operating procedures.
- **Phase 4:** Optimization and Automation. The implementation stage transitions into optimization and maturity as the focus of this final phase. The process involves automation improvements combined with AI and machine learning applications for enhanced threat detection and predictive analytics, and a self-service model for development teams. [14] The system should deliver a seamless experience that

enables developers to access governance and security platform services through APIs, which enables them to build secure applications quickly while following enterprise guidelines.

5.2. Toolchain Architecture: Selecting and Integrating Solutions

The four-layered framework development needs precise tool selection and integration of commercial and open-source solutions. The choice of tools depends on the organization's current technology infrastructure, financial capabilities, and personnel expertise. A reference architecture demonstrates how different components unite to achieve the necessary capabilities.

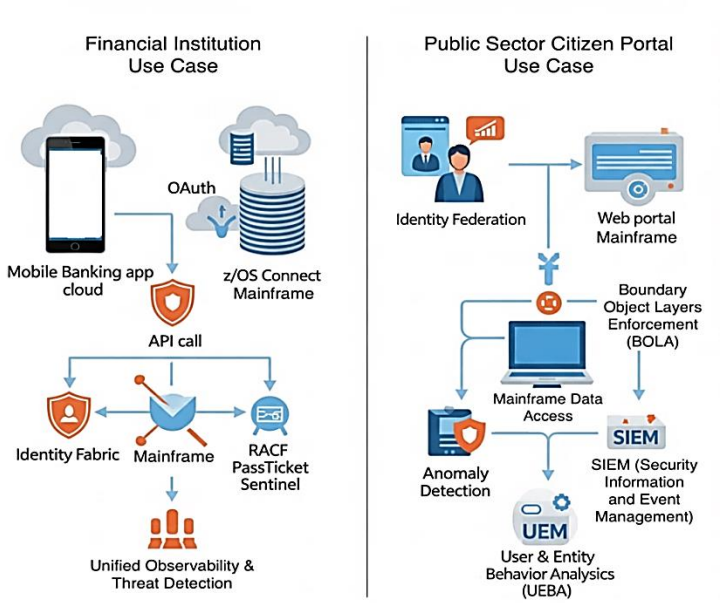

Figure 2. Hybrid API governance applied in financial and public sector use cases.

5.3. Illustrative Case Studies

The framework demonstrates its practical application through two synthesized case studies, which follow.

- **Case Study 1:** *Financial Institution Modernizing for Mobile Banking:* A central retail bank requires the development of a new cloud-native mobile banking application. The application required real-time access to customer account balances, transaction histories, and payment processing functions, all of which were located on their core banking system operating on an IBM z/OS mainframe. They adopted the Retain/Encapsulate strategy by using IBM z/OS Connect to transform CICS transactions into RESTful APIs for exposure.

5.4. Framework Application

- **Policy (Layer 1):** The bank's API Council established rigid design requirements for financial APIs through OPA policy creation, which mandated PCI-Scope APIs to need particular OAuth 2.0 scope authorization and complete transaction logging.
- **Identity (Layer 2):** Microsoft Entra ID served as the leading identity provider for mobile users through identity fabric implementation. The fabric validated the JWT during user login to produce a short-lived RACF Pass Ticket, which authorized the particular transaction on the mainframe while maintaining end-to-end identity propagation.
- **DevSecOps (Layer 3):** A unified GitLab CI/CD pipeline was created. The pipeline executed Checkmarx SAST analysis on both Swift code from the mobile application and exposed COBOL programs. Snyk performed security scans on all open-source libraries.
- **Observability (Layer 4):** Microsoft Sentinel received logs from Azure, the API gateway, and z/OS Connect and mainframe SMF records. The system used custom correlation rules to identify fraud patterns, which included users making fast account transfers between different geographic locations from multiple accounts.
- **Case Study:** *Public Sector Agency Creating a Citizen Portal:* The state government agency established an online system for citizens to access their personal records, which included tax history and benefits information stored in VSAM files from previous decades on the mainframe system.

