Powering the API world

EBOOK

Leading Digital Transformation: Best Practices for Becoming a Secure API-First Company

The enterprise playbook to adopt consistent API governance, enforce API security, and execute fast and with accountability.

By **Marco Palladino**,
CTO and Co-Founder, Kong

KongHQ.com

Content

Introduction

Over the last two decades, APIs have revolutionized every industry, fueling digital growth across organizations, and transforming the way we interact with the world. But more than that, APIs power the web, representing over 83% of global internet traffic.[1] New innovations such as generative Artificial Intelligence, blockchain, and related Web3 projects will only serve to accelerate the growth and adoption of APIs.

Although business and engineering leaders have accelerated API usage and integration across their companies, understanding how to effectively scale and secure APIs remains a serious challenge. This knowledge gap and lack of proper governance can yield disastrous results. In fact, APIs are increasingly becoming the primary target for internet-based attacks.

In response to this growing trend, we worked with Kong data analysts and outside economists to develop research to help us better understand the cyber-security risk of poorly managed APIs that we face in the coming years. The results of our systematic analysis were surprising, even to me.

Some of the topline results our team surfaced include:

- Between the years 2021 and 2030, we project a surge of 996% in API attacks. This is more than a simple increase — it signifies an explosion in the frequency and severity of API related cyber threats.
- We project an average annual increase of 31% in API attacks over this decade. This underscores the persistent efforts by hackers to exploit API vulnerabilities at an escalating rate.
- Not only is the frequency of API-related attacks increasing, but so too are the costs. As of 2023, the average cost for a security breach stands at $6.1 million. This figure reflects not only the direct expense of remediation but also the lost value associated with damage to brand reputation among customers and partners. According to our estimates, these costs are on an upward trend, predicted to be 95% higher, $14.5 million, by 2030.
- In the United States alone, the economic cost of these attacks is currently $10.6 billion per year, a figure that is escalating rapidly. We project the national cost to reach $198 billion within the next seven years, amounting to a cumulative cost of $506 billion this decade.
- Our team has analyzed the U.S. Congressional Budget Office's estimates of the GDP for 2030 and determined that the costs to the American economy will approximate 0.6% of U.S. GDP. To provide some perspective on this figure, these costs exceed the entire GDPs of countries such as Singapore, Israel, and Ireland, respectively.

Recent history bears witness to these trends, with Australian organizations reeling from a series of API-led cyberattacks. These have resulted in severe data breaches, customer data leaks, and a string of class-action lawsuits. The Optus breach, impacting 40% of the Australian populace, sparked intensified government scrutiny of organizational data security and management practices.[2] Two weeks later, Telstra Health fell victim to an API-related hack.[3]

In the U.S., cyberattacks have disrupted the critical infrastructure of one of the largest oil pipelines, leading to oil shortages across the East Coast.[4] This event compelled the White House to release an executive order to mandate zero-trust security for organizations with significant national security relevance.[5] Similar attacks have occurred around the world with major consequences for both the customers and the organizations involved.

As the Chief Technology Officer of Kong, I have the privilege of working with some of the best engineers and technical talent in the API space. I'm also fortunate in being able to work with business and tech leaders from across Fortune 500 and Global 2000 companies, advising on the implementation of robust, secure API strategies that have helped enterprises reach new heights, securely.

While most organizations understand the importance of APIs, too few recognize that APIs constitute mission critical infrastructure that demands the appropriate management. As we've seen with recent API cyberattacks, insufficiently secured and managed APIs can result in severe damage to an organization's reputation, harm customers, and risk the careers of those overseeing the API infrastructure.

In this eBook, we'll analyze a few common mistakes that I've seen organizations make when it comes to APIs — as well as best practices for any organization looking to become an API-first enterprise. My goal is to equip you with insights that will enhance your API management at scale and ensure the long-lasting success of your API strategy.

Are APIs really mission critical infrastructure?

APIs are at the core of every contemporary user experience, enabling us to develop engaging mobile and web user interfaces. They allow organizations to establish global partner networks to expand markets and boost revenue. By transforming isolated products into open platforms, APIs foster innovation and facilitate developer ecosystems. They empower engineering teams to enhance agility and expedite deployment by reusing existing services — and data — in a self-service mode. In essence, APIs serve as the system of record for all resources an organization provides, both internally and externally.

When you consider that APIs are at the heart of everything that an organization does, they undoubtedly constitute mission critical infrastructure. But are they always treated as such? The answer, unfortunately, is no, and therein lies the problem.

I've noticed a pervasive inconsistency: while APIs are recognized as mission critical, their infrastructure is **not** accorded the same importance. This leads to situations where malicious actors can easily identify and exploit discrepancies in API management and security, gaining system access and inflicting damage on enterprises and their customers.

This inconsistency often stems from a well-intentioned but ultimately problematic decision: distributing API infrastructure ownership across multiple teams. While this approach aims to promote speed and autonomy, it unintentionally triggers a chain reaction of adverse issues affecting internal systems and controls, the consequences of which can persist for years.

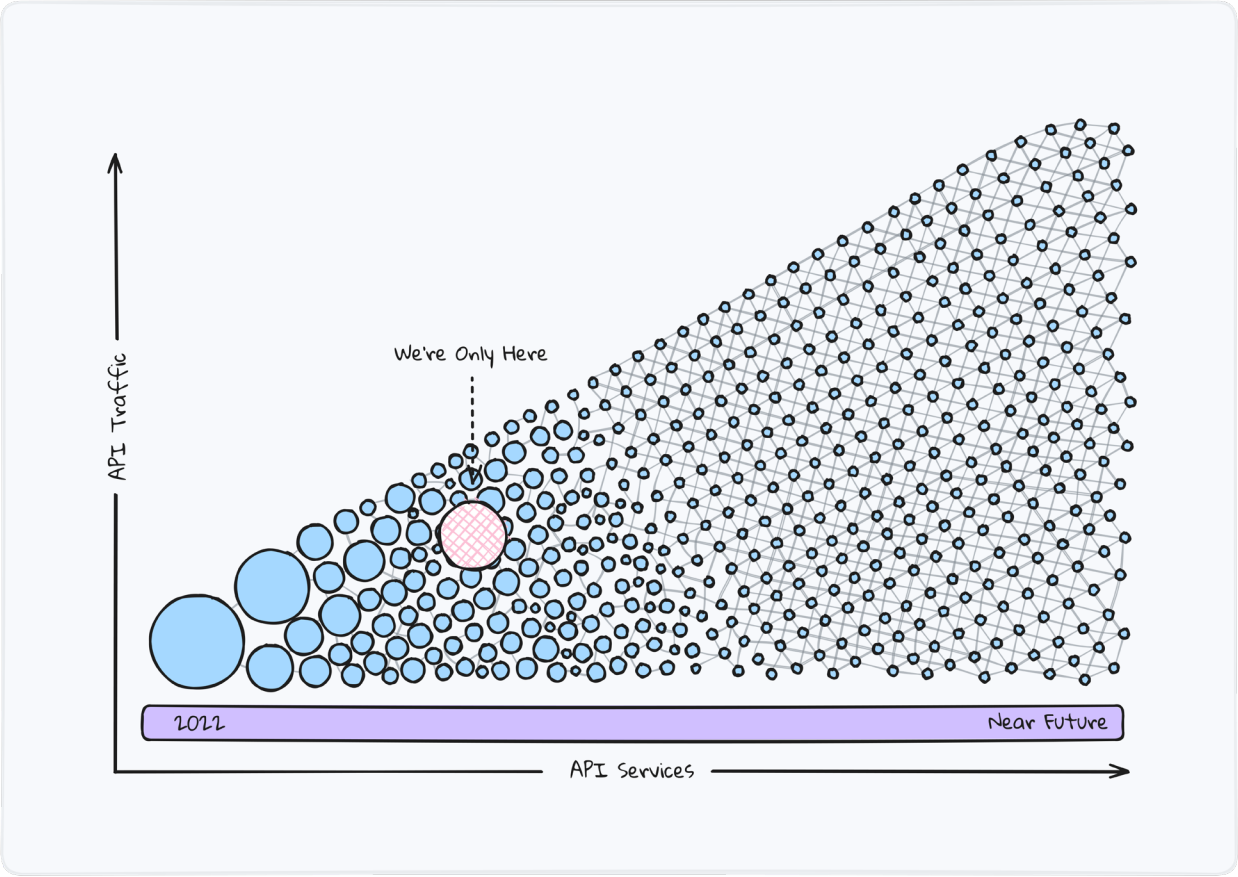
In practice, we can enable teams to move fast and take ownership of operational policies applied to the underlying API infrastructure without disseminating core infrastructure ownership. The ideal objective is to have teams act as "users" of API infrastructure, not its "builders."

