Akamai

WHITE PAPER

API Security Fundamentals

Table of Contents

Introduction

API security fundamentals

API security is one of the fastest-growing priorities for security executives. But it's also arguably one of the least understood. The evolution of application programming interfaces (APIs) from implementation detail to strategic enabler of innovation has been a rapid one. As a result, many security teams are scrambling to increase the sophistication of their API security strategies and practices.

APIs are enabling business operations, but they also carry the crown jewels of an organization's data. Even perfect APIs can be abused by hackers, so it's essential to know the fundamentals of API security to protect your business from evolving threats. As more customer interactions and business processes use APIs, enterprise security teams are reworking their security strategies to put API risks at the forefront.

Whether you're looking to touch up on your basics or unsure of what questions to ask, read our guide for everything you need to know about API security threats, trends, and best practices. You'll get an in-depth look at:

- The different types of APIs
- What API security means for businesses today
- Best practices for mitigating API security risks
- Common API attacks and abuse methods

API Basics

What is a web API?

A web API is a programmatic interface consisting of one or more endpoints to a defined request–response message system, typically expressed in JSON or XML, which are publicly exposed via the web — most commonly by means of an HTTP-based web server.

In other words, a web API is what most people think of when they hear "API." It's a collection of endpoints. Endpoints consist of resource paths, the operations that can be performed on these resources, and the definition of the resource data (in JSON, XML, protobuf, or another format).

The term is useful to differentiate web APIs from other APIs, such as those exposed by the operating system or by libraries to applications running on the same machine. But we all understand "APIs" to mean HTTP-based (web) APIs when we talk about the enterprise digital transformation and API security.

What are the most common types of APIs and API terms?

It is helpful for security teams to be familiar with the following terms that refer to different usage models and technology approaches for API implementations. Web APIs are defined as being based on HTTP, and the four main types of web APIs seen today are RESTful, SOAP, GraphQL, and gRPC. The following table defines these four common types, among others.

API Usage Model	Description
Public API	An API that is made available and shared freely with all developers via the internet.
External API	Often used interchangeably with public API, an external API is an API exposed over the internet.
Private API	An API that is implemented with a protected data center or cloud environment for use by trusted developers.
Internal API	Often used interchangeably with private API.
Third-Party API	Provides programmatic access to specialized functionality and/or data from a third-party source for use in an application.
Partner API	A type of third-party API that is made available selectively to authorized business partners.
Authenticated API	An API that is only accessible to developers who have been granted (or threat actors who have gained unauthorized access to) credentials.
Unauthenticated API	An API that can be accessed programmatically without the need for specific credentials.
HTTP API	An API that uses the hypertext transfer protocol as a communication protocol for API calls.

API Usage Model	Description
RESTful AP	Dating back to Roy Fielding's doctoral thesis in 2000, representational state transfer (RESTful) is the most common type of web API, typically using JSON (JavaScript object notation) for the data. RESTful APIs are easy to consume by modern front-end frameworks (e.g., React and React Native) and facilitate web and mobile application development. They became the de facto standard for any web API, including those used for business-to-business.
GraphQL	GraphQL APIs are the new, Facebook-developed standard that provides database access over a single POST endpoint (typically /graphql). GraphQL APIs solve a common RESTful API problem — that of requiring multiple calls to populate a single UI page — while introducing other additional problems.
SOAP	SOAP uses the verbose eXtensible Markup Language (XML) for remote procedure calls (RPCs). It can still be found in legacy APIs.
XML-RPC	XML-RPC is a method of making procedure calls over the internet that uses a combination of XML for encoding and HTTP as a communications protocol.
gRPC	gRPC APIs are a Google-developed, high-performance binary protocol over HTTP/2.0, which are used mostly for east-west communication.
OpenAPI	OpenAPI is a description and documentation specification for APIs. In its older versions, OpenAPI was known as Swagger, and the terms are still often confused.

What is the difference between APIs and endpoints?

People often use “API” when what they are really talking about is a single API endpoint. APIs, sometimes called services or API products, are collections of endpoints that serve a business function. An endpoint, on the other hand, is a resource (or resource path, also known as URI or Uniform Resource Identifier) and the operation performed on it (create, read, update, or delete — operations that in RESTful APIs are typically mapped to the HTTP methods POST, GET, PUT, and DELETE).

