Sarcouncil Journal of Engineering and Computer Sciences

ISSN(Online): 2945-3585

Volume- 04| Issue- 08| 2025

Research Article **Received:** 01-07-2025| **Accepted:** 23-07-2025 | **Published:** 12-08-2025

The Hidden Cost of Poor API Governance in Large Organizations

Venugopal Reddy Depa

CRH Americas Materials Inc. USA

Abstract: Organizations increasingly rely on API-driven architectures to enable modular development and microservices communication, yet many fail to implement adequate governance frameworks to manage these critical interfaces. Poor API governance creates significant hidden costs that accumulate silently across enterprise systems, manifesting as redundant development efforts, fragmented documentation, security vulnerabilities, and regulatory compliance failures. Major data breaches at prominent technology companies demonstrate how ungoverned APIs become entry points for attackers, exposing sensitive customer information and resulting in substantial financial penalties. The proliferation of shadow APIs—undocumented interfaces created outside official development workflows—further compounds these risks by creating security blind spots that remain invisible to organizational oversight. Traditional governance models often prove too rigid for modern development environments, creating bottlenecks that slow innovation and frustrate development teams. This creates a paradox where governance intended to provide protection instead becomes an obstacle to progress. The solution lies in evolving governance from enforcement-based control to enablement-focused collaboration, incorporating automation tools that embed compliance into developer workflows while maintaining flexibility for innovation. Democratizing governance through developer participation in rule creation and leveraging AI-powered tooling for intelligent policy enforcement can transform governance from a barrier into a catalyst for secure, efficient API development across large-scale organizational environments.

Keywords: API governance, security vulnerabilities, developer experience, automation tooling, regulatory compliance.

INTRODUCTION

In 2024, Twilio's Authy service suffered a significant data breach when attackers exploited an unsecured API endpoint, gaining access to 33 million phone numbers associated with Authy MFA accounts (Trend Micro, 2024). The same year, a similar API vulnerability at Dell exposed the personal data, including names and addresses, of 49 million customers (Freehling, H. 2024). Despite increasing regulatory scrutiny, vulnerabilities like these persist across the industry, demonstrating a systemic industry challenge and underscoring the need for modern API governance.

Drawing from my experience as a MuleSoft developer, I've seen both ends of the governance spectrum: organizations with little to no API oversight and others where rigid policies stifled innovation. In both cases, I experienced operational bottlenecks, poor scalability, and increased friction between teams. When governance was missing, things slipped through the cracks, and when it was too rigid, progress stalled.

This article explores why governance matters in the first place, the potential hidden costs of poor API governance, and how to implement it in ways that serve modern development environments without becoming a hindrance.

Understanding Api Governance

API governance refers to the rules, standards, and processes that ensure APIs are secure, reliable, and aligned with organizational and regulatory requirements. API governance helps dev teams throughout the API lifecycle by enforcing best practices like authentication, rate limiting, and versioning.

Many modern development teams rely on modular development techniques where applications are built as collections of different microservices communicating through APIs. This API-first approach prioritizes APIs at the beginning of the software development process, enabling developers to deliver more powerful and resilient software in less time.

However, this also leads to widespread proliferation of APIs, making it increasingly difficult to ensure that each one is secure, scalable, and cost-effective.

In 2017, Equifax suffered one of the most significant data breaches in US history, costing an estimated $1.4 billion. It compromised sensitive information of approximately 148 million individuals due to an API vulnerability, which was patchable with available updates, but was delayed due to a lack of cohesive governance over API infrastructure (U.S. House of Representatives, Committee on Oversight and Government Reform,

***Corresponding Author:** Venugopal Reddy Depa

DOI- https://doi.org/10.5281/zenodo.16814166

2018). This incident shows how poor API governance can turn your API resources into liabilities.

