# SECURITY BENEFITS WITH POLYAPI

Enhanced API security with innovative and comprehensive protection methods.

Darko Vukovic

PolyAPI

# Summary

In the burgeoning digital landscape, where Application Programming Interfaces form the backbone of interconnectivity, ensuring their security has become paramount. PolyAPI emerges as an innovative player in this field, offering advanced solutions designed to counter the risks threatening API ecosystems. From data breaches to unauthorized intrusions, the potential perils can undermine an organization's credibility and financial health. This whitepaper tells the story of how PolyAPI's state-of-the-art techniques and methodologies reinforce the digital fortifications that businesses rely on.

The urgency of fortifying API security is echoed by startling statistics: 74% of organizations reported at least three API-related data breaches in the past two years, indicating a critical need for enhanced security measures. As APIs expand the attack surface across technology stacks—a sentiment shared by 58% of industry professionals—the challenges in managing API sprawl, accurate inventories, and third-party access grow increasingly complex. Even with tools like Web Application Firewalls WAF and Web Application and API Protection WAAP, over half of the organizations lack complete confidence in their current API security solutions. With 61% of organizations anticipating an increase in API risks over the next two years, the need for robust and innovative security solutions like those offered by PolyAPI has never been more pressing.

Traceable. (2023). https://www.traceable.ai/2023-state-of-api-security

# Table of Contents

# Introduction

API security has transcended the realm of being merely important; it has evolved into an indispensable aspect of numerous business operations. The consequences of lax API security can be catastrophic, with data breaches, unauthorized access, and service disruptions posing significant risks to an organization's integrity and financial stability. It is in this landscape that PolyAPI stands out by providing a number of solutions to these problems, tailored to meet the evolving demands of the modern enterprise.

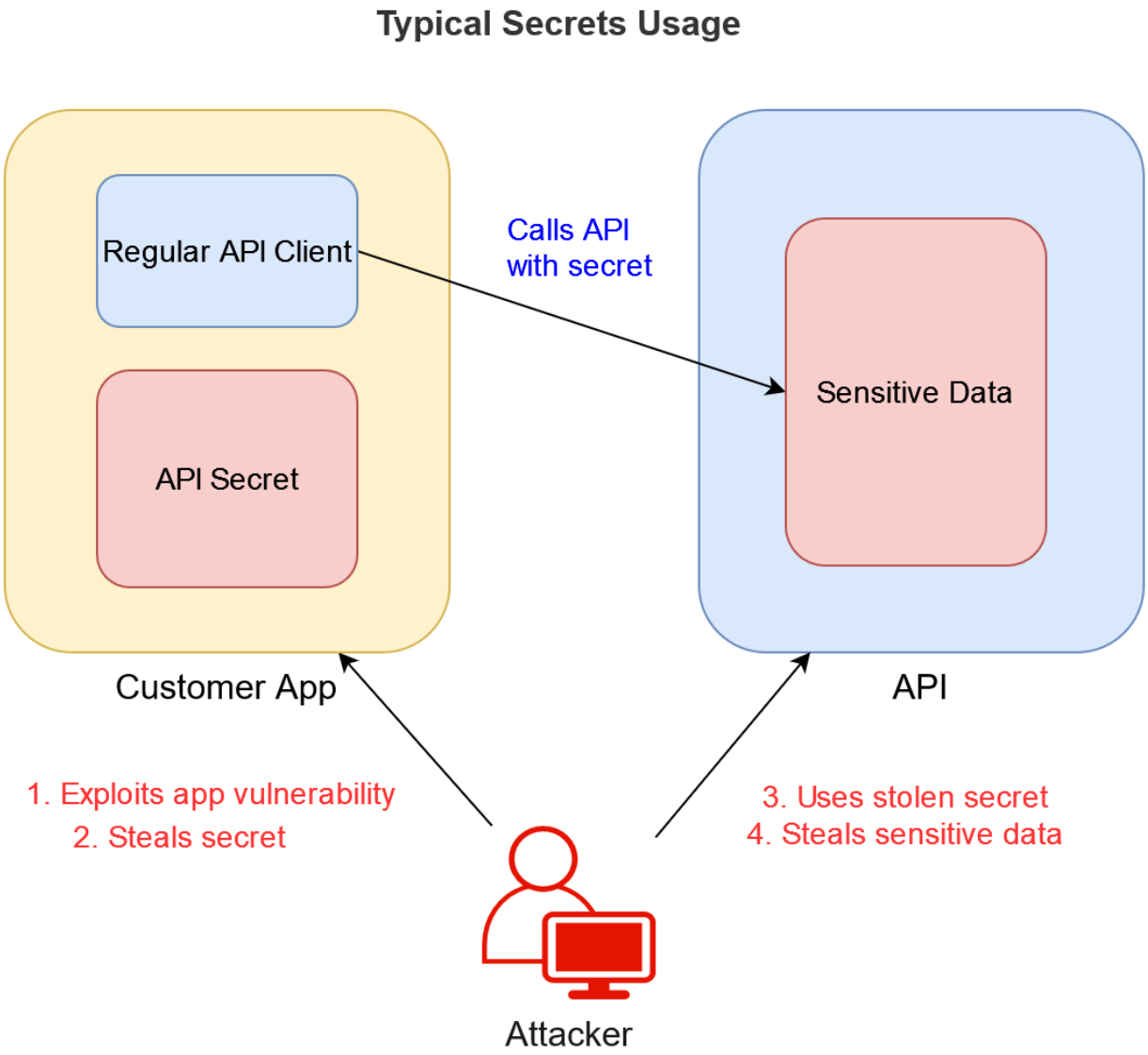
In the pages that follow, we will elucidate how PolyAPI leverages innovative techniques and robust methodologies to fortify API security, providing peace of mind to businesses, their clients, and their stakeholders. Including things such as credential management, detection / response mechanisms, and access control, PolyAPI's capabilities offer a comprehensive approach to safeguarding your use of APIs.