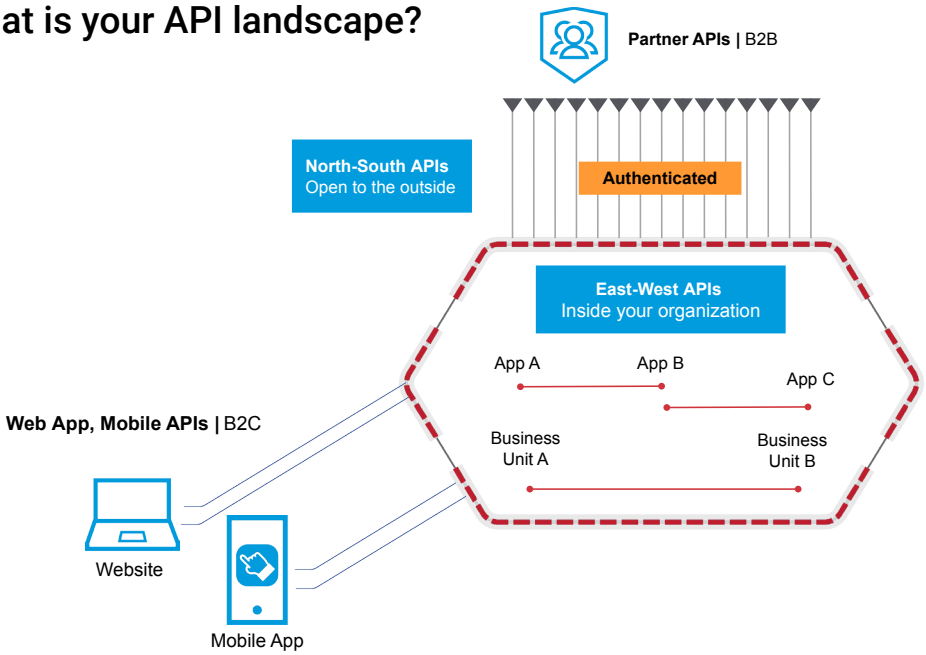
What is a north-south API?

These are APIs that an organization leaves accessible to the outside world, primarily to conduct business with its business partners. This is called API exposure. For example, banks embracing open banking may expose their accounts to other fintech or financial services organizations via APIs. Healthcare organizations may expose patient records to insurance companies and other medical organizations via APIs. Hospitality organizations may expose their reservation system to travel agents or aggregators via APIs. APIs are the connective tissue that links organizations. North-south APIs are often considered safe because access is authorized and authenticated. Typically, this is the fastest growing and largest volume of APIs, and consequently it’s the largest attack surface for most organizations.

What is an east-west API?

These are APIs that an organization uses internally and should not be accessible to anyone outside the business. These APIs connect internal applications, or connect business units or departments.

What is your API landscape?



What are the differences between B2C APIs and B2B APIs?

Business-to-consumer (B2C) APIs are the APIs that power web and mobile applications. They are typically consumed by modern front-end clients to allow end users access to the company's business functionality exposed by these APIs.

Business-to-business (B2B) APIs are the APIs offered by the organization to other organizations to conduct business, and sometimes to provide value to their joint customers (B2B2C).

B2B APIs are fundamental to the enterprise's digital transformation, as they enable it to streamline how it works with its suppliers, resellers, and other partners, as well as provide better experiences to its customers.

Examples of B2B APIs include:

- Open banking APIs
- Supply chain management APIs
- Electronic invoicing and payments among trading partners

Since the consumers of the APIs differ greatly, the security controls available for securing these APIs also vary. The industry has been focused on B2C use cases until fairly recently, but even in those cases, the focus has not been on securing B2C APIs, but rather on securing web applications. The security controls employed for protecting B2C web applications do not guarantee security for B2C APIs (e.g., web application firewall [WAF]/ web application and API protection [WAAP]) or do not offer security for B2C APIs at all (e.g., most bot protection solutions).