Table 1: API Security Management: Framework Elements and Risk Consequences (U.S. House of Representatives, Committee on Oversight and Government Reform, 2018)

Governance Element	Implementation Focus	Risk of Poor Governance
Authentication & Security Standards	Enforce secure access controls, rate limiting, and vulnerability patching across all API endpoints	Unauthorized access leading to massive data breaches affecting millions of users and resulting in billion-dollar losses
API Lifecycle Management	Establish versioning protocols, documentation standards, and systematic deployment processes	Proliferation of ungoverned APIs creating security liabilities and operational inefficiencies across enterprise systems
Organizational Alignment	Ensure APIs meet regulatory requirements and business objectives through standardized processes	Lack of cohesive governance turning valuable API resources into organizational liabilities and compliance violations
Development Best Practices	Implement modular development techniques with microservices communication protocols and API-first approaches	Difficulty ensuring scalability, cost-effectiveness, and security across widespread API implementations
Monitoring & Maintenance	Continuous oversight of API performance, security updates, and infrastructure management	Delayed security patches and untracked vulnerabilities leaving systems exposed to preventable attacks

CONSEQUENCES OF INADEQUATE API GOVERNANCE

In August 2023, APImetrics published an analysis of over one billion API calls across 8,000 endpoints from 100 geographically diverse cloud data centers spanning Amazon, Azure, Google, and IBM. Their findings revealed that poor API performance is costing the industry an estimated $92 billion each year (O'Neill, D. 2023). The $92 billion figure finally put a number to what dev teams across the industry had long suspected but were unable to quantify.

Moreover, it also underscores my central argument: Poor API governance doesn't announce itself; it drains resources quietly. Consequently, accruing costs go unnoticed until there is a performance dip, a customer experience issue, or a regulatory compliance failure, but by then, it becomes virtually impossible to trace the root cause. In this section, we will explore some of the consequences and hidden costs of poor API governance that many organizations might be overlooking.

Redundant Development and Maintenance

API redundancies occur when multiple teams within an organization work towards creating the same or similar APIs unknowingly, due to the lack of a central governance system that flags duplication.

Because of poor API governance, dev teams are unable to find existing APIs, which results in redesigning them from scratch and duplicating functionality. These duplicate APIs then go through full QA, staging, and release phases, and require separate hosting, monitoring, and scaling.

As a duplicate API goes through all these phases, we get wasted developer time in solving problems that don't need solving, additional costs of duplicated testing and QA, double infrastructure cost, and long-term operation drag. According to APIwiz, 64% of organizations report redundant or conflicting APIs (APIWIZ) This slowly builds up a technical debt iceberg: an accumulation of hidden inefficiencies that appears harmless until it starts dragging down engineering velocity and inflating cloud and operational costs.

Fragmented Documentation and Knowledge Silos

During my time at MuleSoft, I've learned that fragmented documentation and knowledge silos aren't just a symptom of poor API governance; they are often the root cause. Good API governance is based on universal visibility, consistency, and discoverability, but fragmented documentation and knowledge silos consistently undermine those goals.

Fragmented documentation means the API documentation is incomplete, inconsistent, spread across multiple systems, outdated, or hard to find.

Publisher: SARC Publisher

Whereas knowledge silos refer to information that is locked away within a team or individual and not accessible across the organization, either due to team structure, culture, or the use of different tools. This kills cross-team coordination, forcing teams to work in isolation.

Let's take a hypothetical example to further understand how fragmented documentation and knowledge silos slowly hemorrhage valuable resources, quietly increasing engineering costs and killing velocity at scale. Imagine a backend team publishes an OpenAPI spec to GitHub but forgets to link it in the shared Confluence space. A frontend team needs the same functionality, but they have no way of knowing that it already exists, so they build a duplicate from scratch. This results in two versions of the same API, both slightly different, both undocumented in key ways, and both requiring long-term maintenance.

Now, multiply that by dozens of teams working simultaneously across large organizations, and you end up with thousands of redundant APIs, conflicting standards, and thousands of wasted engineering hours, all translating into significant financial losses. Over time, these inefficiencies pile up and introduce a wide range of new problems, including compliance risks, increased onboarding time, high cloud costs, and mounting operational overhead.