This idea becomes even more crucial when the organization is a national security asset, such as financial institutions or telecommunication companies. In such cases, API infrastructure should comply with the same rigorous standards and regulatory mandates as other corporate and operating functions. Regrettably, this compliance often falls short when infrastructure ownership is spread across teams.

Building a robust and reliable API infrastructure requires the establishment of an internal playbook that enables us to:

1. **Adopt consistent API controls across teams:** Develop unified, robust, and secure API infrastructure that minimizes inconsistencies in API policy creation and enforcement.
2. **Enforce API security by default:** Ensure that security controls are in place across all teams and workflows, clarifying the organization-wide responsibility for API infrastructure.
3. **Foster speed and self-service:** Encourage teams to innovate within an agile, self-service environment without compromising on consistency, security, or corporate responsibility.
4. **Promote accountability and responsibility:** Cultivate a culture of API accountability, mirroring the responsibility inherent in other organizational areas. There must always be clear ownership of API infrastructure, with accountability upheld 24/7/365.

The ultimate goal is to maintain continuous control over the API infrastructure that drives our organization's present and future. This can be achieved responsibly, ensuring that teams remain productive and agile. Without the right practices in place, however, an organization's ability to scale and expand its API portfolio is reduced, potentially exacerbating problems over time.



Technology leaders are expecting to build more APIs in the next five years, than all the APIs built until now. Are we prepared to manage this scale?

In subsequent sections, we'll dive deeper into each of these areas and build a framework for establishing modern API practices within organizations. By doing so, we can reduce the risk of becoming the next cyberattack victim.

Adopting consistent controls

APIs are being developed across all teams within an enterprise, and their numbers are exploding. Business leaders I've consulted, particularly from the financial services and tech sectors, have told me they plan to introduce more APIs in the next five years than they've created to date. This expected growth necessitates substantial scalability from our API infrastructure to support increased productivity, business expansion, and innovation over the next few years.

Understanding the API development process is key to implementing robust and consistent controls. These controls can minimize the risk of fragmentation and potential backdoor vulnerabilities in our API management system.

API development process

Developers responsible for creating and putting APIs into production typically follow a four-step process:

1. **Design:** Initially, the API must be designed. The team must compile a comprehensive understanding of the requirements they need to meet while designing the API interface, often collaborating with product management. Typically, they pose questions like, "What should the API do? What endpoints are required? What will the responses look like?" This stage is largely research-oriented, involving stakeholders who will consume the APIs. At this point, no code is written, but it's crucial to consider extensibility, as the API's functionality will inevitably evolve over time. Early considerations for extensibility can help mitigate the challenges of migrating APIs to a different version in the future.

2. **Implementation:** After agreement on the API design, the team begins writing code to realize the functionality outlined in the design document. They choose an implementation language, consider a framework, and assess which dependencies to use to meet the original specification. In this stage, it's essential for the team to implement a thorough testing strategy to continuously confirm the API conforms to the expected behavior through unit and integration tests.

3. **Management:** At this point, the API typically lacks management capabilities. The team needs to consider how to protect the API, enforce security and entitlement checks, and utilize the right tools for managing traffic and access to the API once it's deployed in production. They ask, "How will we invite users to consume the API? How will we create and possibly revoke credentials? How will we monitor the API traffic?"

4. **Operation:** Finally, the API is up and running smoothly with API requests flowing through the system. Eventually, the team will want to make changes in production, such as releasing a new version of the API or introducing a beta version of a feature they've been working on under a feature flag. They'll need to use strategies like blue/green deployments or canary releases to gradually transition traffic to the new API version without downtime. Implementing traffic mirroring capabilities to test their staging environment with a subset of mirrored production requests for debugging and other operational changes will also be important.

While step 1 and step 2 are primarily focused on the **API development** cycle, step 3 is more concerned with the **API management** cycle, which is traditionally what APIM solutions are all about: to provide the right infrastructure to support APIs in production. Step 4 is all about the **API operational lifecycle** that a team will need to continuously implement throughout the journey of their own API in an agile way.

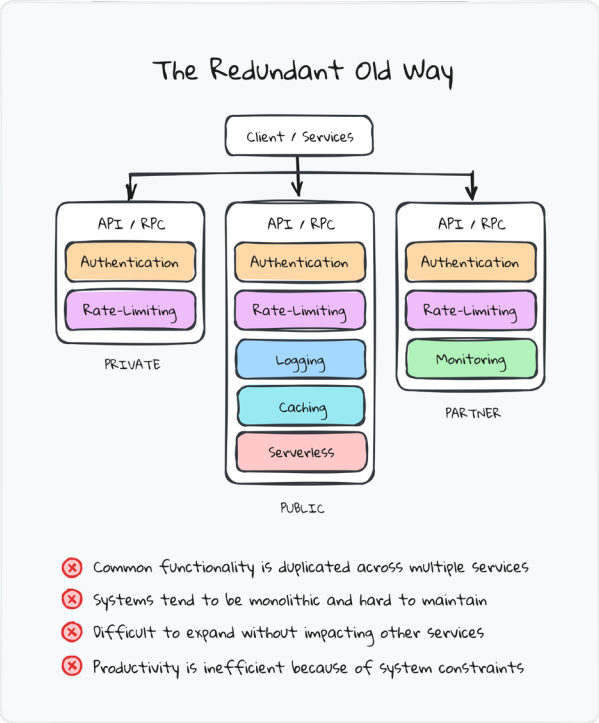
Most organizations confuse **API management** with the **API operational lifecycle**: the former is what the organization and the platform team are responsible for, and the latter is what developers need to be agile and ship fast. Often, organizations give freedom to the teams to implement both use cases, and by doing so they are distributing more API infrastructure ownership than they should to the application teams, which are not prepared to take full responsibility for both areas (despite sometimes happily accepting to do so). We will further address this topic in the "Moving fast" section later in the book.