Examples of B2B APIs		
		
Open banking APIs	Supply chain management APIs	Electronic invoicing and payments between trading partners

Protecting B2B APIs is a growing problem. When it comes to B2B APIs, among first-generation vendors there is no dedicated visibility and security solution that covers bulk data access on behalf of shared users (as is the case, for example, in open banking — in which fintech companies and financial institutions consensually share customer data). Newer API security solutions that offer behavioral analytics are filling the void and addressing this issue.

What are the differences between private APIs and public APIs?

Private APIs, sometimes also referred to as internal APIs, are intended to be used by the company's developers and contractors. Often a part of a service-oriented architecture (SOA) initiative, private APIs are meant to streamline internal development by enabling different departments or business units to access one another's data efficiently and effectively.

In contrast, public APIs, also known as external APIs, are exposed to consumers outside the company. In their most extreme manifestation, as open APIs, they can be freely consumed by anyone. But in any case, they require tighter management and great documentation, so they can be used by engineers outside the company.

It's important to note that private APIs that can be accessed over the internet are not really private after all, in the strict sense of the word. Take for example ACME's B2C API used only by ACME mobile apps (developed in-house by ACME engineers). You may be tempted to call this a private API, but since the traffic to this API arrives from the internet ("outside the company"), this API is not really private — it is simply undocumented. Hackers attack such APIs regularly by intercepting the traffic and reverse engineering mobile apps to find their corresponding APIs.

API Security Explained

What is API security?

API security is a strategy for protecting the APIs that organizations use to support their business processes. However, because API sprawl can occur, it can be difficult to have a complete understanding of your organization's entire API landscape. Identifying and mitigating API security risks requires security controls that are sophisticated enough to address this issue. The APIs that need protection may include:

- APIs that make data easily accessible by customers or business partners
- APIs consumed from business partners
- APIs that are implemented and used internally to make application functionality and data available to various systems and user interfaces in a standardized and scalable manner

An effective API security strategy must include systematic techniques for assessing risk and potential impact, as well as executing appropriate mitigation measures. The first step in assessing risk is to build an inventory of all sanctioned and unsanctioned APIs published and used by the organization. This inventory should include attributes such as:

- Data classifications, which at a minimum distinguish between "not sensitive," "sensitive," and "very sensitive" data
- Risk indicators, such as API vulnerabilities and misconfigurations

Additionally, API visibility and risk mitigation measures must consider a diverse collection of possible threats, including:

- Detecting and preventing the use of unsanctioned "shadow APIs"
- Identifying and remediating API vulnerabilities and misconfigurations that threat actors could potentially exploit
- Preventing instances of API misuse like business logic abuse and data scraping

> **API security is a strategy for protecting the APIs that organizations use to support their business processes.**

How big is the API security problem?

API security risks are already one of the most pressing risks faced by enterprise security teams, and the challenge is only growing as more customer interactions and internal business processes make use of APIs. In short, API usage is exploding, and many security teams are playing catch-up with their API security strategies. For this reason, API security is quickly emerging as one of the top priorities and areas of concern for IT and security executives.

How is API security different from application security?

While API security and traditional application security are related disciplines, API security is a distinct challenge for two key reasons — the scale and complexity of the problem.

Greater scale

Three factors contribute to the rapid growth of API use:

1. The use of microservices, an architecture that mandates the use of APIs for service-to-service communication, is growing.
2. In the direct-user channel, modern front-end application frameworks such as React, Angular, and Vue use APIs and are displacing legacy web apps that sometimes do not.
3. APIs are added to address completely new channels as well (e.g., partners, Internet of Things, and business automation).

Flexibility leading to complexity

Unlike web applications, APIs are designed to be used programmatically in many different ways, which makes differentiating legitimate usage from attacks and abuse extremely challenging.

What are the best practices for protecting APIs?