By the end of this whitepaper, you will have a profound understanding of how PolyAPI empowers organizations to embrace the API-driven future with confidence and resilience, securing the vital links that fuel the digital economy.

# Credential Management

## The Risk of Leaked API Secrets

While application owners should do everything in their power to prevent vulnerabilities, it is unreasonable to expect that it will always be possible to produce a product with no security related defects. Thus, our threat model should assume that at some point a breach is likely to occur. Depending on the severity of the vulnerability involved, this could result in the attacker gaining full control of the application. This attack scenario is our starting point. From here, the attacker steals the secret and uses it for nefarious purposes.




## Introducing Vari: Reinventing the Security of Secrets

Poly takes an innovative approach designed to protect sensitive API secrets, even in the face of a cyber attack. Unlike traditional applications, which access secrets through environment variables, a config file, or a secrets management service, Poly manages secrets on behalf of the application using the Vari service and allows API clients to use those secrets by reference, rather than by value.

Some readers may notice a parallel between this and the approach that some organizations already take to protect sensitive PII data via tokenization. In the same vein, Poly tokenizes credentials.

## How Poly Safeguards Your API Secrets

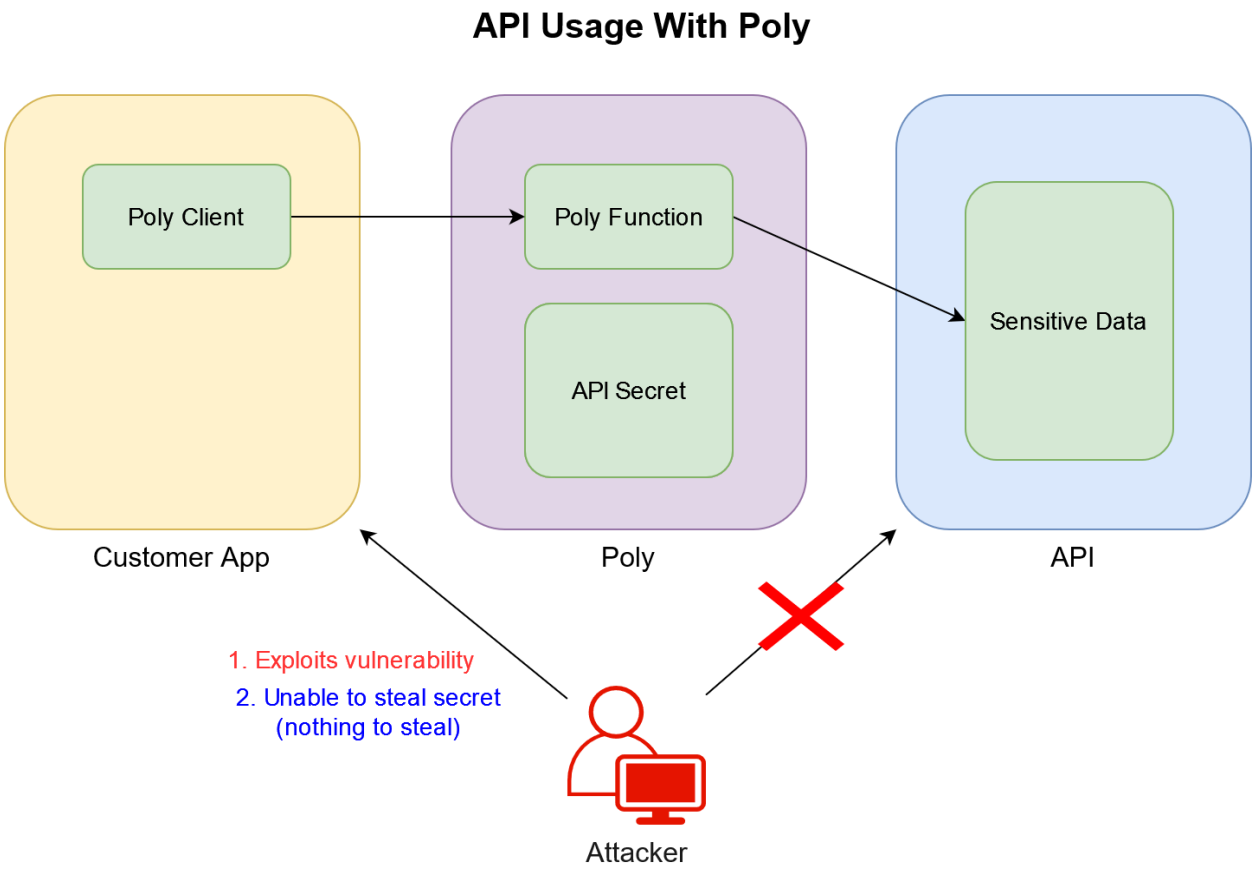
When your application uses Poly as its universal API client, the API secrets there were previously available are no longer directly accessible by the application. Instead, Poly acts as a secure intermediary between your application and the third-party services. Here's how it works:

1. **Poly API Client Integration**: Your application integrates the Poly API client, replacing the traditional API client it was using previously.

2. **Request Forwarding to Poly Server**: When your application needs to make an API request it calls the corresponding API function on the Poly server.

3. **API Secret Addition**: The Poly server securely adds the required API secret to the request before forwarding it to the respective API provider. This can either be invoked manually via code or automatically via configuration on the Poly server.

4. **Response Handling**: The response from the API provider is sent back to the Poly server, which then returns it to your application.



## The Crucial Difference: Zero Access to API Secrets

The most crucial aspect of Poly's security lies in the fact that, at no point, are the API secrets accessible by your application. By removing the secrets from the application altogether, Poly ensures that even if your application falls victim to a cyber attack, the attackers won't be able to retrieve these secrets and, subsequently, won't have unauthorized access to your API provider.

This design follows the established security principle of isolation and compartmentalization. Just like a submarine can withstand an attack by sealing off flooded compartments, Poly reduces the blast radius of a successful hack by denying access to protected secrets.

Because of this, Poly can be used as part of a greater defense in depth security strategy. While a number of preventative controls can be applied to prevent a successful attack in the first place, Poly serves as a second line of defense in case the initial controls fail.

## Self-service Credential Provisioning via Vari

Imagine your organization wants to create an integration for several of your customers to use. The usual procedure for this looks like the following:

1. Create the integration and present it to your customers
2. Customers approve and send their API credentials to your ops team
3. Your ops team adds your customers' API credentials to the integration configuration
4. Integration is now ready to use

Now reimagine this scenario with Poly:

1. Create the integration and present it to your customers
2. Customers approve and have *their own* ops team add their API credentials to the vault
3. Integration is now ready to use

With self-service credential provisioning you can enable your customers to manage and operate their integrations without any contact or credential exchange with your ops team, eliminating unnecessary friction and the need to expose sensitive credentials.

Lastly, imagine a case where your customer needs to rotate their credentials, either because of a security incident or a regular credential rotation policy. In a typical integration, they must once again contact your ops team, transfer the new credentials, and wait until the process is complete. With Poly, rotating credentials is just as seamless as the initial provisioning process, everything can be taken care of by the customer

themselves and the Poly operator is no longer the bottleneck. This represents a major evolution from business as usual.

# Simplifying & Securing OAuth

When it comes to accessing API providers, OAuth is one of the most popular protocols for obtaining access tokens, and thus it is extremely important from a security perspective. Unfortunately, setting up and using OAuth is a consistent pain point for many developers. Poly makes it easier for developers to deal with OAuth by simplifying the process of getting and handling tokens. Not only is this more convenient from a developer experience perspective, it is also a more secure one.