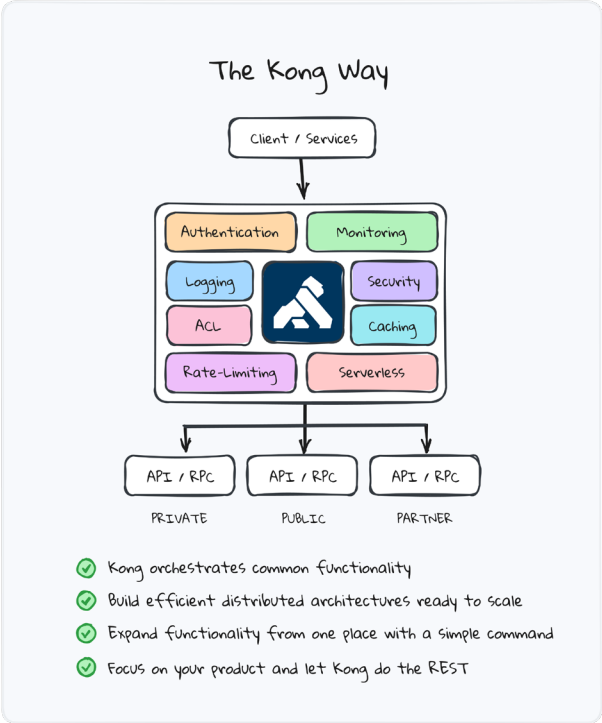
The importance of centralized API management

Without a centralized API management platform that can address the requirements of every API in the organization, teams often expand their API development steps to include ad-hoc API management capabilities. These are tailored to each API, leaving the platform team with no visibility or control. The inherent desire of developers to build API controls within their applications (as developers are builders by nature) conflicts with the need for organizational control over the security aspects of the underlying API infrastructure.

The absence of standardization on a consistent API platform and infrastructure leads to the following issues:

1. **Productivity loss:** Teams start operating outside their core competency area. Their expertise lies in building products and APIs for the organization, not in constructing underlying API infrastructure. Developers also tend to underestimate the complexities involved in building, maintaining, and updating ad-hoc API infrastructure over time.
2. **Fragmentation and risk:** Teams repeatedly build the same API controls for every API they create. This practice results in fragmented API infrastructure silos, reducing visibility on how APIs are secured and managed. This lack of oversight makes it challenging for the organization to assess the risk profile of the APIs across the company. Moreover, fragmentation creates a broad error surface that a malicious attacker can exploit, as each implementation of the same capability might be built slightly differently. The continuous maintenance and updating of these ad-hoc API controls further reduce both the reliability of the APIs and the productivity of the teams.





From a security perspective, the absence of “one control plane” to assess and manage the security profile under a single pane of glass creates a void that can be exploited by malicious actors. This also creates conflicts with security teams, who are unable to chase down every team to validate the security of their implementations, some of which they may not even know about (since these implementations are essentially “shadow IT”).

The API controls that are typically applied on a modern API infrastructure are:

- **Security:**
  - Network layer security: Firewall and DDOS protection, zero-trust security
  - Application security: AuthN/Z, credentials management
  - Traffic security: anomaly detection, backdoor testing
- **Traffic Management:** Rate limiting, tiering, ACL, blue/green deployments, canary releases, API traffic mirroring, and more
- **Analytics and logging:** API Monitoring, usage analytics, access logs and more.
- **Onboarding:** Registering for an API, being able to manage different environments (testing and production, for example)
- **Portal and collections:** Developer portal for API documentation, client libraries, request collections to more easily debug and use the APIs

From a security perspective, the importance of the API development steps (step 1 and step 2) lies in crafting a well-documented OpenAPI specification that meets API usage needs. Additionally, it's crucial to implement appropriate unit and integration testing to ensure that each API request doesn't disclose more data than necessary.

Kong provides a full lifecycle APIM and service mesh platform that also includes Kong Insomnia, an application used by hundreds of thousands of developers to create beautiful API design documents — based on the OpenAPI specification — in a collaborative way. To learn more, go to: https://konghq.com/products/insomnia.

The OpenAPI specification plays a key role in step 3, validating that those using the API can only request endpoints that are explicitly defined. This action stifles any potential attacker's ability to discover backdoors in the API by requesting endpoints not meant to be exposed.

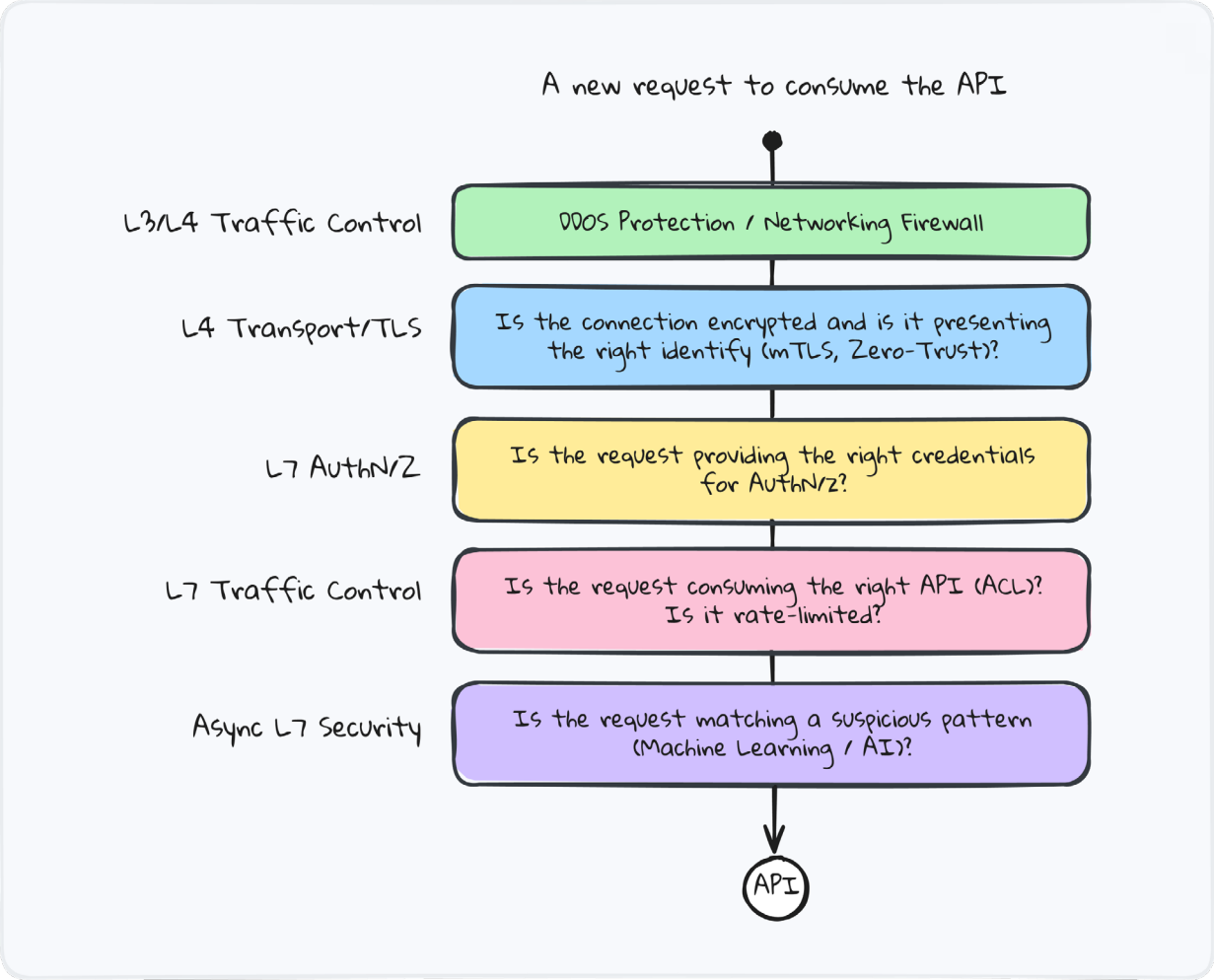
Because APIs are mission critical infrastructure, it's no longer acceptable to have siloed API infrastructure without distinct ownership or insight into the quality of the implemented controls. This outdated "feudal system" leaves each team on their own to protect themselves independently. Rather, we must ensure that the security team has properly assessed and validated the infrastructure to be compliant with their requirements.