The main priorities focused on protecting personal data while implementing detailed access restrictions.

5.5. Framework Application

- **Policy Layer 1:** The governance policies at Layer 1 implemented data minimization through API field-specific data returns, which prevented the disclosure of complete records. The system enforced versioning policies as a method to handle changes while maintaining operational integration with other government systems.
- **Identity Layer 2:** The identity federation solution enabled citizens to access the system through their established trusted digital identities. The identity fabric implemented BOLA enforcement to restrict citizens from accessing any records except their own by validating the identity token against the requested record ID.
- **DevSecOps (Layer 3):** The agency managed deployments through an automated pipeline system that used Ansible. The API code underwent vulnerability scanning followed by Trivy scanning of container images before AWS deployment.
- **Observability (Layer 4):** The SIEM system at Layer 4 monitored for any irregular data access patterns through its configuration. UEBA detected suspicious activities by monitoring when an account tried to access numerous records in a short timeframe, which might indicate compromised accounts or enumeration attacks.

6. Navigating the Regulatory Landscape

The implementation of a strong governance framework becomes essential because it helps organizations meet strict global regulations and improve security while generating auditable evidence of compliance. The transition of compliance from manual periodic checks to automated outcomes happens through development and operational integration.

6.1. PCI DSS 4 0 Compliance

PCI DSS 4.0 will become fully effective in 2024 because it places strong emphasis on API security, as payment ecosystems heavily rely on it [21]. The proposed framework addresses key requirements:

- **Secure SDLC (Req. 6):** The DevSecOps layer (Layer 3) implements automated SAST, DAST, and SCA scanning to integrate security directly into the CI/CD pipeline [21].
- **Access Control (Req. 7 & 8):** The Unified Identity Fabric (Layer 2) implements strong authentication through MFA-backed OAuth 2.0 and least-privilege authorization for API calls (Req. 7 & 8) [21].
- **Logging (Req. 10):** The Unified Observability layer (Layer 4) uses centralized logging to merge API gateway logs with mainframe and cloud logs into a SIEM system for audit trail purposes (Req. 10) [21].
- **Vulnerability Management (Req. 11):** The DevSecOps pipeline uses continuous DAST scanning to detect and fix vulnerabilities [21].

6.2. GDPR and CCPA Compliance

APIs, transferring personal data must meet GDPR and CCPA obligations:

- **GDPR:** The policies in Layer 1 enforce data minimization, purpose limitation, and security principles [22]. Secure APIs operationalize the Right to Access and Right to Erasure, with TLS 1.3 and strict access controls safeguarding data transfers [23].
- **CCPA:** The framework supports consumer rights, including the Right to Know, Delete, and Opt-Out. The policy and identity layers block data-sharing API calls when users exercise opt-out rights [24].

6.3. Preparing for the EU Data Act (2025):

The EU Data Act, which will become effective in September 2025, requires secure data sharing, particularly for IoT-generated data [25]. The framework delivers the required security features, governance structure, and auditability capabilities to fulfil these upcoming requirements.

Table 3. Compliance Mapping of Framework Controls

Requirement	Description	Framework Control(s)
PCI DSS 4.0 6.2.1	Secure development of custom software & APIs.	Layer 3: SAST, DAST, SCA in CI/CD pipeline.
PCI DSS 4.0 8.4.2	MFA for non-console CDE access.	Layer 2: MFA at the API gateway for backend systems.
PCI DSS 4.0 10.2.2	Log all actions by admins/root users.	Layer 4: SIEM with mainframe and cloud log integration.
GDPR Art. 25 & 32	Data protection by design & security of processing.	All Layers: PaC, Zero Trust Identity, DevSecOps, Monitoring.
GDPR Art. 15 & 17	Right to Access & Right to Erasure.	Layers 1 & 2: Policies & APIs with identity verification.
CCPA Opt-Out	Right to opt out of data sharing/sale.	Layers 1 & 2: Policy-based opt-out enforcement at API & identity layer.