We recommend that organizations interested in enhancing their API security start with the following best practices:

- ✔ Integrate API security standards and practices with your organization's software development lifecycle.
- ✔ Incorporate API documentation and automated security testing into your continuous integration/continuous delivery (CI/CD) pipelines.
- ✔ Ensure that appropriate and effective authentication and authorization controls are applied to your APIs.
- ✔ Implement rate limiting measures to help prevent APIs from being abused or overwhelmed.
- ✔ Augment rate limiting and other application-level measures with specialized gateways and/or content delivery networks (CDNs) to mitigate the risk of distributed denial-of-service (DDoS) attacks.
- ✔ Make API security testing an integral part of your broader application testing processes.
- ✔ Perform continuous discovery of APIs.
- ✔ Implement a systematic approach for identifying and remediating common API vulnerabilities, including the OWASP API Security Top 10.
- ✔ Use signature-based threat detection and prevention as a baseline level of protection against known API attacks.
- ✔ Augment signature-based detection with artificial intelligence (AI) and behavioral analytics to make API threat detection more scalable, accurate, business relevant, and resilient against novel threats.
- ✔ Ensure that API security monitoring and analysis extends over multiple weeks and API sessions.
- ✔ Complement API security monitoring and alerting with on-demand access to API inventory and activity data for use by threat hunters, developers, DevOps, and support personnel.

The best way to approach API security best practices is by thinking in terms of organizational maturity using a framework like the one below.

Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
Visibility to API activity	API discovery	Risk audit	Behavioral detection	Response	Investigate and threat hunt
Do you have logs for API environment? Are your logs sufficient? How do you handle sensitive data?	Do you know all your micro-services? Do you know all your APIs?	What is your risk posture? Miscon-figured? Errors? Documented? Sensitive data?	Can you detect misuse or business abuse? Can you identify the entities in your APIs?	Can you deploy automated responses? Are responses customizable?	Can you find threats in your past data? Can you hunt for threats?
Use your own data Sensors not required	Breadth of coverage is most important	Audit of entire estate, not just where sensors deployed	Behavioral analytics requires data and SaaS	Open platform to create response playbooks	Requires historical data and SaaS

API Security Risks and Abuse

What is an API vulnerability?

An API vulnerability is a software bug or system configuration error that an attacker can exploit to access sensitive application functionality or data, or otherwise misuse an API. The OWASP API Security Top 10 offers a useful overview of some of the most widely abused API vulnerabilities that organizations should attempt to identify and remediate.

Are all API vulnerabilities tracked on the OWASP API Security Top 10?

The OWASP API Security Top 10 is an excellent starting point for organizations seeking to improve their API security posture. Its categories cover a wide range of possible API risks. But it's important to note that the categories included in the OWASP API Security Top 10 are quite broad. So it's important to drill down and apply focus to the sub-areas in every one of them. API attackers frequently attempt to exploit authorization issues (covered by OWASP extensively), but there are also API risks that fall completely outside of the OWASP API Security Top 10, such as the abuse of logic bugs (not covered by OWASP at all).

How can APIs be abused?

APIs can be attacked and abused in many different ways, but some of the most common examples include:

- **Vulnerability exploitation:** Technical vulnerabilities in underlying infrastructure can lead to server compromise. Examples for these types of vulnerabilities range from the Apache Struts vulnerabilities (CVE-2017-9791, CVE-2018-11776, and friends) to Log4j vulnerabilities (CVE-2021-44228 and friends).
- **Business logic abuse:** These are the scary scenarios that keep CISOs up at night, as legacy security controls are useless against them. Logic abuse is when a threat actor exploits application design or implementation flaws to prompt unexpected and unsanctioned behavior.
- **Unauthorized data access:** Another common form of API abuse is exploiting broken authorization mechanisms to access data you should not be allowed to access. These vulnerabilities carry many names, such as broken object-level authorization (BOLA) and insecure direct object reference (IDOR), as well as broken function-level authorization (BFLA).
- **Account takeover:** After a credential theft or even a cross-site scripting attack, an account can be taken over. Once that happens, abuse of even the most well-written and perfectly secured API is possible. After all, if you're not performing behavior analysis, any authenticated activity is considered to be legitimate usage.
- **Data scraping:** As organizations make datasets available through public APIs, threat actors may aggressively query these resources to perform wholesale capture of large, valuable datasets.
- **Business denial of service (DoS):** By asking the back end to perform heavy tasks, API attackers can cause "erosion of service" or a complete DoS at the application layer (a very common vulnerability in GraphQL, but something that can happen with any resource-intensive API endpoint implementations).