Security Blind Spots and Shadow Apis

Poor API governance leads to the proliferation of security blind spots. These are interfaces that are left undocumented, untracked, untested, and outside the visibility of the organization's security team. One of the most common blind spots that often goes unnoticed is Shadow APIs. These APIs are often created without following the official development workflow for speed and convenience. However, when not properly cataloged, secured, or decommissioned, they end up becoming long-term liabilities, potentially exposing sensitive data or serving as a convenient entry point for attackers.

In 2024, Escape, a leading authority in API threat research and discovery, uncovered 18,000 exposed API secrets, 41% of which were deemed highly critical with the potential to cause serious financial and reputational damage for organizations. This 2024 study was based on crawling 189.5 million URLs across one million of the most frequently visited websites. It also revealed that many of the exposed API keys belonged to financial services, and exposed Stripe payment tokens alone could have been used to siphon an estimated $20 million (Beaujard, S. & Charikova, A. 2024).

The leaks included AWS keys, GitHub tokens, RSA keys, and credentials from major tech companies across different sectors, including fintech, e-commerce, and education. What is important to note is that these leaks weren't caused due to a lack of security; the main reason was that these interfaces had become blind spots with no centralized inventory, lifecycle management, or security scanning, underscoring the hidden financial costs of API blind spots. Without robust governance, shadow APIs become security liabilities that organizations pay to run and secure without knowing they even exist. In other words, they are paying to secure what is making their systems insecure.

Table 2: Hidden Costs and Security Risks of Poor API Management in Enterprise Organizations (O'Neill, D. 2023; Beaujard, S. & Charikova, A. 2024)

Consequence Category	Root Causes and Characteristics	Financial and Operational Impact
Redundant Development and Maintenance	Multiple teams unknowingly create identical APIs due to lack of central governance system and inability to discover existing interfaces	Wasted developer time, duplicated QA and testing costs, double infrastructure expenses, long-term operational drag affecting engineering velocity
Fragmented Documentation and Knowledge Silos	Incomplete documentation spread across multiple systems, outdated information, team-specific tools creating isolated knowledge barriers	Cross-team coordination breakdown, thousands of redundant APIs, conflicting standards, increased onboarding time, mounting operational overhead
Security Blind Spots and Shadow APIs	Undocumented interfaces created outside official workflows, lack of centralized inventory and lifecycle management, no security scanning protocols	Exposed API secrets creating entry points for attackers, potential multimillion-dollar financial losses, serious reputational damage to

Publisher: SARC Publisher

		organizations
Performance and Cost Degradation	Poor API performance across geographically diverse cloud infrastructure, hidden resource drain that accumulates silently over time	Industry-wide estimated annual losses reaching tens of billions, impossible root cause analysis after problems manifest
Compliance and Regulatory Failures	APIs operating outside organizational visibility, lack of proper governance controls, inability to track and manage API security posture	Regulatory violations, customer experience deterioration, difficulty meeting compliance requirements, cascading business disruptions

SLOW DEVELOPMENT VELOCITY AND DELAYED INNOVATION

According to APIwiz, API security gaps force 66% of companies to delay new releases, which harms both time-to-market and competitive edge (APIWIZ). This proves that the consequences of poor API governance extend to an organization's ability to innovate and compete. Moreover, the duplicated effort, the integration bottlenecks, the inconsistent versioning, and the lack of lifecycle management leave development teams trying to resolve conflicts and debug undocumented endpoints rather than innovating, ultimately delaying product launches.

In late 2024, Volkswagen Group had a data breach that leaked the personal data of over 800,000 electric vehicle drivers. It was caused by a misconfigured API accessible through an AWS instance. The leaked data included names, emails, GPS history, and daily movement patterns. It even leaked sensitive data about high-ranking government personnel, highlighting the real-world consequences of poor API governance (Kharod, S. 2025).

