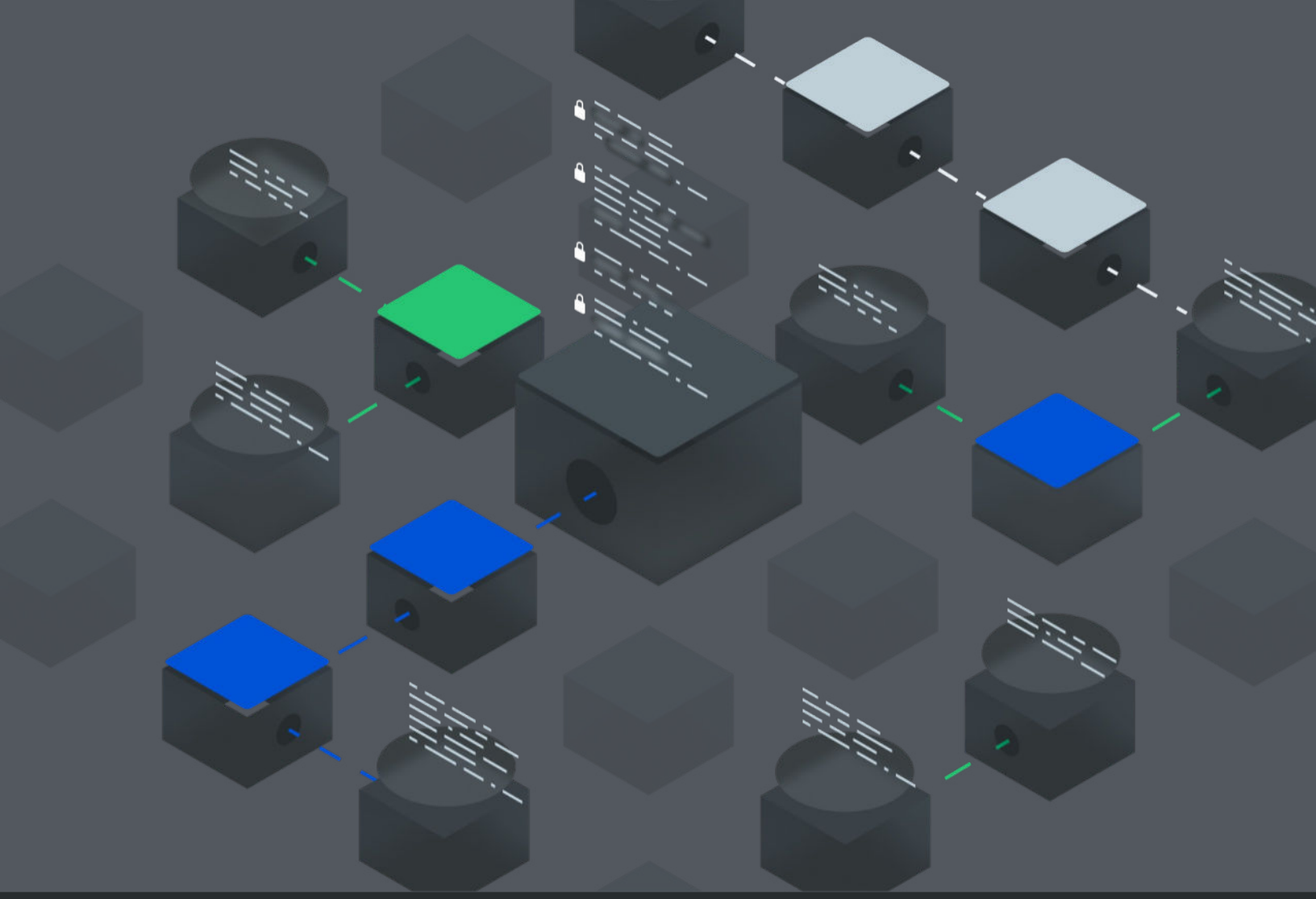


Striking a proper balance: Why comprehensive API security requires both agentless and agent-based data collection

TRACEABLE

Engineers have long-devised various ways to collect reams of data from across their ever-expanding IT infrastructure operations. This treasure-trove of insights – buried deep within systems and software – helps IT operators improve such critical digital business requirements as network throughput, application performance, and increasingly security.

Along a spectrum of data gathering options, agentless observability quickly analyzes data from network traffic to create a simple-but-general picture of events. Elsewhere on the spectrum, agents-based gathering methods have evolved and matured to listen for and record deeper and wider to ascertain more data-driven insights and analysis.