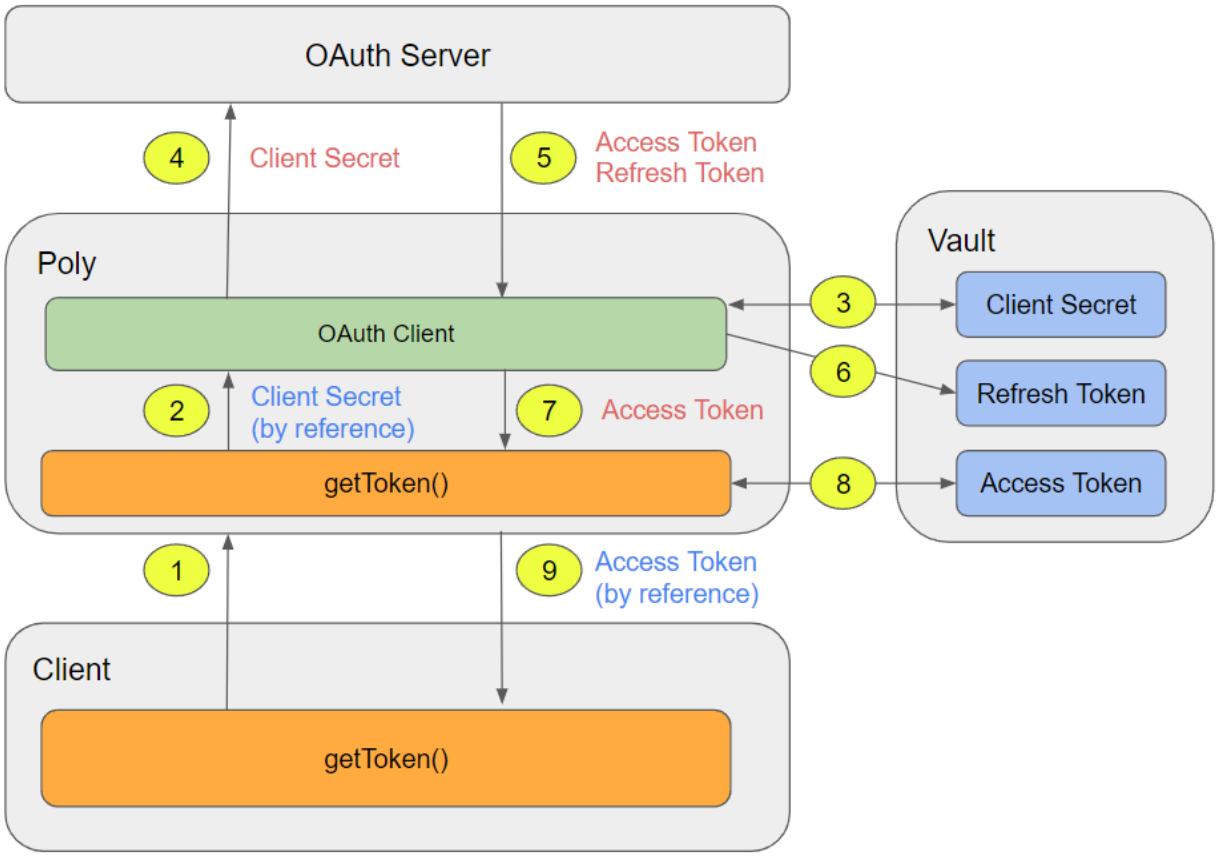
## Initial Configuration

Setting up an OAuth client is straightforward, there is no code to write and all OAuth settings are intuitively mapped to a config that is updated via API. Once the client is configured, there is also an additional brief step which involves creating a server-side custom function which will handle calling the OAuth client, injecting the necessary client credentials, and finally handling the response. An example code snippet can be requested from Poly to use as a template.

## Operation

After the initial configuration, retrieving the token becomes as easy as calling a single function:

1. Server side custom function getToken() called by API client
2. Custom function calls OAuth client and passes client secret by reference
3. OAuth client retrieves OAuth client secret from Vault
4. Client secret is sent to OAuth Server
5. OAuth Server responds with access token and refresh token
6. Refresh token stored in Vault
7. Access token returned to custom function
8. Custom function stores access token in Vault and a reference is returned to the custom function
9. Access token reference is returned to the client




Using the token is just as simple. There isn't even any need to worry about manually refreshing it as the custom function that was setup previously can handle that logic all on its own:

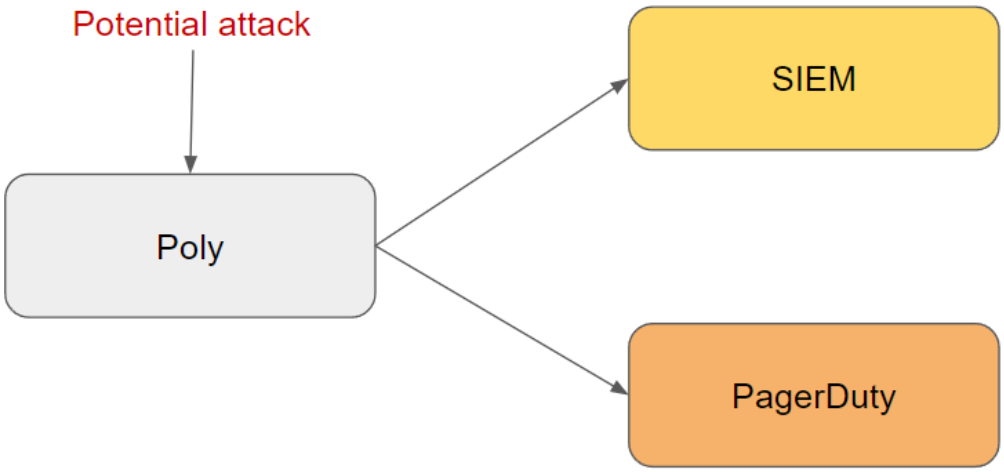
1. API client calls the getToken() custom function
2. Custom function checks if token is expired
3. If token is expired, a new one is fetched
4. A token by reference is returned to the client
5. Client calls API function with access token reference
6. API function retrieves access token from Vault
7. API function calls the API provider with the access token
8. API provider returns response to Poly
9. Poly returns the response to the API client

# Event Driven Architecture & Error Handling

Poly has an event driven architecture which emits an event any time an error occurs. In concert with this, it supports the implementation of custom error handlers which can be scoped to specific functions. This provides the system operator with a number of potential benefits, primarily, alerting and responding.

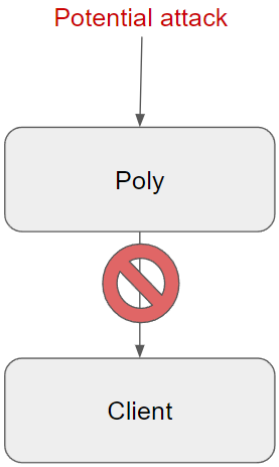
## Integration with Security Information and Event Management (SIEM) & Real-Time Ops Platforms

For instance, if a particular type of error is triggered, it could signal that an attack on the system is occurring. The error handler can respond by raising an alert, either in a Security Information and Event Management (SIEM) system or in a real-time operations system such as Pagerduty.



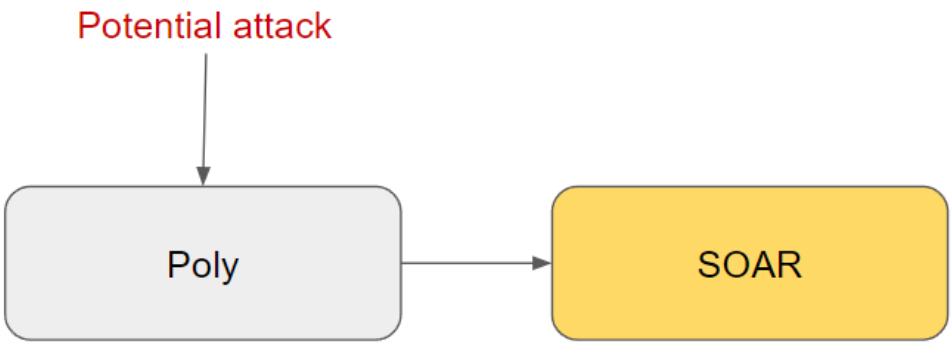

## Proactive Response

Beyond simple reporting and alerting, Poly can actually take proactive measures to handle whatever is occurring at the time in the system. This includes things such as blocking a particular state change in the system, or even rolling back a state change in the system which has already occurred. Being able to take action within the platform itself is useful in the sense that it happens in real-time and there is more context available, thus enabling a more intelligent and robust response.



## Integration With Security Orchestration, Automation, and Response (SOAR) Platforms

Some teams may already have complex incident response automations setup via a Security Orchestration, Automation, and Response (SOAR) platform. In this case, Poly can call out to these systems and provide the context necessary to kick off any number of sophisticated flows. The close integration of these two platforms can provide a response capability that isn't possible to achieve with only one platform alone.



