gravitee.io

API Security:

why you should care, addressing the challenges, and building an effective strategy

Table of Contents

2 **API-First**

3 **The Challenges of API Security**

The API Lifecycle
Access Control
Inventory Control
Understanding and Prioritizing Risk
Recognizing and Addressing Attack Vectors

7 **The Current Solutions and Their Shortcomings**

OAuth2 and OpenID Connect
Web Application Firewalls and Gateways
Protection for Synchronous, but Not Asynchronous, APIs
Search and Audit

9 **An API Security Strategy that Works**

Minimalistic Practices
Strong and Adaptive Access Management
Dashboards, Discoverability, and Audits
Think Like an Attacker
API Lifecycle Management
Dealing with the Moving Target of API Threats

13 **The Gravitee Solution**

API-first

Today's software teams are adopting an API-first model for development. Modular API development is the model of choice for some of the largest organizations in the world, used for both their internal operations and their customer-facing applications. In fact, at least 83% of all internet traffic today is driven by APIs.

APIs have been one of the accelerators of agile development, and the industry has benefited from the high velocity of updates and the simpler maintenance of smaller, decoupled components. In terms of security, however, the progress of API development has brought with it new challenges and new attack surfaces. Gartner actually predicted that, by 2022, APIs would become the single greatest attack vector for causing data breaches within web applications, and, according to more industry research, "The risk is real – 95% of respondents have suffered an API security incident in the past 12 months, 21% admit they've been the victim of an API breach."

Clearly, as you scale out your API strategy, security needs to be built in. Understanding the potential threats and taking the appropriate security measures helps protect your company as well as the clients using your APIs.

In this whitepaper, we'll look at the API security from the following angles:

- The challenges of API security
- The current solutions and their shortcomings
- An API security strategy that works

The Challenges of API Security

The challenges of securing APIs come from several different directions, Let's consider each of these in detail.

The API lifecycle

While many security vulnerabilities are exploited after an API has been published, the introduction of those security risks likely occurred early in the lifecycle. Secure practices begin with awareness and adoption of an organization's overall security strategy at the very start of the API lifecycle, often at the API Design phase.

However, creating organization-wide security standards is difficult because of the myriad use cases, tools, and teams. Many organizations are simply unable to govern how security is layered into the lifecycle.

Even if a company could implement clear security policies around API development, developers might circumvent those policies. They may delay communicating with the security team or bypass an intentional security chokepoint for expediency.

Example: from internal use to public use

Let's consider an API that was intended for internal use only. The original team designed the API only to be used by people on that team. All of the users were known within the company, so it seemed unnecessary to build robust security and authentication for the API.

Another team within the organization finds out about the API and requests to use it for some of their internal operations. The original developers agree, sharing the API with this other group that, frankly, they don't know as well. However, they're all in the same organization, so no harm done.

Not long after, teams across the entire company begin using the API. Eventually, the API is opened up for use by close corporate partners. Before you know it, an API that was originally developed for single-team internal use becomes a public, customer-facing API.

Finally, at this point, the original team begins to consider security best practices. But it's too late now. The risk exposure has already been extensive. Security by confinement—which obviated the need for practices like strong authentication, levels of isolation, and role-based access control—can no longer be assumed. And, the team is too far past the API design phase to incorporate these measures simply and effectively.

Other threats within the lifecycle

Furthermore, several insider threats can make your APIs and API development vulnerable. Some insider threats may be unintentional, such as malware installed on a developer's computer, resulting in the exfiltration of data or the injection of vulnerabilities into the API code in development.

Other insider threats may be overtly intentional, as in the case of an employee inserting backdoors into the code that they can later exploit to manipulate the behavior of the API.

Finally, insecure coding practices or human error might lead to the introduction of security vulnerabilities during API development. Bugs—including those related to security—are inevitable, no matter how careful the engineers are. Bugs can be minimized with thorough code review and robust testing, but some errors will always slip through.

Access Control

Most security breaches are not targeted at your application code base, but rather involve user account attacks such as credential theft, leakage, and phishing that provide attackers with a broad range of access to victim accounts and the resources that those accounts have access to. Depending on the level of permissions a compromised account might have in your organization, this kind of attack might lead to disastrous consequences.