What is a zombie API?

Driven by changing market and business requirements, APIs are in constant flux. As new endpoint implementations are released to meet new business needs, fix bugs, and introduce technical improvements, older versions of these endpoints are sunset. Managing the decommissioning process of old endpoints is not trivial. Often, endpoint implementations that should have been deprecated remain alive and accessible — those are called zombie endpoints.

How can I find the various types of shadow APIs?

The best way to conduct enterprise-wide shadow API discovery is to ingest and analyze API activity log data from the sources providing the broadest coverage. Deploying per-app sensors around every API can provide rich information about the API activity, but by definition cannot provide broad coverage, as legacy APIs or APIs you don't know about remain in the shadows.

Examples of API activity data log sources include:

- CDNs
- API gateways
- WAFs
- Kubernetes container orchestration

Once the raw data from all available sources is collected, AI techniques can be used to transform it into a human-understandable inventory of all APIs, endpoints, and parameters. From there, additional analysis can be performed to classify these elements and identify shadow APIs that should be eliminated or brought into formal governance processes.

How do you protect internal APIs and business-to-business APIs?

It really depends on the definition of "internal." Some people frequently refer to APIs exposed over the internet to their own organization's web and mobile applications as "internal APIs." And while the documentation for these APIs may indeed be accessible only by company employees and contractors, hackers have become very adept at analyzing apps and reverse engineering the APIs serving them via app disassembly toolkits and proxies such as Burp Suite.

However, if "internal APIs" are defined as east-west APIs, which cannot be accessed from outside the organization, then the main threat is reduced to insider threat. Protect internal APIs (in the former sense of the word) and your business-to-business (B2B) APIs like most APIs: Start by protecting the secure software development lifecycle (SSDLC) and continue by ensuring authenticated and authorized access; managing quotas, rate limits, and spike arrests; and protecting against known signatures with WAFs/WAAPs. Because of the sensitive and often bulk nature of the transactions in B2B APIs, consider adding — if possible — strict authentication mechanisms such as mTLS for B2B APIs.

And for both — we recommend you employ behavioral analytics, especially if you have many entities involved, which may make the process of distinguishing between legitimate and illegitimate behavior difficult. For example:

- How do you know if the API credentials of a specific user have been compromised?
- How would you know if your invoicing API is being abused by a partner enumerating through invoice numbers to steal account data?

Protection of B2B APIs and internal APIs requires business context that cannot be gained by analyzing technical elements like IP addresses and API tokens alone. Using AI and behavioral analytics to gain visibility into business-relevant entities is the only way to understand and manage B2B API and internal API risk effectively. Business context and historical benchmarks for normal use of APIs by specific entities like your users or partners — or even business process entities (invoice, payment, order, etc.) make it possible to see anomalies that would otherwise go undetected.

Do API gateways have built-in security?

Many organizations that use a strategic approach to APIs use API gateways. Most API gateways have rich integrated security features that organizations should take advantage of — first among those is authentication (and authorization as well, if you can leverage OpenID Connect). However, merely doing authentication, authorization, and quota management at the API gateway is not sufficient, for several reasons:

- The discovery gap of API gateways: API gateways only have visibility and control over APIs that they are configured to manage, making them ineffective at detecting shadow APIs and endpoints.
- The security gap of API gateways: API gateways can enforce authentication and, to some degree, authorization schemes, but do not inspect payloads (as do WAFs and WAAPs), nor do they profile behavior to detect abuse.

What are the most common API misconfiguration errors?

The number of possible API misconfigurations is nearly endless, given the wide number of ways that APIs are used. However, the following are some common themes:

Broken or no authentication

Authentication is foundational to securing sensitive data that is made available via APIs. Step one is to ensure that all APIs carrying sensitive data have authentication in place initially. But it's also important to protect authentication mechanisms from brute-force attacks, credential stuffing, and use of stolen authentication tokens via rate limiting. Misconfigurations that allow API consumers to bypass authentication mechanisms can sometimes happen, often around token management (for example, some notorious JSON Web Token validation issues, or not checking the token scope).