If pockets of API infrastructure remain entirely under the application teams' management, it's practically impossible for any organizational leader to accurately assess risks and act accordingly. As recent cyberattacks have demonstrated, API security isn't merely the team's responsibility, as they cannot fully manage the entire lifecycle. It falls under the platform team's purview and yours, as a leader of the organization.

Through my discussions, a recurring theme emerges: leaders anticipate having time in the future to implement these changes and instruct application teams to utilize a more secure API platform provided by the platform team (and validated by the security team). Unfortunately, this non-specific point in the future rarely arrives, while the reality of our world clearly indicates that time has run out. The necessity to own API infrastructure is immediate, given that it's today's mission critical infrastructure, and it faces constant 24/7 attacks by malicious actors.

Enforcing API security

API security strategies are multi-layered and encompass various aspects of the API journey. We can identify at least five security layers that require enforcement:



The first level of security pertains to low-level network security at L3/L4, particularly crucial for edge APIs designed to be used by third parties outside the organization. We must inspect traffic flows using features like inbound encrypted traffic inspection, stateful inspection, and protocol detection. Implementing outbound traffic filtering is necessary to prevent data loss, assist with compliance requirements, and block known malware communications. We must scrutinize active traffic flow through stateful inspection, protocol detection, and more. Essentially, we need to verify the legitimacy of incoming traffic before progressing to the next security layer.

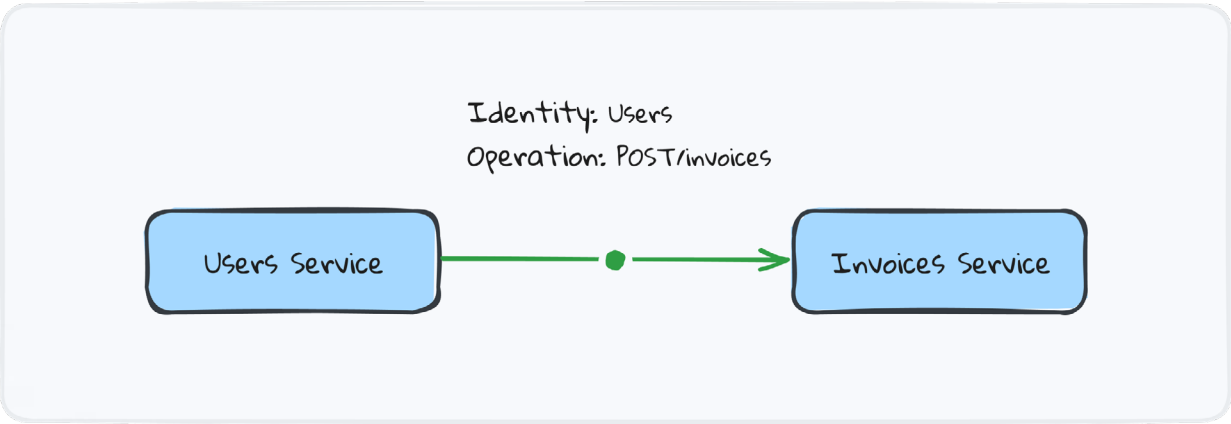
While these strategies are essential for external API traffic, forward-thinking organizations also implement stringent security measures for internal traffic. These measures help combat potential threats from internal malicious actors and bots.

It's an important shift for many leaders to recognize that we are safeguarding against both external and internal threats. Not implementing internal security measures creates an organizational vulnerability, as an attacker may run malicious software internally via other backdoors outside the API perimeter, threatening the API infrastructure itself. This acknowledgment leads us to the next layer of security.

The second layer of security is implementing the concept of **zero-trust**, to remove the concept of trust in our applications: we cannot trust that the client is who they claim to be, and we cannot trust internal clients or external ones.

A practical example: When we travel to foreign countries, we need to carry our passports to validate our identities at the border. Without passports, immigration agents would have to take our word for our identities — an easily exploitable system. Similarly, APIs resemble a border with no immigration control: anyone or anything can make requests and potentially spoof the identity of other clients to perform malicious operations.

To secure our "API borders," we need to implement a zero-trust solution that validates every client's identity. In the context of APIs, this "passport" could be an mTLS certificate issued to each service instance, validated with every request.