While Access Management has become somewhat commonplace, we still see many organizations having not adopted methods of more advanced Access Management, such as Multi-factor authentication and advanced auditing, monitoring, and logging capabilities that can help teams know exactly who, when, and how APIs and applications are being accessed.

Inventory Control

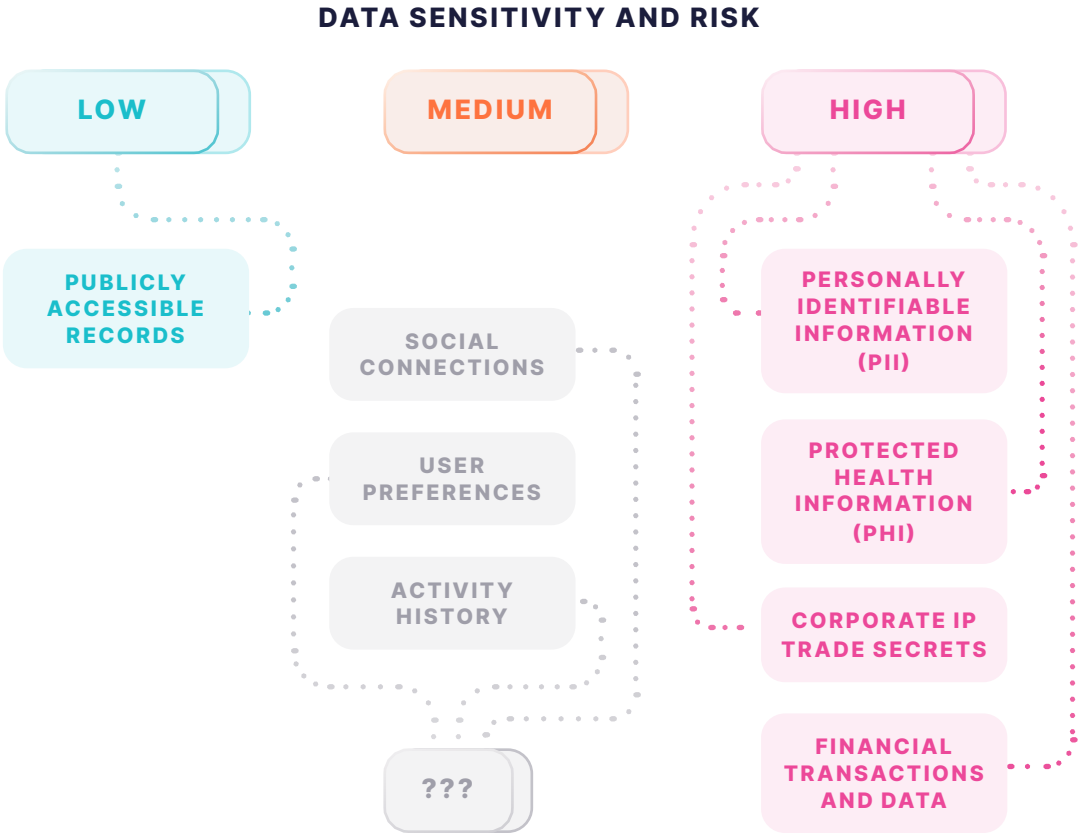
As API development composes a majority share of the software development efforts in an organization, the challenge emerges of managing the ever-growing library of APIs. Many organizations, apart from using automated tools for API discovery, may be unaware of many of the APIs developed and used by their teams. Manual documentation doesn't equate to proper tracking and systematic updating of an API library list. Because documentation is seldom the primary concern of API developers, updating an organization-wide inventory of existing APIs is overlooked or done in a non-standard manner.

Related to inventory control is the difficulty of ensuring that old APIs are sunsetted or replaced with updated versions. Recall from the previous example that APIs may often be distributed to partners or other departments through unofficial channels. As a result, it can be difficult to track down where older versions of an API are being used, replacing those with newer—and more secure—versions.

Understanding and prioritizing risk

When it comes to security, some APIs are more important to protect than others; prioritization often depends on the types of data an API handles and the level of risk associated with that data. APIs that deal with sensitive data such as financial transactions need to receive special attention from the security team. However, many companies are unable to identify which of their APIs fall into these high-risk categories. Without a focus on proper security, APIs that handle sensitive data can be accessed by malicious actors, and that data is at risk of exposure.