A similar spectrum of observability choices and trade-offs is playing out in the application programming interface (API) security market. An assortment of security tools and platforms rely on a variety of agentless and/or agent-based detection and recording methods. On the one hand, agentless methods allow a quick and simple start. On the other hand, agent-based methods satisfy a growing roster of security requirements by deploying proven agents where appropriate to improve security analysis depth and ultimately reduce time to remediation.

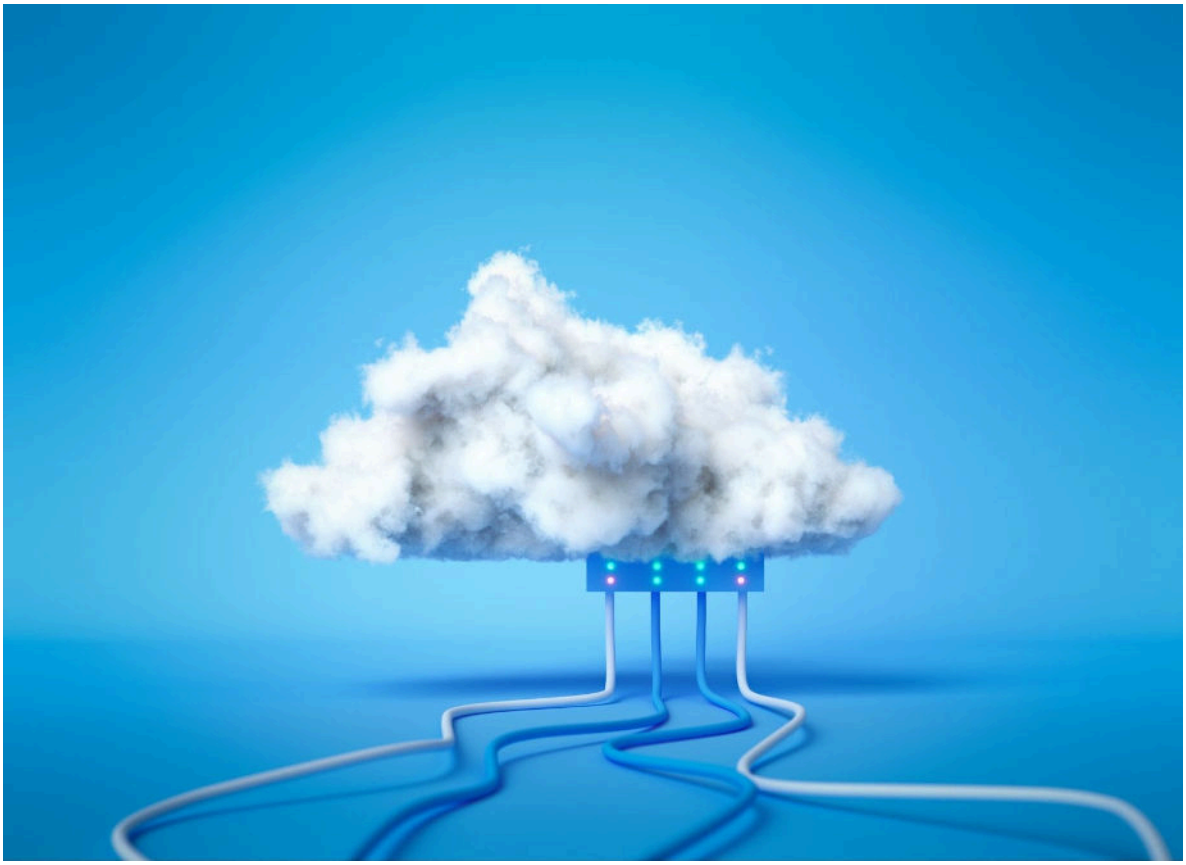
New threats demand **new tools**

Traditional IT security infrastructure was built and optimized for older, monolithic applications and on-premises data centers. Over the last decade, enterprises have rushed to adopt development environments that split functionality across a collection of smaller microservices often hosted in many clouds or distributed data centers. This agility allows companies to build with constellations of assembled services to rapidly create new business functions, as well as to improve their services via rapid and iterative code changes across separate teams of developers.

These different services and applications are often and increasingly interconnected using APIs. The API-driven approach allows microservices to be developed, scaled, and deployed independently, often using containers and cloud-native services. The trade-off is that the expanding use of interconnected web services protocols and integration methods has also created an expanding array of API security risks. Furthermore, application security tools and techniques that worked for older monolithic applications and systems do not satisfy the extended enterprise security requirements of widely dispersed, cloud-native API and microservices architectures.