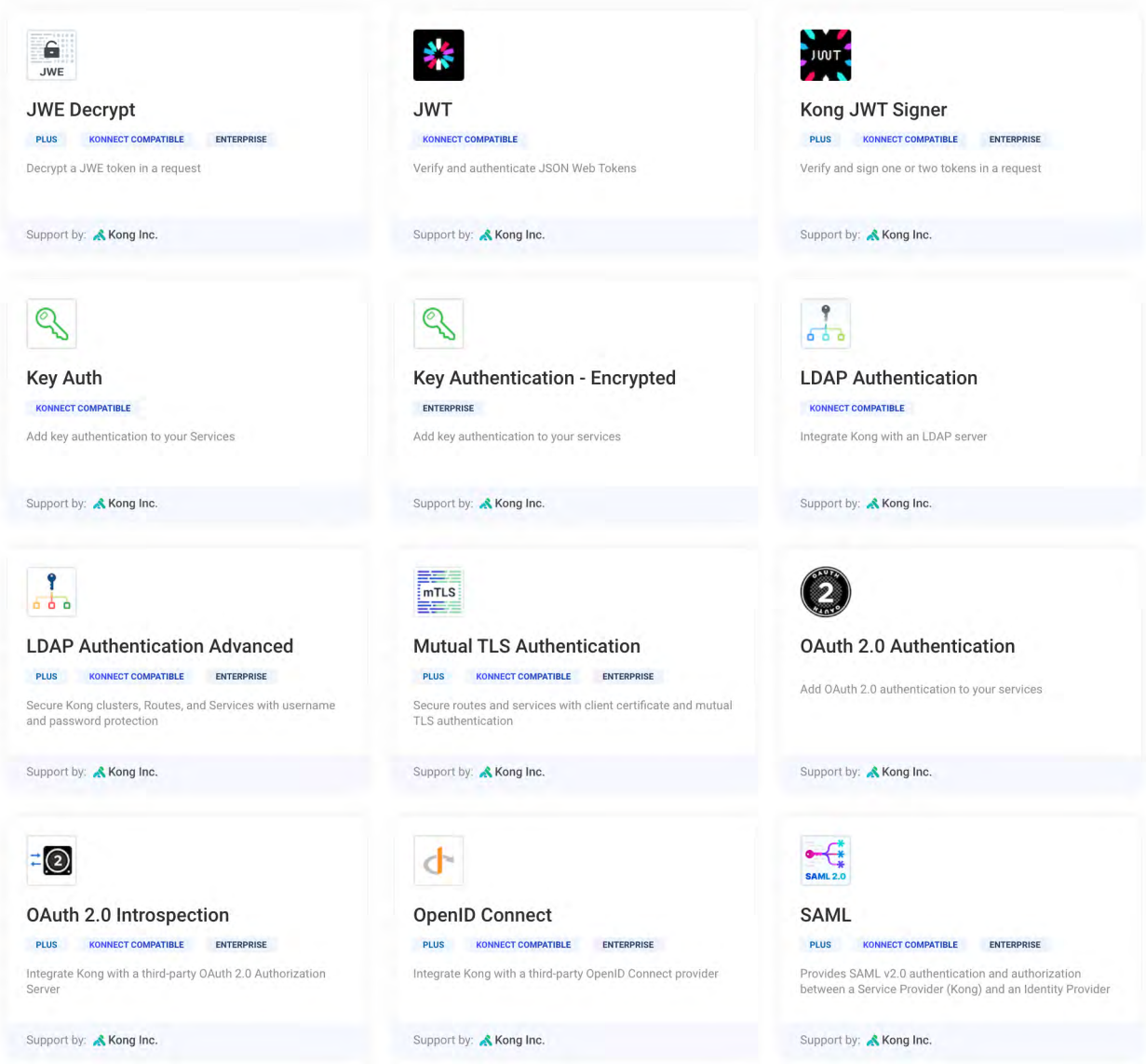


Implementing a zero-trust architecture across every application and environment could be a complex task. It would involve supporting enforcement across all runtimes, issuing, rotating, and revoking certificates for each service instance - potentially hundreds of thousands or even millions.

Fortunately, the implementation of a "Service Mesh" can simplify this endeavor. It manages the entire certificate lifecycle (issuance, rotation, revocation) automatically, while the enforcement is delegated to a sidecar proxy running transparently alongside our services. Hence, teams can be users of zero-trust, not builders.

Kong provides an enterprise service mesh product called Kong Mesh, that extends CNCF's Kuma and Envoy. Kong Mesh allows you to implement an enterprise zero-trust solution across the organization in days instead of years. It handles the whole certificate lifecycle across both containers and legacy virtual machines, across clouds and even private data centers. To learn more go to: https://konghq.com/products/kong-mesh.

Once we've validated the identities of the services utilizing our APIs, we need to further identify the user attempting the request using authentication and authorization (AuthN/Z) strategies. These strategies could involve validating an API key or integrating with a third-party OpenID Connect (OIDC) or OAuth provider. Such integration helps us ascertain both the user's identity and their permission level or entitlements, which dictate what operations the user can perform on the API. As we need this security level across all APIs our teams build, it would be beneficial to centralize how these policies are enforced. Decentralizing such enforcement could lead to considerable security risks.



Some of the many policies that can be applied for AuthN/Z with Kong Gateway.

Kong provides an enterprise APIM product called Kong Gateway that provides 10+ AuthN/Z plugins to seamlessly provide authentication and authorization out of the box to every edge or internal API. To learn more go to: https://konghq.com/products/kong-konnect.

Once we've authenticated the user and verified the legitimacy of the incoming request, we might still want to restrict the traffic directed toward the API. This control measure somewhat resembles the first layer of security, but it's further refined to manage user-level access. We may wish to implement rate-limiting or throttling strategies. Such strategies can help us limit API access to, for instance, prevent cascading failures if excessive traffic is received. They also enable us to create consumption tiers for our APIs, which can serve as an additional revenue stream.

Kong provides sophisticated traffic control capabilities both at the L4 mesh layer and at the L7 API Gateway layer, and it enforces them with extremely good performance and low latency to preserve the quality of the end-user experience. To learn more go to: https://konghq.com/products/kong-konnect.

Finally, even after all traffic has been validated through each of the preceding security layers, the most sophisticated enterprise organizations will still construct models of API traffic to detect any unexpected operations carried out by known users. This could be triggered by a known user turning malicious or by a known user's credentials falling into the hands of a malicious actor.

In essence, our API users are also vulnerable to attacks, and their credentials could be compromised without our knowledge.

To exercise this level of control, we may want to employ intensive API monitoring and analytics, coupled with asynchronous machine learning capabilities that construct a model of our traffic for each client and user. Again, building these enforcement measures on a team-by-team basis is challenging and should ideally be provided by the platform team, which is responsible for building the organization's mission critical, security-approved API infrastructure.

Policy enforcement

The final aspect to consider in API management is policy enforcement. This includes setting up our API policies, which is done by our organization's employees. This process isn't immune to human error, both accidental and intentionally malicious.

To mitigate this risk, every organization should enforce a policy approval workflow requiring at least one additional approval from a person in a separate team.