Such events often shift the focus away from innovation as organizations scramble to deal with large-scale internal investigations, reputational damage, and executive shakeups. Consequently, development roadmaps take the back seat as focus shifts to damage mitigation, internal audits, and rebuilding customers' and stakeholders' trust, effectively paralyzing the entire organization. In fast-moving sectors like automotive and finance, such delays can result in stalled product launches, lost opportunities, and falling behind for good.

ONGOING REGULATORY & COMPLIANCE OVERHEAD

In addition to causing engineering and security challenges, poor API governance can become a regulatory liability that steadily drains enterprise resources. With undocumented, inconsistent, and unmanaged APIs, organizations open themselves up to compliance violations, which can result in hefty fines, breach notifications, and prolonged audit cycles. Many compliance frameworks, including GDPR, CCPA, HIPAA, and PCI-DSS, require tight control over data access and traceability, both of which are compromised when APIs are not properly governed.

The scale of financial risk is immense. According to a 2023 report by MicroSolved, API-related breaches resulted in an estimated $12 to $23 billion in losses in the U.S. alone, and up to $75 billion globally (Huston, B. 2025). These costs include fines, forensic investigations, legal settlements, operational remediation, and brand damage.

One recent case illustrates this vividly: In January 2025, the New York State Department of Financial Services (NYDFS) fined PayPal $2 million after a 2022 data breach that exposed the sensitive data of over 35,000 customers (Wilson, S. 2025). The breach was linked to unauthenticated access to an exposed API, further underscoring the regulatory dangers of leaving APIs ungoverned.

But PayPal's case is just the tip of the iceberg. In 2023, T-Mobile disclosed that attackers accessed customer data through a poorly secured API, affecting over 37 million accounts (Bicchierai, F. L. 2023). Consequently, T-Mobile agreed to pay $15.75 million to the U.S. Treasury and make a $15.75 million investment over the next two years to bolster its internal technology (Morris, A. 2024).

Such numbers show how even the smallest of oversights can lead to millions of dollars in fines, settlements, operational remediation, and most importantly, brand damage, underscoring the increasing need for robust API governance policies across the industry.

Publisher: SARC Publisher

Table 3: Development Velocity Challenges and Compliance Costs in Ungoverned API Environments (APIWIZ; Kharod, S. 2025; Wilson, S. 2025; Morris, A. 2024)

Impact Category	Governance Failures and Triggers	Organizational Consequences
Development Velocity Reduction	API security gaps, duplicated development efforts, integration bottlenecks, inconsistent versioning protocols, lack of lifecycle management systems	Majority of companies forced to delay new product releases, compromised time-to-market capabilities, reduced competitive positioning in rapidly evolving markets
Innovation Paralysis	Misconfigured APIs exposing sensitive customer data, undocumented endpoints requiring extensive debugging, conflicting development priorities between security and feature delivery	Development teams diverted from innovation activities to conflict resolution, debugging efforts replacing creative problem-solving, product launch delays becoming systematic organizational issues
Crisis Management Overhead	Large-scale data breaches involving hundreds of thousands of customer records, exposed personal information including location data and government personnel details	Organizational focus shifts to damage mitigation, internal investigations, executive restructuring, customer trust rebuilding efforts effectively paralyzing normal operations
Regulatory Compliance Violations	Undocumented and unmanaged APIs failing to meet compliance framework requirements, inadequate data access controls, compromised traceability across multiple regulatory standards	Exposure to hefty regulatory fines, mandatory breach notifications, prolonged audit cycles, violations of GDPR, CCPA, HIPAA, and PCI-DSS compliance requirements
Financial and Reputational Penalties	Unauthenticated access to exposed APIs, poorly secured interfaces allowing unauthorized data access, systematic governance oversights accumulating across enterprise systems	Multi-million dollar regulatory fines, Treasury payments, mandatory technology investments, forensic investigation costs, legal settlements, irreversible brand damage affecting long-term market position

RETHINKING API GOVERNANCE: A DEVELOPER'S PERSPECTIVE

In an increasingly complex digital ecosystem, organizations look at API governance as a protective force. It ensures consistency, compliance, and control. However, from a developer's standpoint, I have often come across situations where it can become a bottleneck rather than a safety net.