Broken authorization

One of the most common uses of APIs is to provide access to data or content, including sensitive information. Authorization is the process of verifying that an API consumer is eligible to access the data they are trying to access, prior to making it available to them. This can be done at the object or resource level (for example, I can access my orders but not someone else's), or at the function level (as is often the case with administrative capabilities). Authorization is hard to get right because of the high number of edge cases and conditions, and because of the various flows API calls can take between microservices. If you don't have a centralized authorization engine, your API implementation likely includes some of these vulnerabilities, such as BOLA and BFLA.

Security misconfiguration

There are many possible types of security misconfigurations beyond the authentication and authorization issues mentioned above, including insecure communication (e.g., using vulnerable cipher suites or not using TLS [formerly SSL]), unprotected cloud storage, and overly permissive cross-origin resource sharing policies.

Lack of resources and rate limiting

When APIs are implemented without any limits on the number of calls that API consumers can make, threat actors can overwhelm system resources, leading to service degradation or full-scale DoS. At the very least, rate limits must be enforced on access to any unauthenticated endpoint, with authentication endpoints being of critical importance — or else brute-force attacks, and credential stuffing and credential validation attacks, are simply bound to happen.

What are API attacks?

API attacks are attempts to use APIs for malicious or otherwise unsanctioned purposes. API attacks take many forms, including:

- Exploitation of technical vulnerabilities in API implementations
- Use of stolen credentials and other account takeover techniques to masquerade as a legitimate user
- Business logic abuse that enables use of APIs in unexpected ways

What is credential stuffing for APIs?

Leakage of user ID and password information from websites and software as a service (SaaS) platforms has become a regular occurrence. Often, these incidents result in large sets of credentials being shared widely online. Credential stuffing is the practice of using authentication credentials leaked from previously breached websites to perform automated login attempts to other websites. This technique is based on the premise that some percentage of users use the same credentials for multiple sites. Increasingly, this practice is being applied to APIs directly (not through the front end — be that a web or mobile application). This enables attackers to automate the attack more easily, since — after all — APIs are created for ease of consumption.

What is data exfiltration through APIs?

Data exfiltration is a frequent outcome of successful API attacks and abuse. In some cases, it refers to highly sensitive, nonpublic information that has been stolen by a threat actor through API attacks and abuse. However, it can also apply to less severe types of API abuse, including aggressive data scraping of publicly available data to assemble large datasets that are valuable in aggregate form.

API Security Solutions and Trends

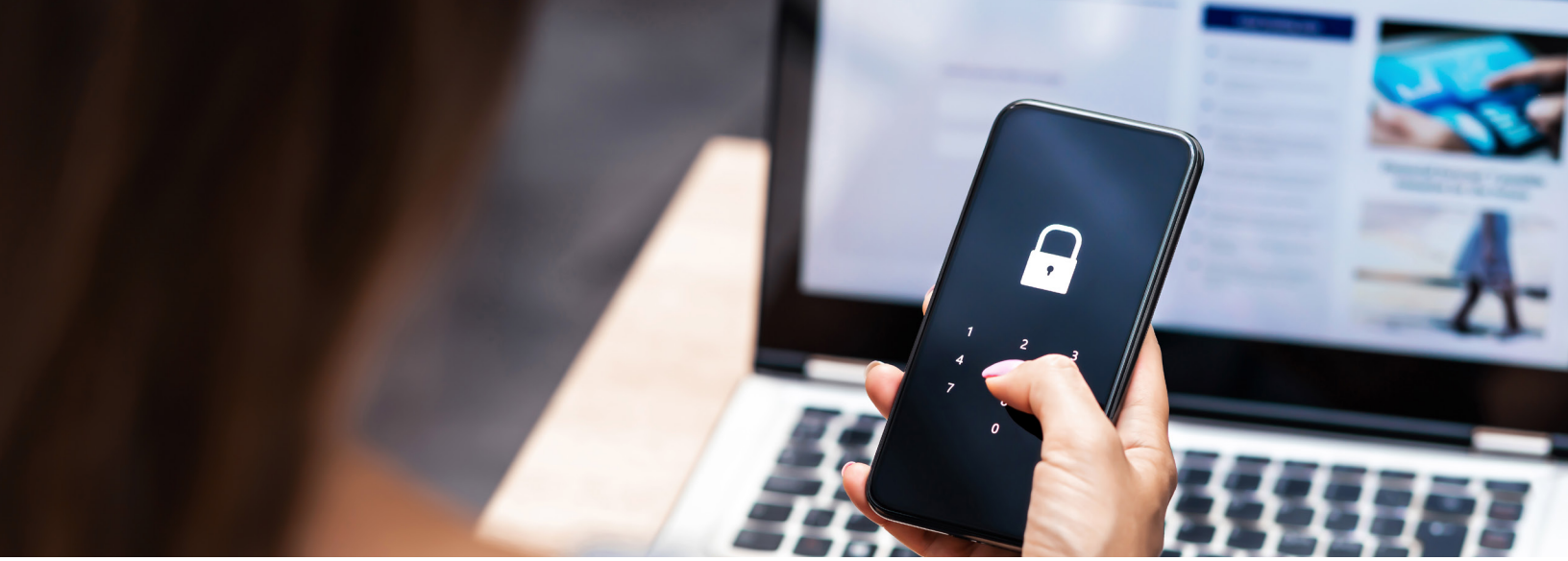
What are the latest trends in API security?