Policy enforcement has several facets:

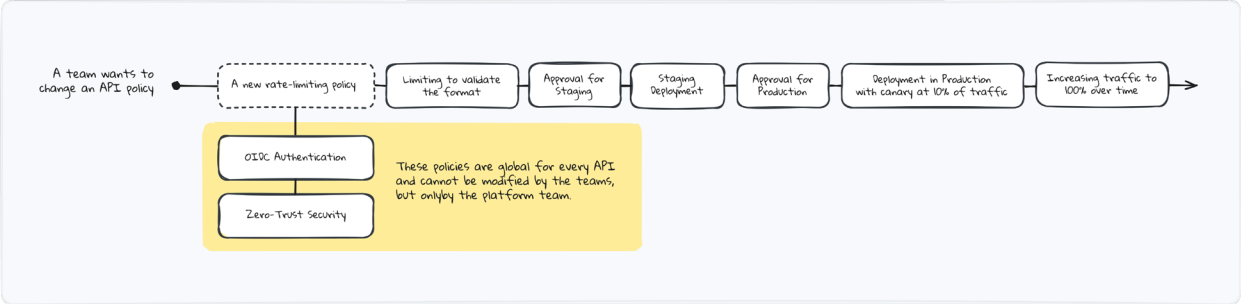
- Ensuring that necessary policies are applied
- Confirming that policies are correctly configured
- Checking that policies are neither malicious nor causing unexpected results

Just like every other topic in this book, policy enforcement is a cross-cutting concern that impacts every team within the organization. To enhance our teams' productivity, we should offer them a ready-made solution that has already been validated by both the platform and security teams of the organization.

Policy enforcement and control equate to compliance, which is mission critical.

It should not be fragmented into solutions for which individual teams are responsible, as this would result in a lack of visibility and corporate responsibility in assessing our API infrastructure policies.

We can further minimize the margin of error by implementing global policies that teams can't alter and that are always applied, such as security policies.



Kong provides controls for policy enforcement and linting throughout its Kong Konnect API platform that can be integrated via APIs and GitOps to any existing solution that your organization may be adopting. To learn more go to: [https://konghq.com/products/kong-konnect.](https://konghq.com/products/kong-konnect)

Moving fast

The desire for teams to move quickly and maintain agility often leads organizations to conflate ownership of infrastructure with ownership of policies. In reality, these responsibilities are **distinct and different:** the teams should be rapidly iterating and developing their API policies, while the ownership of infrastructure falls to the platform team.

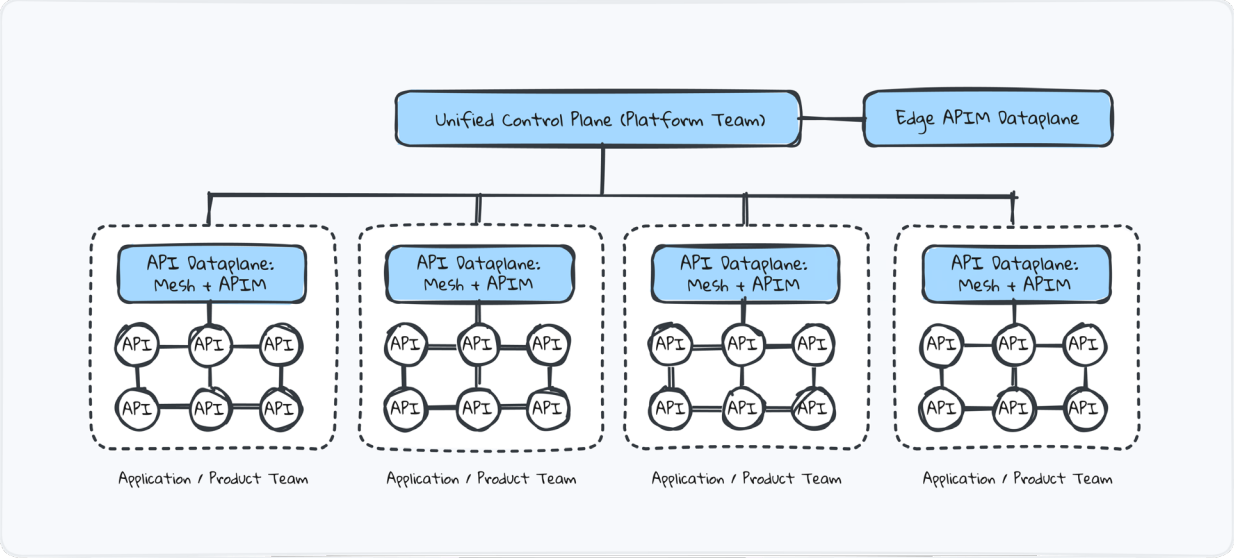
Teams are tasked with creating compelling products and user experiences, and they need to iterate quickly on their API policies to ship quickly. Typically, they need a subset of the capabilities that a modern API infrastructure can offer:

- Traffic routing configuration to new versions of the services and APIs, including:
  - Generic split traffic
  - Traffic mirroring
  - Blue/green deployments and canary releases.
  - Traffic introspection and request injection
- Feature flagging to route across different versions of an API
- Access to observability metrics and logs, and traces
- Access to other traffic capabilities like:
  - Retries
  - Timeouts
  - Circuit breaking

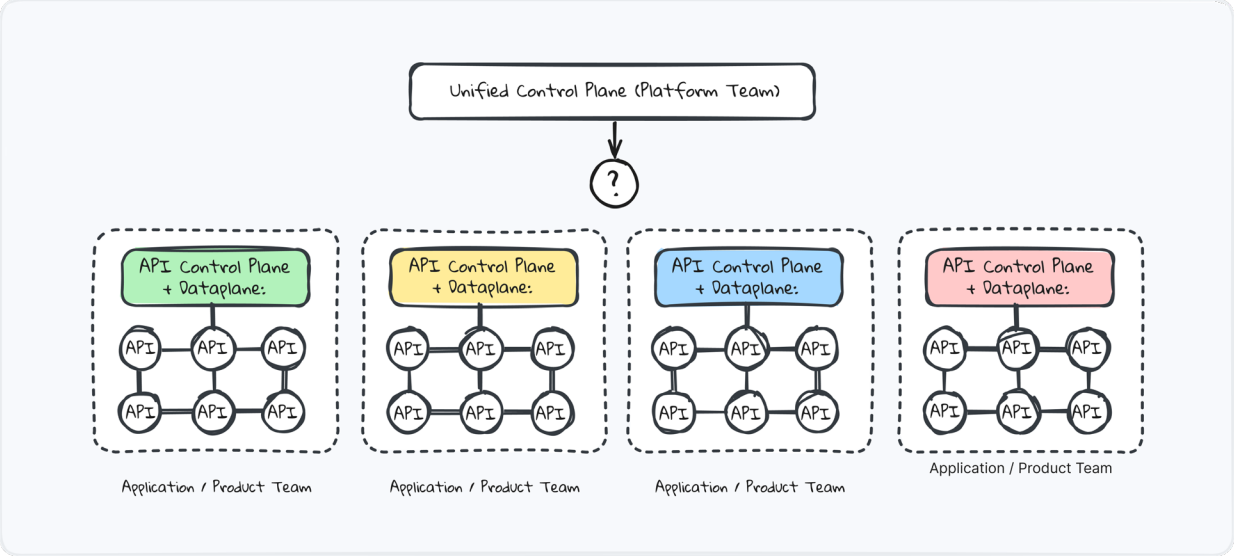
However, there are elements that the teams should not be responsible for:

- The overall deployment of the infrastructure
- Infrastructure operations, including no-downtime upgrades
- Security and encryption configuration of the traffic, as security is non-negotiable
- While they can influence authentication and authorization (AuthN/Z), it should align with the broader security stance of the organization —thus, they should not be the ones owning it
- Logging and debugging infrastructure, including tracing infrastructure —while teams should have access to these capabilities, they should not be the ones operating them
- Firewall rules
- Cross-cloud and cross-datacenter connectivity

When selecting API infrastructure technology, it's crucial to evaluate whether the technology allows for both infrastructure and configuration segregation. This ensures simplified deployments across the organization with a unified control plane, while still compartmentalizing the data plane infrastructure and the applied configurations:



In the absence of a solid strategy, teams may forge ahead and develop their own customized solutions for managing API infrastructure. In doing so, they may inadvertently encourage shadow IT practices, leading to a lack of control, scalability issues, organizational inefficiencies, and an increased risk of security vulnerabilities.