Because of this, its important to consider what kind of data that your APIs are handling and categorize them appropriately based on risk. The below diagram might help with this:



Recognizing and addressing attack vectors

Of course, the threats to API security can arise from a variety of outside attack vectors. Security needs to cover risks from all fronts.

Client-side risks

In addition to writing secure API code, protection at the client side includes mitigating issues stemming from shared accounts, stolen credentials, cross-site request forgery, or other risks.

Server-side risks

SQL injections, improper input validation, and insecure endpoints can all open up the server side of your API to attack. Even though software engineers should know better, they can still fall victim to phishing and social media attacks which may leak their credentials for accessing code or infrastructure. Finally, poor encryption or the mishandling of secrets increases the vulnerability of an API.

Network risks

Today's API solutions need to take into account not just the software but the entire environment in which the software operates.

Most companies have invested a fair amount of resources towards preventing network risks such as credential theft or denial of service. Still, even with the implementation of different perimeter protection solutions, organizations cannot ignore the potential risks that can come to their APIs through the network.

The Current Solutions and Their Shortcomings

The challenges to API security are many. And while there are some solutions to some challenges, it's important to understand which challenges these solutions actually address, and where these solutions fall short.

OAuth2 and OpenID Connect

In recent years, the trend of API development has contributed to the increasing adoption of frameworks like OAuth2 for authentication and authorization, along with authentication standards like OpenID Connect.

Occasionally, some development teams decide to create a custom authentication solution in house. This is almost always a bad idea. The threat landscape changes too rapidly for custom solutions to be able to handle the security threats as they emerge.

Standards and frameworks are battle-tested and continually updated to stay ahead of threats.

However, offloading some of the access control burden to technologies such as OAuth2 or OpenID Connect (OIDC) does not relieve organizations of further responsibilities. Application secrets and access tokens must be encrypted upon storage and transported across secure channels. Development teams need to ensure that they're using tools correctly, according to best practice guidance. Fortunately, authentication problems make up a small minority of today's security breaches, and this is probably due to the increased adoption of frameworks like OAuth2.

Web Application Firewalls and Gateways

Web application firewalls (WAF) and gateways provide some level of detection and defense, identifying common attacks such as SQL injection, cross-site scripting, and JSON injection. However, one study noted that 95% of respondents reported a security incident in 2021, with 55% of them relying on gateways and 37% relying on WAFs. These tools are limited in their effectiveness, highlighting the need for a more holistic approach to API security.

> *Firewalls are ubiquitous, they are everywhere, we think API gateways are going to become the next core infrastructure like a firewall.*
>
> **– Guy Duncan**
> **CTO, TIDE**

Protection for synchronous, but not asynchronous, APIs

Many of today's security solutions and policy implementations rely on standards that are appropriate for synchronous API protocols, even though many development teams are beginning to build asynchronous APIs. Security needs for asynchronous APIs are more complex than the requirements for synchronous APIs.

To speed up response times, asynchronous APIs handle multiple requests simultaneously using protocols defined in the AsyncAPI spec. While the results are more scalable and efficient software systems, asynchronous APIs rely on dedicated backchannel communication, which can be difficult to secure.

Search and audit

Recently, the market has seen more sophisticated tools that leverage artificial intelligence in order to fix some of the problems with APIs. Agent-based software, documentation projects, and discovery tools can crawl through a software solution or company to detect and categorize APIs. Automated testing and security audits can detect flaws that have emerged in the API development process. Progress has been made towards software that can identify the presence of code snippets from open-source libraries as an aid in detecting older API code.

However, all of these solutions have one thing in common: they are identifying and repairing mistakes of the past rather than putting in place the procedures and processes that would prevent the mistakes in the first place.

An API Security Strategy that Works

So far, we've touched on the need in organizations for a holistic, far-reaching strategy that addresses API security effectively. The challenges of API security are immense, and the traditional solutions currently in play are limited. However, there is a way forward, and it begins with the following tools and guiding principles.

Minimalistic practices

In the past, organizations have adopted the approach of collecting and storing as much data as they could about their users; but in today's environment, having too much data becomes a liability. The most publicized and damaging security breaches involve private customer data. For that reason, modern policies seek a minimalist approach to data collection and openness, and APIs follow that lead:

- Expose only those interfaces which are necessary.
- Collect only the bare minimum data that is necessary.
- Store data only for as long as it is needed.
- Share data only if it's necessary to be shared.
- Grant access only to the systems that each user needs.