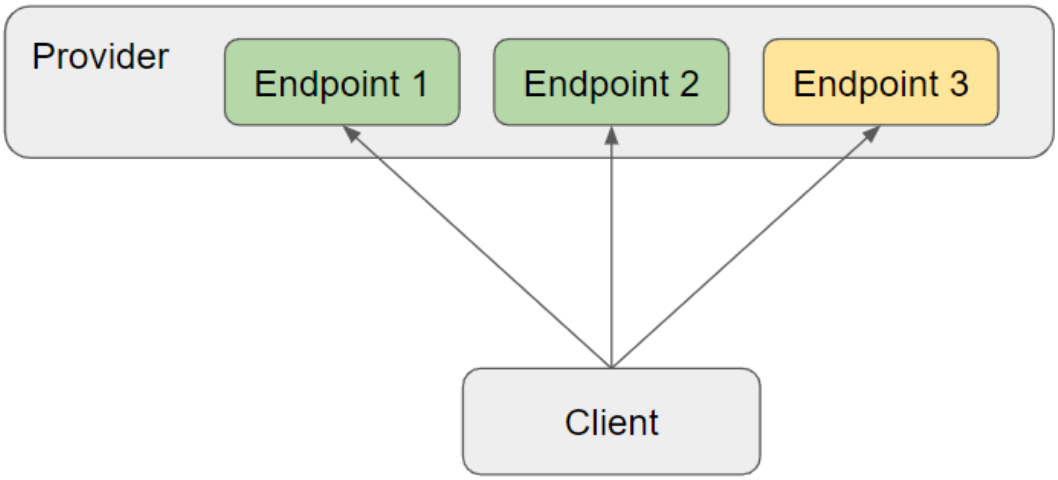

Centralizing this type of logic means that it won't need to be applied to each individual client. Those who have built and implemented security controls before understand that half of the battle involves not just the development of the control, but the consistent application of it.

## Additional Benefits

Lastly, there are a number of non-security related benefits as well. Not all errors and events signify an attack, some might simply be indicators of a benign system misconfiguration or unexpectedly altered system state. From an operational perspective, this information and the ability to respond to it is useful beyond the scope of just security.

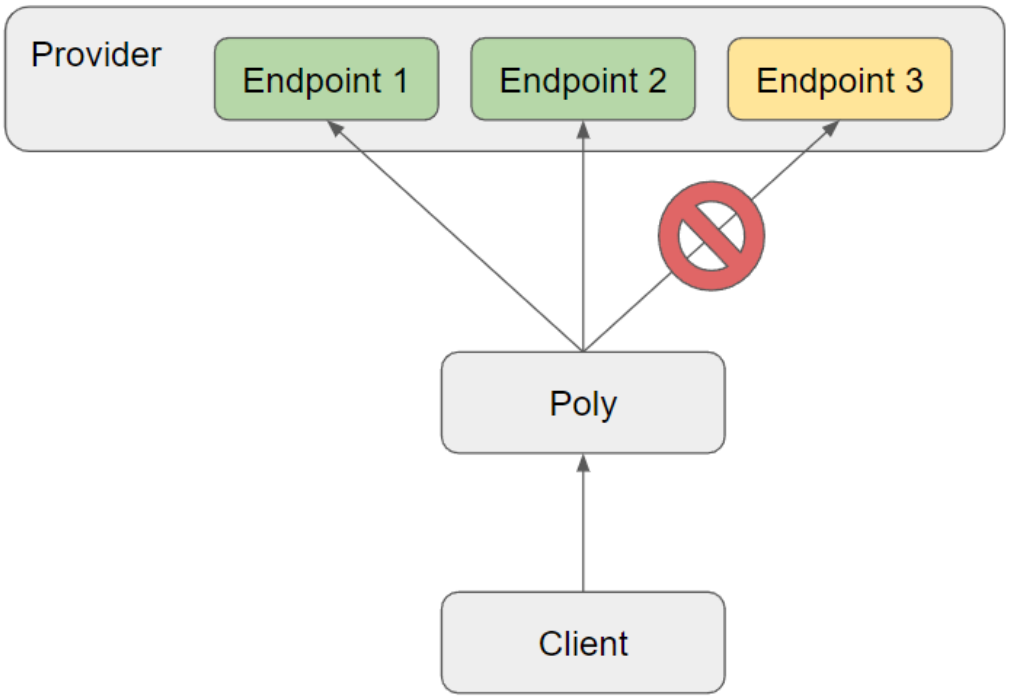
# API Shaping & Scoping

Access control is a critical aspect of any system that deals with sensitive data or functionality. Traditionally, the level of access provided to an API client was largely derived from the scope of the token. While this is effective to a degree, the range of scopes offered by the API provider may not be sufficiently granular to restrict access in the exact way desired.