We all agree that inefficiencies and bottlenecks should not impede our teams. This situation can be avoided if the platform team provides them with a degree of autonomy to apply policies, while still managing the underlying infrastructure on their behalf, at both the API management and service mesh layers. This approach requires planning, and it might be tempting to delegate the entire setup to the teams.

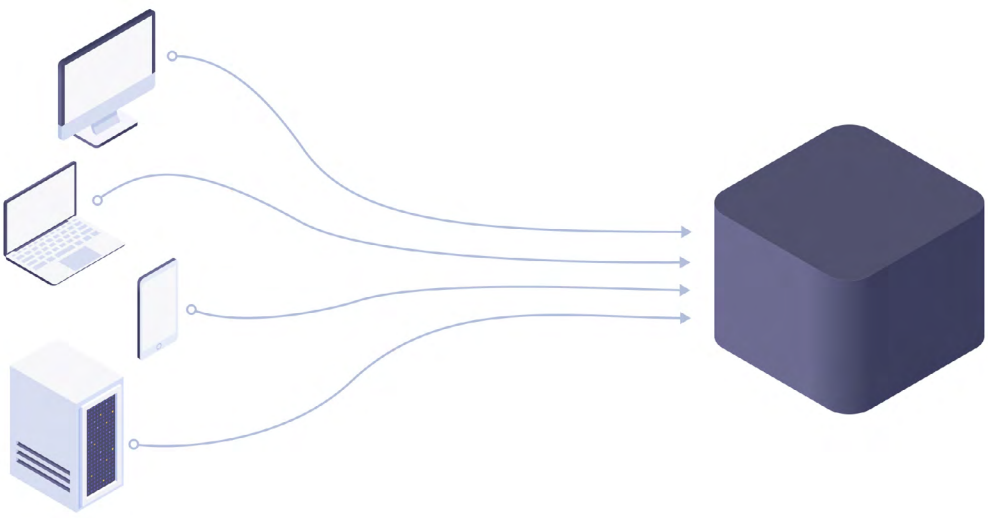
However, such a course of action will inevitably lead to disaster.

Our corporate responsibility is to fully own the API infrastructure without assigning to our teams a scope of responsibility that they might struggle to adequately address alongside their other tasks.

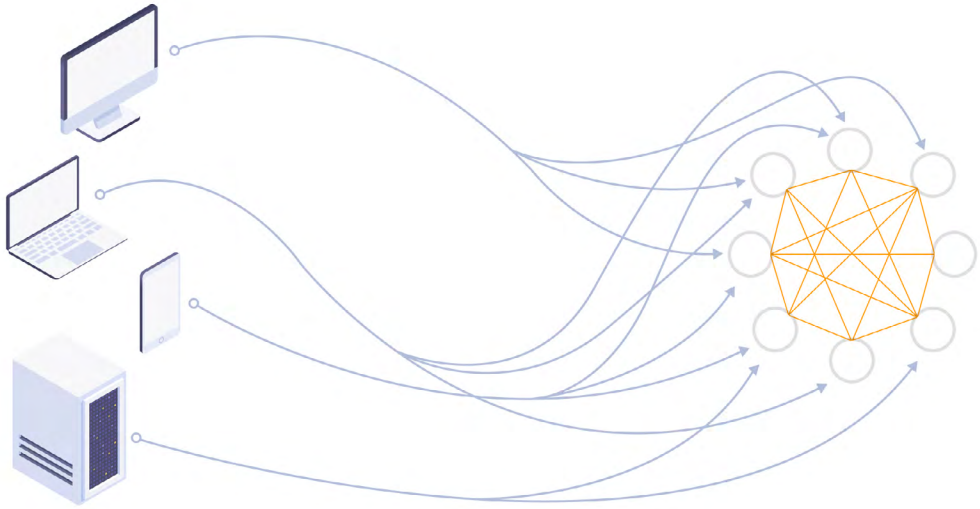
Kong provides both infrastructure and configuration segregation for hundreds of top Fortune 500 and Global 2000 organizations that have deployed a single pane of glass to manage API infrastructure across every team, while still allowing developers to be quick and agile in their rapid iterations. To learn more, go to: https://konghq.com

As our applications move from monolithic to microservices architectures, the networking requirements become more critical. Essentially, we're substituting the reliability of the CPU in monolithic applications with the unpredictability and security issues of the network in microservices. Of course, this trade-off brings tremendous benefits such as enhanced scalability, resilience, and agility.

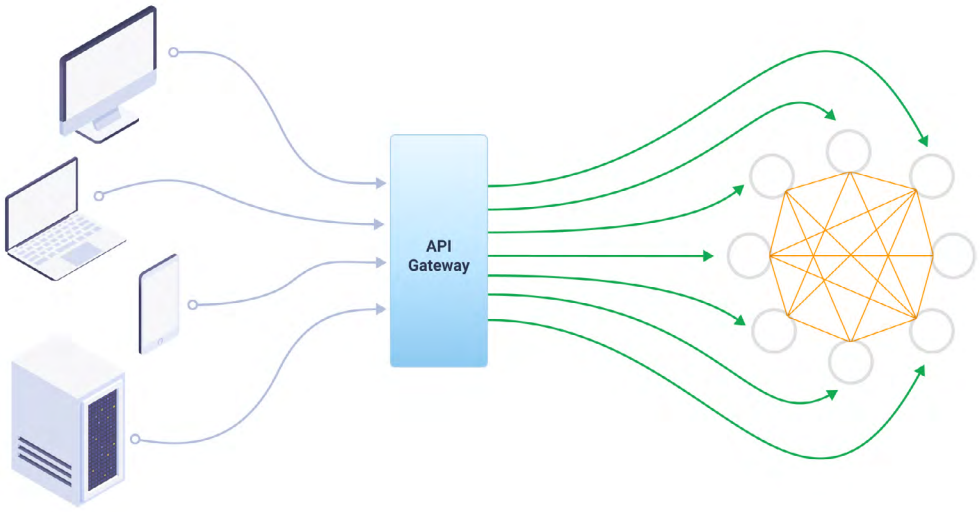
Before 2013: Direct access to the monolith