The following are key trends that security executives should consider when developing an API security strategy.

- **Behavioral analytics and anomaly detection:** Rather than trying to predict possible attacks and relying only on signature-based detection and predefined policies (e.g., WAF) to mitigate risk, organizations are increasingly adding AI and behavioral analytics to view other API activity in business context and detect anomalies.
- **Transition from on-premises to SaaS:** While many first-generation API security products were deployed on-premises, SaaS-based approaches are growing in popularity because of their speed, ease of deployment, and ability to harness the power of AI and machine learning at scale.
- **Analysis of larger time windows:** API security approaches that only analyze individual API calls or short-term session activity are being supplanted by platforms that analyze API activity over days and sometimes weeks, from completing basic automated WAF policy optimization to performing behavior analytics and detecting anomalies.
- **DevSecOps — embracing nonsecurity stakeholders:** One of the best ways to reduce API risks is by creating greater linkage among API security strategies and tools and the developers and systems involved in the creation, implementation, and configuration of APIs.
- **API-enabled API security:** While detecting and mitigating active API attacks and instances of abuse is critical, forward-thinking organizations are finding ways to use on-demand access to API security data and insights to improve threat hunting, incident response, and API development practices.

What is signature-based API security?

Signature-based API security techniques monitor for known attack characteristics and patterns, and generate security alerts and other automated responses when matches are observed. This is typical of a WAF. The value of signature-based detection is that it is in-line, which means that it can block threats instantly. This means that if an organization is informed about incoming API traffic that is compromised or behaving abnormally, it can use signature-based API security to block the threat immediately.



Look for a WAF that is part of a larger WAAP solution. That WAAP should be able to offer advanced detections through machine learning that learn from the attack signature patterns and can remain agile at scale. Additionally, look for a WAAP that is integrated with an API security solution that offers behavioral analytics and customized responses to get the best of both worlds. Together, these solutions would offer complete visibility, detection, and response internally and externally.

What is API detection and response?

API detection and response is an emerging category of API security focused on deep analysis of historical data to:

- Determine a baseline of the behavior of all API consumers
- Detect attacks and anomalies that indicate possible API abuse and misuse

Effective API detection and response at scale can only be delivered under a SaaS model because of the large datasets involved in the need for resource-intensive AI and machine learning techniques.

What is advanced API threat protection?

Advanced API threat protection is a SaaS-based approach to API security that combines behavioral analytics with threat hunting to:

- Discover all APIs in use by an organization, including shadow or zombie APIs
- Apply machine learning to overlay business context about how APIs are being used and abused
- Perform behavioral analysis and threat hunting on APIs and API activity data that is stored over extended time windows

What is an API security platform?