API governance is meant to empower teams, but rigid governance structures often do the opposite by restricting innovation, causing friction between teams, and leading to a delayed time-to-market. I am not claiming to be an industry authority on governance frameworks; rather, the idea is to offer a fresh perspective from the developer's seat, where the impact of rigid governance policies is felt the most.

This section will focus on the need for modern governance policies to evolve into something more adaptive, scalable, and aligned with the developer experience.

Evolving the Governance Mindset

Evolving the governance mindset doesn't mean getting rid of everything that works. The fundamentals are there for a reason and solve real problems. What we need to focus on are the unanticipated consequences that some of the governance policies can create.

Suppose an internal team begins developing a prototype for a fraud detection API powered by machine learning. In the early stages, they need to expose experimental endpoints quickly for internal testing, before any finalized naming or versioning conventions have been approved. They also need to integrate with third-party APIs that might break internal security scanning rules, use a new language (e.g., Rust or Julia) for performance reasons that isn't on the "approved" list, and deploy to a new microservice architecture that's not yet recognized by the deployment gatekeeping system. Even though these developers aren't creating shadow APIs, skipping documentation, or ignoring security guidelines, they still get blocked because the governance model was built for consistency and risk control, not for exploration and exception handling.

As a result, they might try to bypass the official controls and deploy shadow APIs independently, leading to a loss of centralized visibility. So, the system "worked on paper" but failed in reality

Publisher: SARC Publisher

because it lacked adaptability. This is where traditional API governance frameworks need evolution, because if they don't evolve to handle novelty, they ultimately undermine themselves. It is time that organizations move from "governance as enforcement" to "governance as enablement," making it more adaptive and context-aware.

Operationalizing Governance Through Automation and Tooling

Many organizations still think about governance as an afterthought; often decoupled from the developer experience. However, in the real world, governance only works when it is made a part of the daily workflow. Through automation and tooling, it is now possible to embed governance into the development life cycle. Linting rules, CI/CD gate checks, design validators, and standardized API scaffolds can make adherence effortless and scalable.

Platforms like MuleSoft's Anypoint provide a compelling example of how governance can be embedded directly into the developer workflow without becoming an obstacle. With prebuilt policy enforcement, flexible rule configuration, and tiered permissions for developers and architects, organizations can find the perfect balance between centralized control and local autonomy.

However, as mentioned earlier, overly rigid governance may achieve short-term compliance but backfires in fast-moving environments where flexibility is essential. To move past this bottleneck, we need to design governance systems that are not just strict, but smart and capable of guiding developers without restricting them.

Rather than hardwiring every rule into the system as a binary pass/fail, organizations should build governance automation that distinguishes between critical and non-critical policies. For example:

Critical: Authentication required, rate limits enforced, PII not exposed.

Non-critical: Consistent naming conventions, field ordering, and documentation completeness.

Only the first category should block a build or deployment. The second should warn and provide actionable guidance rather than halting progress. Developers should also be able to flag intentional deviations with a reason. The exceptions could be logged with metadata, made visible in dashboards or review tools, and be subject to periodic audits. This might seem like intentionally leaving loopholes in the governance policies, but in reality, it acknowledges the realities of development by offering a flexible governance model that empowers the people doing the actual work, rather than controlling them.

Democratizing Governance: Roles, Ownership, and Community Input

The stability promised by API governance hinges on one critical assumption: that the rules are both sound and accepted by those expected to follow them. When developers aren't involved in shaping the policies they're expected to follow, friction becomes inevitable. To avoid this, governance must be democratized, not by dismantling structure, but by inviting broader participation in defining, refining, and applying it.