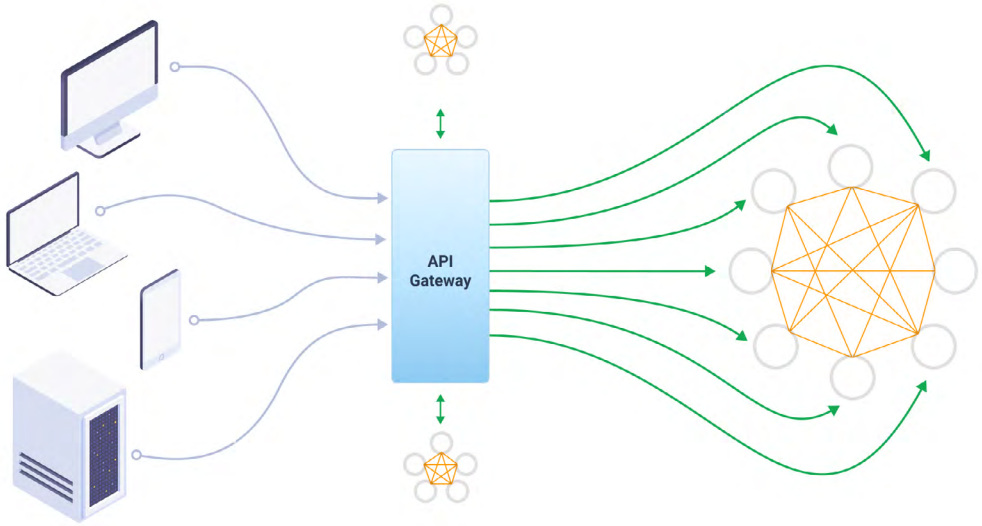
2013-2015: Direct access to microservices



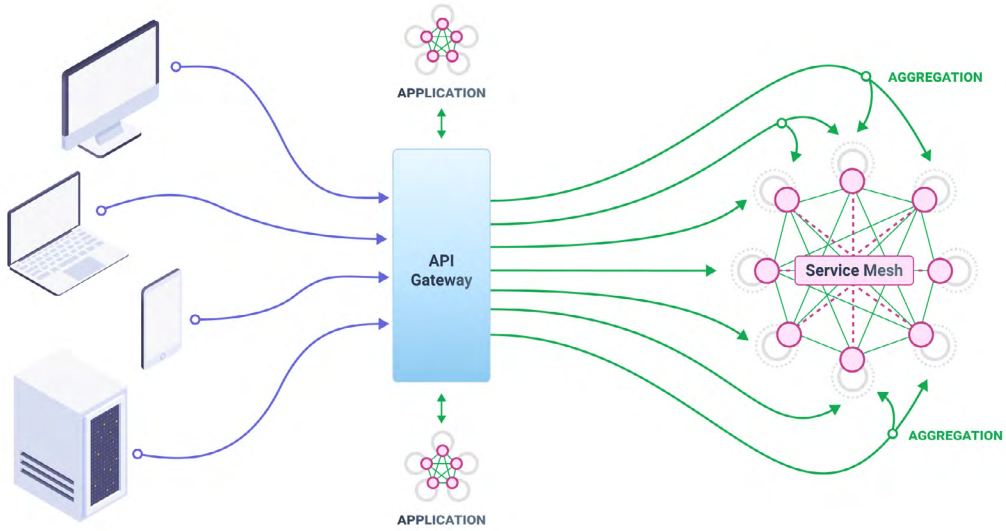
API gateway for unified ingress and control



For edge and internal communication



Service mesh overlay for security, HA and observability



The evolution of our applications to microservices has created more API traffic than ever before, at an unprecedented scale.

Ensuring accountability in execution

As a technology leader within your organization, one of your key roles is to establish an organizational structure that encourages responsibility and accountability.

As applications evolve, expectations from developers and architects do too. Our initial strategies may need to be revised to effectively manage risk, maintain compliance, and drive success. The decisions we made yesterday may not be optimal tomorrow.

In recent years, organizations have implemented short-term solutions to incentivize teams to transition quickly to new architectures such as microservices and Kubernetes to accelerate the process of breaking away from the constraints of traditional monolithic systems. In doing so, they granted considerable freedom to teams in making strategic infrastructure decisions. At that time, this approach made sense: we needed teams to iterate rapidly, drive success, and inspire others to follow their lead, unhindered by excessive constraints.

However, as more applications — and teams — adopted the microservices transformation, these early incentives have metamorphosed into problematic long-term solutions. We've seen the rise of shadow IT infrastructures, inefficiencies in teams tasked with both building products and managing infrastructure, and challenges for platform and security teams in assessing the security of our API environments. With the rapid explosion of APIs, it's time to reassess the situation and manage the organizational risks in this new era, where APIs and microservices are now the norm.

In the final analysis, our teams require scalable, secure infrastructure to succeed. They don't have the bandwidth to both build applications and manage infrastructure, especially considering the inevitable fallout of cyberattacks. When such attacks occur, the responsibility ultimately lands on the organization's leaders. Thus, it's time to recognize our API infrastructure as the mission critical component it has become, rather than an experimental component of an earlier era.

To achieve this, we need organizational processes that enable the security teams to approve the configurations deployed in our API infrastructure. The platform team should be aware of all running APIs and services to monitor, secure, and control them effectively. Teams should be able to focus on their applications and shift from building to using infrastructure. Our APIs are under constant threat today — we just might not be aware of it yet. It's only a matter of time before our internal inefficiencies transform into tangible attack vectors.

Conclusion

In conclusion, as APIs become the system of record for each data and service that the organization provides internally and externally, API infrastructure essentially becomes the nervous system of the modern enterprise. As such, the management of API infrastructure can no longer be distributed; it must be owned responsibly and with accountability like any other mission-critical system.

I hope this brief book has been helpful to you in assessing the tremendous changes our organizations are experiencing in this new API world. May it serve as a source of inspiration in your journey to transform your organization into a thriving, API-first enterprise.

Cheers,

Marco Palladino
CTO & Co-Founder at Kong

Methodology

Our study examined trends in API and cyber security vulnerabilities and used statistical modeling to forecast future API attacks and the inflation-adjusted economic costs of these attacks. Economic costs include both direct and indirect costs, weighted and adjusted for inflation as compiled by the U.S. Bureau of Labor Statistics. These economic costs were compared to U.S. Gross Domestic Product (GDP) productions developed by the Congressional Budget Office.

References

1. State of the Internet Security Report (2019), Akamai
2. Australian government slams Optus for cybersecurity breach (October 2022) Reuters
3. Australia's Telstra hit by data breach, two weeks after attack on Optus (October 2022) Reuters
4. The Attack on Colonial Pipeline: What We've Learned & What We've Done Over the Past Two Years (May 2023) CISA
5. Executive Order on Improving the Nation's Cybersecurity (May 2021) The White House

Powering the API world

Konghq.com

Kong Inc.
contact@konghq.com

77 Geary Street, Suite 630
San Francisco, CA 94108
USA