An API security platform is a SaaS-based offering that is specially designed to:

- Create a continuously updated inventory of all APIs in use enterprise-wise (whether sanctioned or not)
- Analyze APIs and their usage with AI and machine learning techniques to discover business context and determine a baseline of expected behavior
- Detect anomalies in API usage and, when necessary, provide alert and supporting data to security information and event management (SIEM) and security orchestration, automation, and response (SOAR) workflows
- Provide on-demand access to API inventory, activity, and threat information to both security and nonsecurity stakeholders

What is an API company?

Now that IT and security leaders are using APIs more strategically, they may need to engage specialized API partners. The three most common types of API companies are:

- API gateway companies that provide technology to accept API calls centrally and route them to the appropriate back-end resources and microservices
- API security platform companies that ensure businesses have awareness of all active APIs, detect instances of attacks and abuse, and provide rich data about how APIs are being used
- WAAP and API security platform companies that can help seamlessly transfer API traffic data while still offering the capability to discover APIs on and off platform; this is ideal for vendor consolidation and closing digital gaps

What is threat hunting in APIs?

Many security teams conduct proactive threat hunting activities to identify possible threats early and respond with countermeasures. Many first-generation API security products provide limited value to threat hunting teams, since they are focused on alerting, and do not store API activity at all, which means there is no data to query and hunt through. More advanced API security companies store large and context-rich API activity datasets and make this information available both in a GUI and through APIs, so threat hunters can leverage the data.

What is WAAP?

Web application and API protection (WAAP) is a categorization that the research firm Gartner uses for its industry coverage of emerging web and API threats. It is an evolution of earlier industry coverage of the web application firewall (WAF) market in response to the growing strategic importance of API security and the move by WAF platforms to the cloud as managed SaaS.

API Security Best Practices

What is an API documentation example?

The most common form of API documentation for RESTful APIs (which are the most common type of web API) is a collection of Swagger files based on the OpenAPI specification. Ideally, API documentation is created by developers when an API is designed or implemented. In reality, however, API documentation is frequently out of date, resulting in a mismatch between real-world API usage and documentation. To address this issue, some API security platforms, including Akamai API Security, can generate Swagger files from the actual API activity, highlighting the gaps between what is documented and what is actually deployed — an integral component in any API risk assessment.

Is there an API security checklist businesses should follow?

Effective API security requires many detailed steps and ongoing practices. However, the following is an API checklist that security teams can use as a starting point as they move toward a more sophisticated approach to API security:

- ✔ Does your API security approach include a mechanism for continuous enterprise-wide API discovery?
- ✔ Are you leveraging the cloud/SaaS to gain access to AI and machine learning techniques, and avoid unnecessary deployment complexity?
- ✔ Is your API security approach analyzing data over a long enough time horizon (ideally, 30 days or more)?
- ✔ Are you implementing a general purpose API security approach that won't lock you into specific data center or cloud infrastructure models?
- ✔ Will your approach give your teams the business context they need to truly understand the API activity and possible risks that are being observed?
- ✔ Do you have a strategy for two-way automation between your API security platform and other related business processes like SIEM/SOAR, threat hunting, documentation, DevOps tooling, etc.?
- ✔ Are you taking steps to welcome nonsecurity stakeholders, like developers, into your API security tools and processes?

Is there an API taxonomy that security teams should understand?

The following are common categorizations and descriptions of APIs that may come up in a security context.

Sanctioned APIs	Unsanctioned APIs	Out-of-date APIs
Published API (with Swagger documentation or similar)	Shadow API	Deprecated API
	Rogue API	Legacy API
	Zombie API	Zombie API
	Hidden API	Orphaned API

Feeling more equipped to begin, restart, or build on your API security journey? Akamai can help.

Explore the product portfolio or talk to a sales representative.

Akamai protects your customer experience, workforce, systems, and data by helping to embed security into everything you create — anywhere you build it and everywhere you deliver it. Our platform's visibility into global threats helps us adapt and evolve your security posture — to enable Zero Trust, stop ransomware, secure apps and APIs, or fight off DDoS attacks — giving you the confidence to continually innovate, expand, and transform what's possible. Learn more about Akamai's cloud computing, security, and content delivery solutions at akamai.com and akamai.com/blog, or follow Akamai Technologies on X, formerly known as Twitter, and LinkedIn.