Strong and adaptive access management

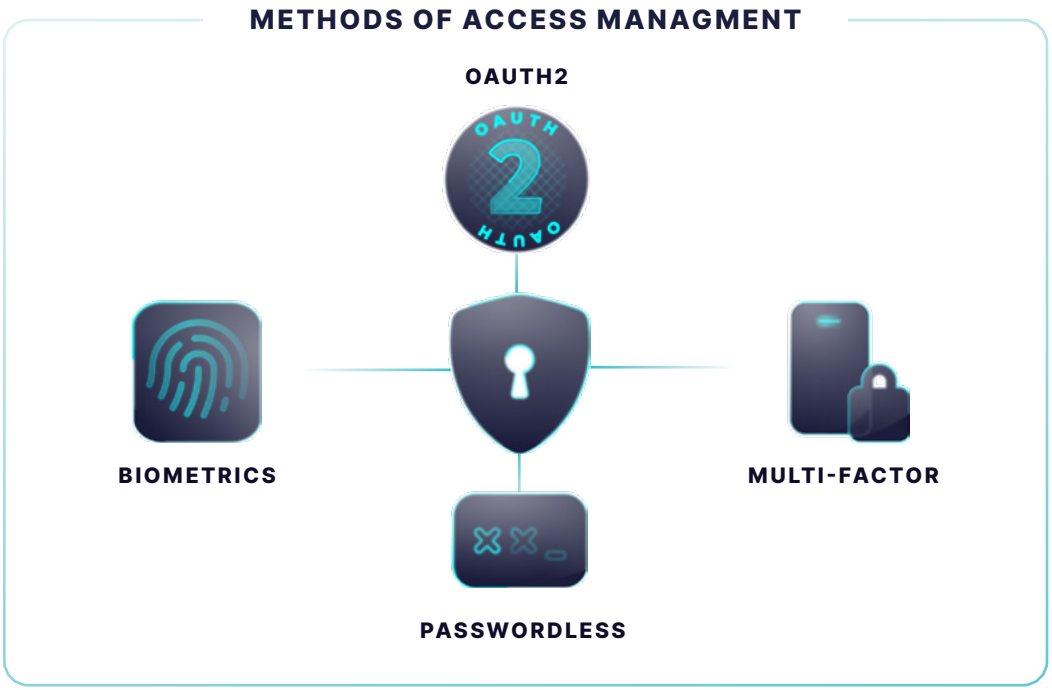
Access management is, of course, integral to API security. As we covered earlier, OAuth2 is the industry standard framework for authentication and authorization. Maintaining the most appropriate and up-to-date OAuth2 implementation continues to be the best practice.

Beyond OAuth2, however, advanced methods of access management are gaining traction. Biometrics, multi-factor authentication, and passwordless logins are all strong measures and are being adopted more widely. The industry is seeing an uptick in the development and deployment of passwordless management solutions, as seen in recent adoption of the FIDO2 standard, which can improve both the user experience and the security of the systems protected.

Adaptive Access Management should also be considered. Adaptive Access Management refers to the process of introducing “decision making” into your Access Control and authentication mechanisms.

Two common mechanisms for Adaptive Access are Adaptive MFA and Step-up authentication. Both mechanisms rely on prompting the user with a MFA method for additional authentication. However, Adaptive MFA dynamically compares user context variables to policy rules such as geo-location or number of log-in attempts. Step-up authentication, on the other hand, only applies MFA if a static rule is met. Such a rule might be: if the user is initiating a payment transaction which requires additional authentication.

Not only does this save time for teams, as they can configure automated adaptation to deal with the variety of users and potential attackers, but it also gives teams confidence that their Access Control mechanisms will be able to adapt in real time to a variety of potential threats.



Identity and Access Management for applications and APIs

One of the questions you might be wondering is: “Okay, I know about Access Management for applications, but why should I adopt Access Management for my APIs as well?”

The answer: because its worthwhile to standardize and centralize how you authorize down to the “last mile.”

By “the last mile,” we mean authenticating all the way down the level of communications between apps and systems, which usually means securing access to API calls themselves and not just securing access to logging into an application.