7. Conclusion

API-based mainframe and cloud system integration has created a significant governance gap that traditional security models, designed for silos, cannot address. The paper established a complete framework that connects four essential domains to close this gap through Policy-as-Code (PaC) automation and Unified Identity Fabric. It integrated the DevSecOps lifecycle and the Unified Observability plane. This integrated automated framework enables enterprises to master secure integration within hybrid ecosystems for both innovation and critical information asset protection.

The framework's principles will gain essential importance because autonomous agentic AI systems will emerge as primary API service consumers in the future. [4] Security will transition from human access management to Non-Human Identity (NHI) governance because of this new paradigm. The security of AI-to-AI interactions requires advanced policy engines and dedicated AI gateways to protect these interactions. The framework's extensible Identity Fabric, together with automated Policy-as-Code, enables the necessary management of emerging security challenges, including the development of password-less authentication methods, which verify that a strong verifiable identity stands as the foundation of contemporary security.

References

[1] N. Das, "Why mainframe modernization no longer optional: A 2025 Strategic Imperative for CIOs and CXOs - IntErRAIT," *InterraIT*, Jun. 05, 2025. https://interrait.com/news-update/why-mainframe-modernization-no-longer-optional-a-2025-strategic-imperative-for-cios-and-cxos/

[2] M. Pacheco, "What is Mainframe Modernization & Why Does it Matter?," *TierPoint, LLC*, Jan. 29, 2025. https://www.tierpoint.com/blog/mainframe-modernization/

[3] Yilia Lin, "5 Best Practices for API governance in 2025 - API7.ai," *5 Best Practices for API Governance in 2025 - API7.ai*, Feb. 06, 2025. https://api7.ai/blog/api-governance-best-practices-2025

[4] Balaganski and M. Reinwarth, "Why API Security is the New Cybersecurity Imperative," *KuppingerCole*. Jul. 07, 2025. [Online]. Available: https://www.kuppingercole.com/watch/api-security-new-imperative

[5] P. Young and D. Bryan, "Mainframe state of the Platform: 2025 security assessment," *NetSPI*, Jun. 26, 2025. https://www.netspi.com/blog/executive-blog/mainframe-penetration-testing/mainframe-state-of-the-platform-2025-security-assessment/

[6] G. Navot, "Mainframe modernization solutions: A practical guide for 2025," *Medium*, Feb. 25, 2025. [Online]. Available: https://medium.com/@gilad_nvt/mainframe-modernization-solutions-a-practical-guide-for-2025-c9676b19f79c

[7] M. Flinders and I. Smalley, "Mainframe modernization," *IBM Think*, Feb. 09, 2024. https://www.ibm.com/think/topics/mainframe-modernization

[8] "Legacy Mainframe Modernization: A Complete Guide for 2025," *Quinnox*, Apr. 16, 2025. https://www.quinnox.com/blogs/legacy-mainframe-modernization/

[9] V. Pujar, "Enterprise Modernization: Unlocking Mainframe Capabilities via APIs with z/OS Connect," *Medium*, May 25, 2025. [Online]. Available: https://vikaspo.medium.com/enterprise-modernization-unlocking-mainframe-capabilities-via-apis-with-z-os-connect-bd99e88e64c3

[10] S. Steuart and S. Loomis, "Modernize Mainframe Applications for Hybrid Cloud with IBM and AWS | Amazon Web Services," *Amazon Web Services*, May 09, 2022. https://aws.amazon.com/blogs/apn/modernize-mainframe-applications-for-hybrid-cloud-with-ibm-and-aws/

[11] N. Mehta and D. Yahalom, "Accelerate mainframe modernization with Google Cloud AI," *Google Cloud Blog*, Apr. 04, 2025. https://cloud.google.com/blog/products/infrastructure-modernization/accelerate-mainframe-modernization-with-google-cloud-ai