Effective governance should not rest solely on the shoulders of a central team. Instead, it should become a shared practice where:

- Developers get a say in the rules relevant to their domains (e.g., defining how APIs in their teams should handle authentication or pagination).
- Leads and architects review and approve new governance proposals, ensuring alignment with larger enterprise goals.
- Reviewers and maintainers enforce governance through automated tooling (like linters and CI checks), not by manual gatekeeping.
- It is important to clarify that this does not mean abandoning the idea of a central framework. Without a baseline, every team would end up creating its own version of governance, leading to fragmentation, inconsistency, and the exact problems governance is meant to solve. Instead, the central governance team should serve as:
- Custodians of core principles, such as naming conventions, data privacy, versioning strategy, and security requirements.
- Facilitators provide the tooling, documentation, and feedback mechanisms that enable others to contribute meaningfully.
- Ultimately, democratizing governance is not about loosening standards. It's about shifting from a model of control to one of collaboration. Governance remains essential, but it becomes stronger, more resilient, and more widely adopted when those expected to follow it are also empowered to shape it.

Publisher: SARC Publisher

Incorporating AI in Modern API Governance

In recent years, AI-powered tooling has started playing a larger role in API governance. Many platforms now use automated models to detect issues, apply rules, and generate documentation. For example, some systems use behavior-based detection to catch unusual API traffic or parse OpenAPI specs to recommend policy changes. In large, distributed ecosystems, this provides speed, consistency, and automation.

But these benefits often come with hidden costs, especially for developers. Most AI systems used in governance operate as black boxes. Developers can't see how decisions are made, when thresholds are triggered, or why suggestions are offered. That lack of visibility erodes trust and makes it harder to debug unexpected behavior. Worse, many systems don't allow overrides, leaving teams stranded when a model gets it wrong.

In 2019, Netflix began transitioning to a federated GraphQL architecture to manage the growing scale of its Studio engineering teams. This shift allowed individual teams to independently define parts of the API while contributing to a unified schema. But it also introduced governance challenges, especially around coordinating schema design across teams. Developers started experiencing friction when schema changes required centralized review, leading to slower iteration and growing internal pushback (Medium, 2020).

AI systems face the same risks, but with even less transparency. When governance blocks progress without explaining why or offering a way forward, developers try to route around it. That's a failure in both design and trust. Platforms like MuleSoft offer a better path. Its Anypoint Platform combines automated policy enforcement with real-time monitoring, schema validation, and clear feedback. Developers gain consistency and control without being locked out by opaque decisions.

Governance that serves the platform must also serve the people building on it. It's time to rethink how we design these systems. Developers should be able to see why a rule was applied, what triggered it, and how to change it. They should understand how automated decisions are made — and have the power to override them when needed. Transparency, context, and control are what turn governance from a blocker into a guide.

Table 4: Adaptive Governance Frameworks: Balancing Control and Innovation in API Development (Medium, 2020)

Governance Component	Traditional Challenges and Limitations	Modern Adaptive Solutions
Governance Mindset Evolution	Rigid structures restrict innovation, create team friction, delay time-to-market, block experimental development for fraud detection APIs and machine learning prototypes	Shift from "governance as enforcement" to "governance as enablement," making systems adaptive and context-aware to handle novelty and exception scenarios
Automation and Tooling Integration	Governance treated as afterthought, decoupled from developer experience, overly rigid systems achieving short-term compliance but backfiring in fast-moving environments	Embed governance into development lifecycle through linting rules, CI/CD gate checks, design validators, standardized API scaffolds making adherence effortless and scalable
Policy Classification and Enforcement	Binary pass/fail systems hardwired into deployment processes, no distinction between critical and non-critical requirements, developers unable to flag intentional deviations	Distinguish critical policies (authentication, rate limits, PII protection) that block deployment from non-critical ones (naming conventions, documentation) that provide guidance warnings
Democratized Governance Structure	Central team control creating friction when developers lack input in policy creation, manual gatekeeping processes, rules imposed without stakeholder acceptance or understanding	Shared practice where developers influence domain-specific rules, leads approve proposals, automated tooling enforces standards, central team serves as facilitators rather than controllers
AI-Powered Governance Systems	Black box decision-making processes, lack of transparency in rule application, no override capabilities when models make incorrect decisions, developers	Transparent systems showing rule triggers and decision logic, clear feedback mechanisms, developer override capabilities, combining automated enforcement with real-