After all, API’s offer access to data in another interface than the GUI, so making sure to implement the same protocol (i.e. Oauth) for API protection is an easy and secure way of protecting the business.

This tactic also allows for nuanced and fine-grained authentication and Access Control, as you can implement security mechanisms for specific APIs and API calls that deal with sensitive data within an organization and then choose to not implement strict mechanisms for APIs that might not deal with sensitive data. This method is actually called step-up authentication, and is a method that’s employed by many high-performing and strict-security-postured organizations.

Dashboards, discoverability, and audits

Day-to-day management and control of the use of APIs is an important part of security. Several tools today provide dashboarding, alerts, and other relevant information about the use of APIs.

API discovery is not only good business practice—consumers of your APIs need a mechanism for finding them—but it’s a good security practice as well. If a company is not fully aware of the APIs it develops or gives access to, then it is also frightfully unaware of the surface area it exposes to attack.

In addition to discoverability, companies need clear visibility into who their API consumers are and the levels and patterns of usage by those consumers. Auditing, metrics, and dashboards can all aid in the effort to gain insight and visibility.

Think like an attacker

An organization’s security team has a critical role in informing the design and implementation of APIs. Developers should lean on security professionals and other resources to “think like an attacker.” Understanding the mindset of a malicious actor is part and parcel to threat modeling, by which APIs can be designed to hold up against different security threats.

Tactics to help developers think like an attacker include:

- Keeping up with security news, particularly as it relates to cybersecurity. Pay particular attention to others in the same industry and the challenges they face. Vulnerabilities vary depending on the types of applications and data involved, so staying up-to-date on the security breaches in your specific niche will inform your development of more resilient software.
- Engaging in discussions with security and compliance experts. Regulations are constantly changing to maintain people's privacy. While compliance regulation is always a bit behind the curve, knowing the company's regulatory obligations points in the direction of the most common attacks.
- Actively seeking out "horror stories" of security breaches. Talk to coworkers about their worst experiences or the worst breaches in their previous roles.
- Considering how a scriptwriter for "Black Mirror" would write a story about the worst possible scenario for the use of the API.
- Assessing the value of the data being accessed, provided, and stored by the API. For example, financial or critical infrastructure data is more valuable than weather data. Assessing the monetary value of data—either for resale or a ransomware attack—gives you a picture of the measures that potential attackers might take to get their hands on that data.

API lifecycle management

Building into the API lifecycle a clear set of processes for development, testing, and integration is, of course, a key best practice for API security. For these phases of the lifecycle, today's API development tools have a lot to offer.

Beyond the development itself, lifecycle management tools should also help API teams with the following areas:

- Deployments
- Interfaces
- Access
- Consumption monitoring
- Understanding when it's time to sunset an API or update it with other functionality.

One of the great advantages of developing APIs is their modularity, making it simpler for developers to phase out older code. With proper lifecycle management tools in place to handle the deployment of APIs, companies can now plan for API end-of-life or replacement with newer components, rather than leaving loose ends once solutions become obsolete.

Dealing with the moving target of API threats

As with any other area of security, managing API threats is a field that is in constant development. As more companies depend on APIs, they become more popular as targets for malicious actors. The best way to stay updated is by working with vendors and industry leaders who can benchmark API threat development.

The Gravitee Solution

The Gravitee solution provides a single platform to manage, secure, govern, and productize your synchronous and asynchronous APIs.

On top of our API Management and Gateway's access control and threat protection policies such as data logging masking, bot detection, rate limiting, and more, Gravitee offers fully-featured Identity and Access Management and API monitoring and alerting.

Even better, Gravitee's API Security solutions are infused with Adaptive Access, a layer of intelligence that allows your Access Management and Alerting solutions to detect potential threats and implement proper access control and threat protection mechanisms on their own.

If you're interested in exploring Gravitee to solve for your API Security needs, feel free to contact us or go ahead and schedule a demo to see the solution in real time.

How to **Contact Us**

gravitee.io**/contact-us**

If you're interested, and want to reach out, you can contact us here

gravitee.io**/demo**

If you'd like to skip (some of) the Sales pitch and see a demo, you can book one of those here

community.gravitee.io

If you want to give OSS a go, check out our community forum, where you can find links to our github repo and connect with the folks who have driven over 350,000 Docker pulls / month


gravitee.io