[12] L. Wilson, "Mainframe security in 2025: Countering new threats, using AI, and getting the basics right," *Planet Mainframe*, Feb. 21, 2025. https://planetmainframe.com/2025/02/mainframe-security-in-2025-countering-new-threats-using-ai-and-getting-the-basics-right/

[13] "OWASP API Security Project | OWASP Foundation." https://owasp.org/www-project-api-security/

[14] TRGoodwill, "API Governance - API Central - Medium," *Medium*, Mar. 23, 2025. [Online]. Available: https://medium.com/api-center/api-governance-3be87aab17b4

[15] Z. Ghalleb, "What is Policy as Code?," *wiz.io*, Jul. 09, 2024. https://www.wiz.io/academy/policy-as-code

[16] M. Kuppinger, "Identity Fabric and Reference Architecture 2025: Future-Proofing your IAM," *KuppingerCole*. Jan. 15, 2025. [Online]. Available: https://www.kuppingercole.com/watch/future-proofing-your-iam

[17] A. Santhanam, "What Issues Arise Integrating IAM with Legacy Systems?," Jan. 07, 2025. https://www.infisign.ai/blog/issues-arise-integrating-iam-with-legacy-systems

[18] "15 DevSecOps Tools to know in 2025," *Codefresh*, Mar. 26, 2025. https://codefresh.io/learn/devsecops/15-devsecops-tools-to-know-in-2025/

[19] R. McCune, S. Art, and C. DePinto, "Key learnings from the 2025 State of DevSecOps study | Datadog," *Datadog*,

Apr. 23, 2025. https://www.datadoghq.com/blog/devsecops-2025-study-learnings/

[20] Gebremeskel, "SIEM for hybrid Environments: Essential for cloud & On-Prem," *TECKPATH | Managed IT Services | Business IT Support*, Feb. 14, 2025. https://teckpath.com/the-importance-of-siem-for-organizations-using-cloud-and-on-prem-infrastructure/

[21] Planet 9, Inc, "PCI DSS 4.0. Requirements for API Security - Planet 9 Inc.," *Planet 9 Inc*. https://planet9security.com/pci-dss-4-0-requirements-for-api-security/

[22] A. Bradshaw, "GDPR: Data Compliance Best Practices for 2025," *Alation*, Sep. 23, 2024. https://www.alation.com/blog/gdpr-data-compliance-best-practices-2025/

[23] Devtips, "GDPR-Compliant Hosting: best practices for developers in 2025," *Medium*, Apr. 12, 2025. [Online]. Available: https://medium.com/@dev_tips/gdpr-compliant-hosting-best-practices-for-developers-in-2025-253763a3a77d

[24] "California Consumer Privacy Act (CCPA)," *State of California - Department of Justice - Office of the Attorney General*, Mar. 13, 2024. https://oag.ca.gov/privacy/ccpa

[25] O. Vasylyk, "Data protection digest 16 Feb - 2 Mar 2025: Data Act to strengthen EU digital market, vigilance over US data transfers," *TechGDPR*, Mar. 04, 2025. https://techgdpr.com/blog/data-protection-digest-4032025-data-act-to-strengthen-eu-digital-market-vigilance-over-us-data-transfers/

[26] Singhal, S., Kothuru, S. K., Sethibathini, V. S. K., & Bammidi, T. R. (2024). ERP excellence a data governance approach to safeguarding financial transactions. Int. J. Manag. Educ. Sustain. Dev, 7(7), 1-18.

[27] L. N. R. Mudunuri, V. M. Aragani, and P. K. Maroju, "Enhancing Cybersecurity in Banking: Best Practices and Solutions for Securing the Digital Supply Chain," Journal of Computational Analysis and Applications, vol. 33, no. 8, pp. 929-936, Sep. 2024.