Vulnerabilities creep into **distributed services**

A new approach to security is clearly necessary to address the current and next generation of vulnerabilities and attacks. The distributed nature of how today's applications operate alone demands new ways of tracking the services and data they use and share. For example, the business logic once embedded in a monolithic application is now strewn across hundreds – or possibly thousands – of microservices. One software change in a single microservice can create a vulnerability and expose the entire application to potential exploits. Critical API vulnerabilities such as Broken Object Level Authorization (BOLA) and Mass Assignment are challenging businesses and security teams to find better defenses than traditional web security approaches.



Modern API security tools take a different approach to understanding how applications work and change over time. They construct a threat model gathered from across the services that form the application to better understand the vulnerabilities and business flaws that malicious cybercriminals could exploit. These tools collect metadata from across the communication flows and services to build a robust model for detecting new API security vulnerabilities. These machine learning (ML) models can then learn how the application "should" work and operate by normal users. Deviations from normal baseline behavior can then be surfaced quickly and flagged as malicious.

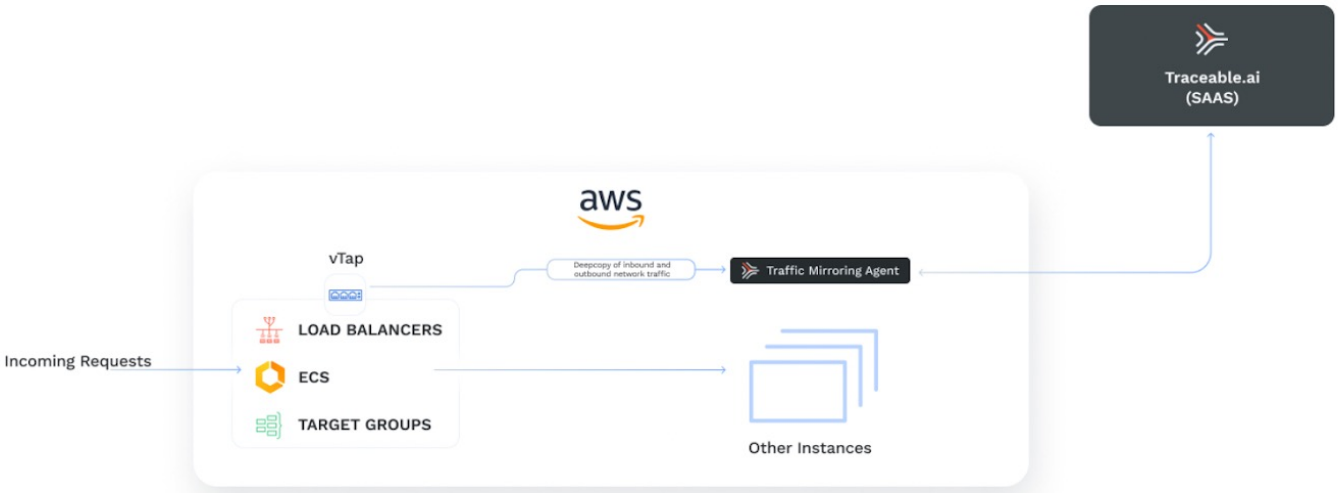
Different approaches to **gather data**

Agentless deployments

An agentless security approach gathers data without a service, daemon, or process running within the application, application component, or microservice. Agentless security deployments sit outside of the application and reduce administrative overhead and cost. There are two flavors of agentless: traffic mirroring (or just mirroring), and edge.

Mirroring

In the mirroring cases, agentless deployments involve configuring existing cloud or Kubernetes infrastructure to mirror specific types of traffic to the API security service for analysis. This approach is out of band of any API traffic, which is considered safer since the API security tool is not in a position to interfere with, alter, or slow down the application traffic.



Traffic mirroring for out-of-band data collection

Agentless mirroring options commonly support mirroring from cloud services running on Google Cloud Platform, Microsoft Azure, and AWS cloud infrastructures. Some agentless options work by configuring Kubernetes DaemonSet mirroring to capture out-of-band traffic for any Kubernetes-orchestrated app running on containers anywhere.

To optimize the collection of the mirrored traffic, agent traffic relays are deployed outside of the application and operate on the mirrored copy of the application traffic. Instead of deploying the agent within the application, the customer can deploy a traffic relay in a separate virtual image that runs on the same virtual private cloud (VPC).

An agentless mirroring approach tends to be easier to deploy and is a good option for enterprises to kick-start their API security practice. It is easy to begin since security teams do not need new permissions from the application development team. The downside is that an agentless mirroring solution can not provide a deep view into the internals of the application as it runs, meaning the API security solution is operating with a reduced understanding of application context.