Publisher: SARC Publisher

	unable to debug unexpected behavior	time monitoring and schema validation

CONCLUSION

Poor API governance erodes systems slowly through fragmentation, duplicated effort, and hidden technical debt. By the time its costs become visible, the damage is already done. That is exactly why effective governance needs to be proactive rather than reactive and baked into day-to-day workflows rather than enforced as an afterthought.

However, the goal of operationalizing governance isn't simply to centralize control; rather, it's to enable developers to adopt the right behaviors with ease. When governance frameworks are transparent, adaptive, and developer-centric, they reduce accidental violations, allow justified deviations, and promote trust across the organization. On the other hand, organizations that fail to invest in thoughtful governance frameworks risk losing alignment, velocity, and the trust of their teams. And that's a price no architecture can afford.

REFERENCE

1. Trend Micro, "Twilio Authy Data Breach: 33 Million Phone Numbers Compromised." (2024). https://news.trendmicro.com/2024/07/10/twilio-authy-data-breach/
2. Freehling, H. "The Dell API Breach: It could have been prevented." *Salt,* (2024). https://salt.security/blog/dell-breach-review
3. U.S. House of Representatives, Committee on Oversight and Government Reform. "The Equifax Data Breach." *Washington, DC,* (2018). https://oversight.house.gov/wp-content/uploads/2018/12/Equifax-Report.pdf
4. O'Neill, D. "Over $90 Billion Lost Each Year to Poor API Quality," *EIN Presswire,* (2023). https://www.einpresswire.com/article/652619603/over-90-billion-lost-each-year-to-poor-api-quality
5. APIWIZ, "The Hidden Costs of Siloed Tools." https://www.apiwiz.io/use-case/api-governance
6. Beaujard, S. & Charikova, A. "Methodology: How we discovered over 18,000 API secret tokens." *escape.tech,* (2024). https://escape.tech/blog/how-we-discovered-over-18-000-api-secret-tokens/
7. Kharod, S. "Volkswagen's API Breach Exposed: How Observability Can Save Data." *Treble,* (2025). https://treblle.com/blog/volkswagen-api-breach-observability-security
8. Huston, B. "The Ripple Effect of API Breaches: Analyzing Business Consequences and Mitigation Strategies." *State of Security,* (2025). https://stateofsecurity.com/the-ripple-effect-of-api-breaches-analyzing-business-consequences-and-mitigation-strategies/
9. Wilson, S. "PayPal 2 Million Data Breach Settlement." *ABCMoney,* (2025). https://www.abcmoney.co.uk/2025/07/paypal-2-million-data-breach-settlement/
10. Bicchierai, F. L. "T-mobile says hacker accessed personal data of 37 million customers." *ene* (2023).
11. Morris, A. "T-Mobile takes $31.5m FCC hit for data breaches." *TelcoTitans,* (2024). https://www.telcotitans.com/deutsche-telekomwatch/t-mobile-takes-315m-fcc-hit-for-data-breaches/8512.article
12. Medium, "How Netflix Scales its API with GraphQL Federation (Part 1)." (2020). https://netflixtechblog.com/how-netflix-scales-its-api-with-graphql-federation-part-1-ae3557c187e2

Source of support: Nil; **Conflict of interest:** Nil.

Cite this article as:

Depa, V. R. "The Hidden Cost of Poor API Governance in Large Organizations." *Sarcouncil Journal of Engineering and Computer Sciences* 4.8 (2025): pp 423-430.

Publisher: SARC Publisher