Multiple clients present an additional challenge. The presence of multiple clients implies that they likely serve different functions, and thus require different scopes. However, in the case that the API provider only allows the consumer to register a single client per tenant, the set of allowed scopes in the client configuration must become a combination of all the scopes necessary for each client. In such a scenario, a well-intentioned developer may inadvertently request a token which contains scopes that are not essential to the use case at hand. This type of error violates the security principle of least privilege and creates new risks for both the consumer and provider involved.

Poly facilitates least privilege with its capability to shape and scope access to APIs without relying on control mechanisms from the provider. Instead, a tenant can selectively train certain functions, thus restricting access to only a subset of the total endpoints available.




## Example Scenario

Suppose an organization wants to enable better integration with their ERP platform. They begin by identifying which modules they want to expose. Next, Poly functions are created for those and only those modules.

Let's say that in our example the organization wants to enable access to the customer relationship management module, but not the manufacturing and warehouse management modules. Poly can accomplish this goal with ease.

## Reduced Attack Surface

Since ERP platforms can have a massive amount of modules and capabilities, allowing your applications to access them without restrictions exposes a significant attack surface. Likewise, using Poly to restrict API access greatly reduces this attack surface.

For instance, a module that allows uploading and downloading of files may become a primary vector for spreading malware. Poly can block your client's access to such high risk modules while still facilitating the normal traffic needed to run the business. This way, if an application in your own system is somehow compromised, you will still be able to prevent the spread of the attack and protect your partner ecosystem because the blast radius of the attack is significantly reduced. Partners which understand that you are capable of applying such controls are likely to perceive your security posture as being

mature, thus enabling more trust between your organizations, and ultimately, more business.

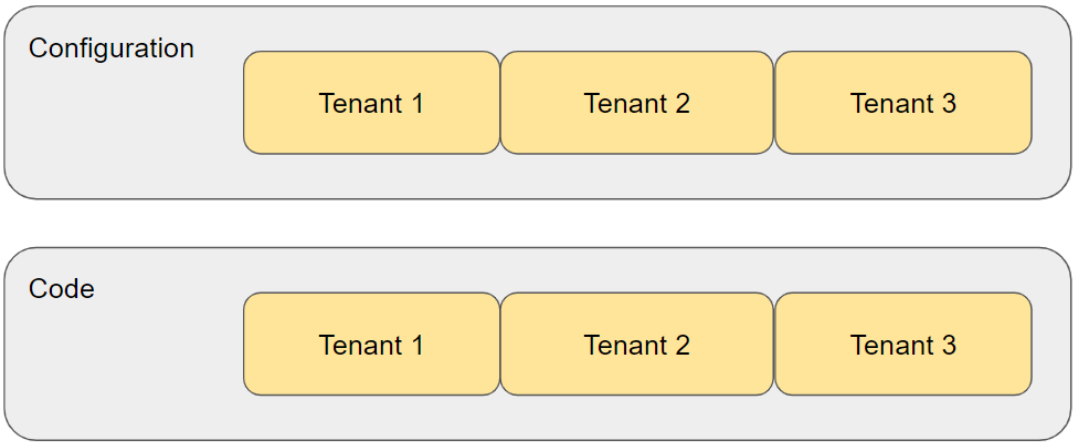
## Least Privilege In Action

The main idea is simple: restricting access to certain APIs can greatly reduce the potential for both accidental misuse and deliberate abuse. The more challenging part is actually implementing the idea in a reliable fashion. Poly can accomplish this effectively because centralizing traffic makes the traffic easier to control.