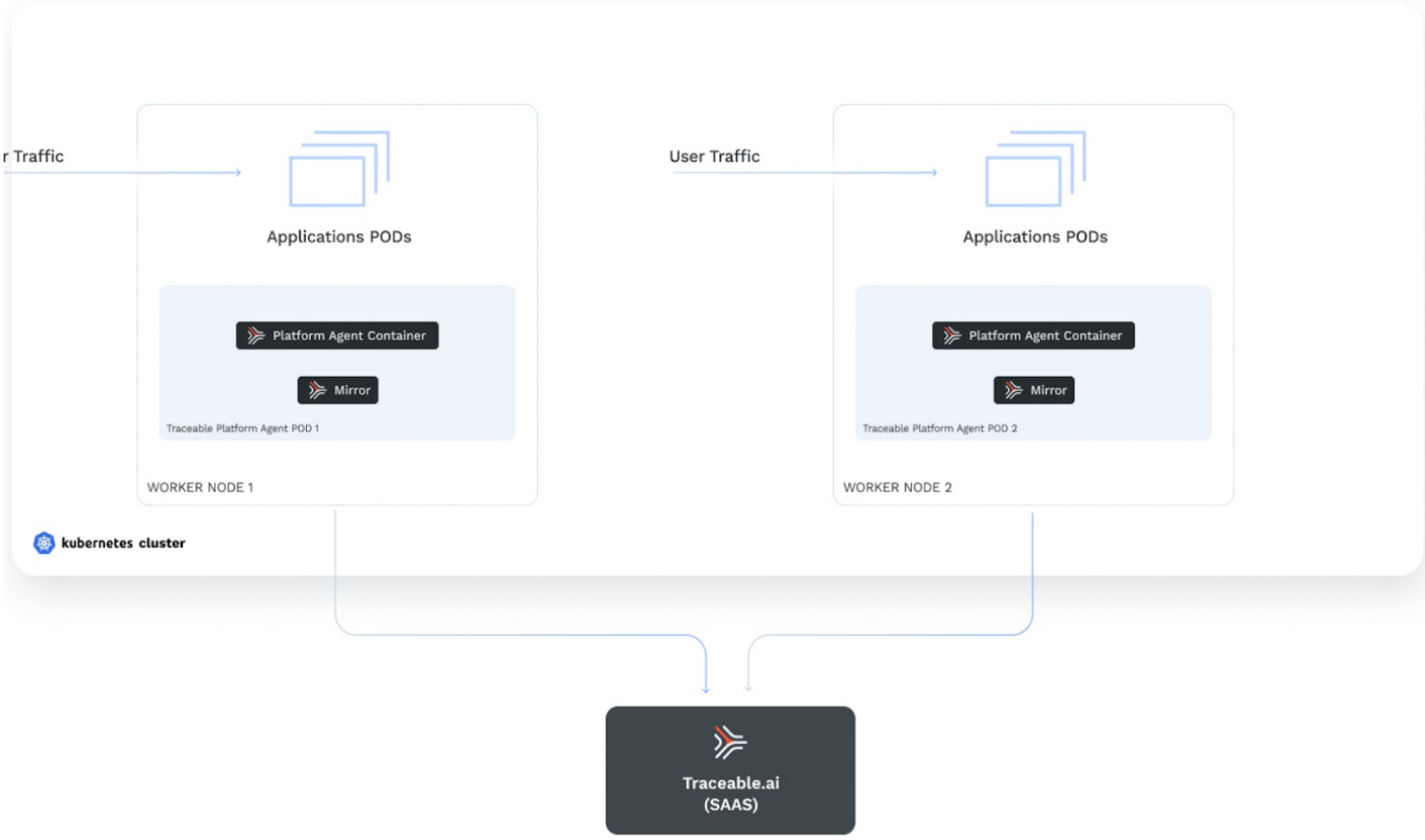
Additionally, the API security solution that relies on agentless cannot directly block malicious traffic, since it is out-of-band, and the decisions are delayed since the analysis is done outside of the path of the application. This universal drawback is applicable to any software using a mirroring approach.

Edge

Some enterprises choose to adopt a hybrid approach in which components are installed on infrastructure in the path of the traffic. This includes modules installed on proxies, API gateways, Kubernetes service meshes, or Kubernetes containers or pods. In this case, it is still technically agentless since no code is installed on the application or service itself. This approach requires slightly more collaboration with the operations team, but it can gather finer-grained information and can directly block some malicious traffic that passes through the infrastructure it is on, such as a proxy or gateway. However, it can not directly block traffic between the services that sit behind the gateway (ie, east-west traffic).

Agent-based deployments

Further along the API security data collection spectrum of choices is the agent-based approach. It integrates agent code into the application runtime. Like the software agents that have become a mainstay of application performance monitoring tools for years, this approach provides a more granular view of API traffic, data flows, and code paths.



In this deployment model, an agent (a small executable) sits within or alongside applications that need to be secured. Such agent code inserts, for example, into Java, Golang, Python, and Node.js. In addition, agent-based deployments can support serverless installations on AWS Lambda serverless cloud services built using languages such as Python and Node.js.

Regardless of the environment and deployment model, such agent-based security solutions collect a plethora of granular data on many variables, including the request/response full header and body payloads, performance characteristics, infrastructure availability, resource consumption, and the operating systems' and applications' behaviors.

Agents and advantages

There are several advantages of the agent-based approach. One is that it can provide deeper security analytics and protection by providing that rich bevy of runtime data, payload, and user behavior over time. Agent-based solutions can also intercept and block requests between services at wire speed, and – in cases of encrypted traffic – can still detect issues because they sit outside of the encryption path. An additional advantage of an agent-based approach is that for Log4j- and Spring4shell-type attacks, where libraries being called have high criticality recently known vulnerabilities, in-app agents provide surgically precise protection by blocking API calls to the vulnerable libraries only and exactly at the point in the app where they are being called from.

Upfront there may be the need for added effort with agents. Agent-based approaches can require more coordination with different teams to properly deploy the agents onto production systems. But while the agents do sit in-line with the application, added latencies and additional risk from modern agents have proven negligible when compared to the value of the extended security data and insights gained.

Evolve to an **API security strategy**

There are clear tradeoffs with implementing either agents or agentless observability deployment options. These range from the ease of deployment that enables quicker time-to-value with an agentless approach to a set of powerful security capabilities when an organization fully implements an end-to-end agent-based security solution.

The reality is that organizations should not and do not need to make such an either-or decision.

Traceable AI delivers a solution providing the most flexible deployment options for both agentless and agent-based deployments. Organizations do not have to choose one method over another, especially in order to gain highly capable API security rapidly. Traceable AI offers agentless deployments using mirroring, as well as across such edge options as API gateways, proxies, load balancers, and Kubernetes meshes. Concurrently, Traceable delivers agent-based options for Java, Python, GoLang, and Node.js, on both server and serverless installations. And with the agent-based deployments, users gain advanced security capabilities including end-to-end sensitive data flow analysis and user-attributed, end-to-end cross-session activity tracing.

Every organization has a range of applications and security requirements – as well as a variety of stakeholders – that evolve and come and go over time. A common approach adopted by many organizations is to implement a quick-time-to-value strategy through an agentless deployment, learning what security features are available, and educating cross-functional teams on the range of available security options and advantages for each deployment option.

As various groups obtain value from an agentless deployment, they can begin to explore deeper security capabilities that can be rolled out over time in strategic steps. This approach helps security and development teams properly prepare for in-app agents to obtain the capabilities, insights, and visibility not available with an agentless only deployment. These choices can also grow and adapt based on their cloud, hybrid, or on-premises deployment model decisions for their various applications and data services.

Establish priority of security-first for **observability**

In conclusion, a spectrum of choices and trade-offs is playing out in the API security space when it comes to how to gather and use data about apps, systems, and services. The most important decision, at the end of the day, is on how best to make APIs known, monitored, secure, and easy to remediate on an ongoing basis. The choices about API security should not be made about the agents' requirements alone, but in the context of the end-to-end security needs.

TRACEABLE_

Traceable AI offers a full range of both agentless and agents-based observability options to provide the best and broadest means to gather all the data about APIs in production to protect them. The agents are the starting point of a data-driven journey to assemble an API security baseline by which to monitor, protect, and improve the full API ecosystem across their full lifecycle for any business, for any IT environment.

Depending on your role and the needs at your organization, there are multiple options to get started with Traceable AI and its many options for observability and API security:

- If you're a CISO or DevSecOps security leader and want to evaluate your API security risks, try the API Security Posture Assessment.
- To start your journey, sign up for a Free Tier and learn all about your APIs — internal, external, third-party, and even the shadow or rogue APIs you may not even be aware of.
- If you want to compare different API security solutions in the market, check out the API Security Tools comparison guide.
- You can also view a demo or book a meeting to learn more and ask your questions on how Traceable can meet your API observability and security requirements.