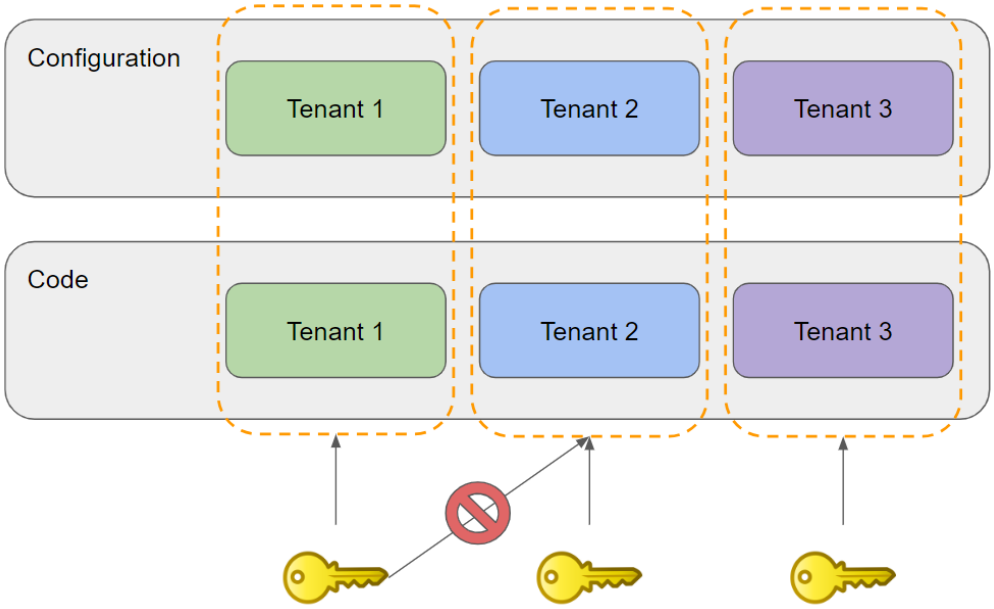
# True Multi-Tenancy via Secure Design

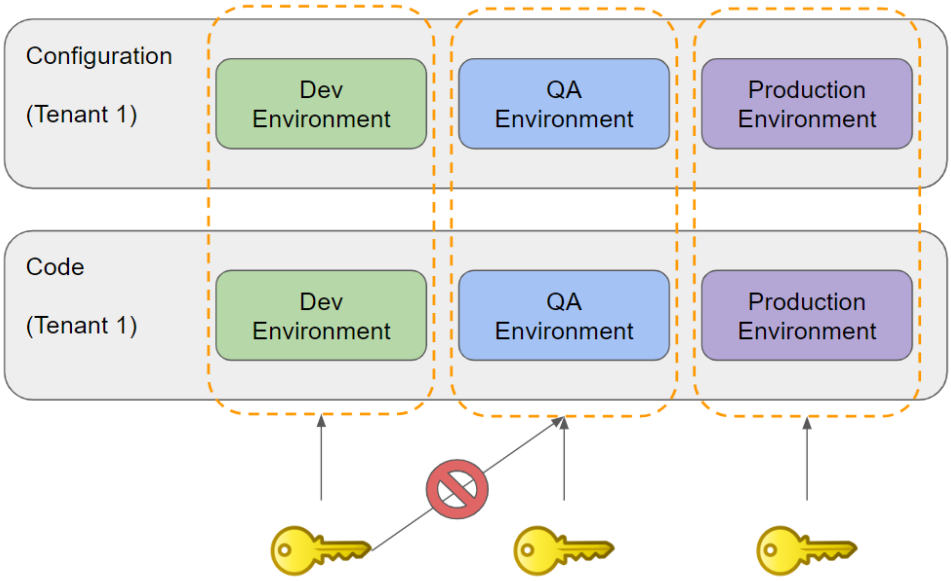
Poly's architecture is designed to foster true multi-tenancy, a feature that sets it apart from other integration platforms in the industry. Many of these platforms have notable deficiencies in this specific area. Their access controls often lack the robustness and flexibility required to prevent integration developers from inadvertently misconfiguring a tenant they did not intend to. This deficiency results in a troubling lack of separation in data, configuration, and functionality, which can lead to numerous issues.

A common scenario encountered on such platforms involves companies which specialize in creating and maintaining integrations for their customers. The goal is to use a common flow or integration to service multiple customers to avoid having to instantiate a set of flow or application per customer. In this case the flows or applications should be completely logically isolated from one another. Unfortunately, due to the nature of these platforms, they essentially end up working with a single configuration/secrets table or file which contains the credentials/configuration for all customers. This practice constitutes a significant anti-pattern in the integration world, hindering not only efficiency and maintainability, but security as well.



Poly, in contrast, prioritizes security through thoughtful design and architecture. It accomplishes this by employing unique API keys for each tenant and every environment within that tenant. Each instance of a custom function being executed (akin to a flow in other iPaaS platforms) is done within the context of a single environment (customer). Credentials and configurations are logically separated per customer while the code executing the logic can still be written once and shared. This approach eliminates the potential for integrators to make mistakes, such as applying the wrong configuration to a customer's integration. Poly's commitment to these principles ensures that data and configurations remain secure and separate, offering a superior multi-tenant integration experience.



# Conclusion

We encourage you to take the next step towards safeguarding your API infrastructure by delving deeper into the capabilities of the PolyAPI. Now is the time to take action and ensure the resilience of your API ecosystem. To learn more about how PolyAPI can benefit your organization and to experience its capabilities firsthand, we invite you to explore a risk-free demo of our product.

Don't wait; get started today by visiting our website, where you can access more in-depth resources, try out the platform, and engage with our experts. PolyAPI is your trusted partner in securing your API usage and fostering a safer, more interconnected world. Join us in this transformative journey and embrace the future with confidence.