Akamai

WHITE PAPER

API Security Fundamentals: Build Your Knowledge, Secure the Enterprise

Introduction

APIs have evolved rapidly from an implementation detail to a strategic enabler of digital innovation. Every time a customer, partner, or vendor engages with a business digitally, there's an API behind the scenes facilitating a seamless data exchange.

As APIs proliferate, so do their risks. In the race to quickly create and release new applications and AI-enhanced services, the underlying APIs are too often misconfigured, lacking in security controls, and vulnerable to easily executed attacks.

As a result, APIs have emerged as a top attack vector, leaving many security teams to play catch-up with their API security strategies. Therefore, API security is quickly emerging as a top strategic priority for IT and security executives.

Whether you're looking to ground yourself in API security basics or are assembling a list of the right questions to ask, this guide offers the details you need to know, including:

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

To go directly to API security best practices, you can skip ahead to page 10.

Table of Contents

API basics	4–9
API security explained	10–12
API security risks and abuse	13–18
API security solutions and trends	19–22

API basics

What is a web API?

A web application programming interface, or API, consists of one or more endpoints of 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).

Web APIs are different from other APIs, such as those exposed by the operating system or by libraries of applications running on the same machine, but the general term "API" usually refers to a HTTP-based (web) API, especially in the context of enterprise digital transformation and API security.

What are the most common types of APIs?

The following table contains terms that refer to different usage models and technical 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 table defines these common types, as well as 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; these types of APIs are exposed to the internet
Private API	An API that is implemented in 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 access (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

RESTful API	Representational state transfer (RESTful) is the most common type of web API that uses plain text, HTML, XML, YAML, or JSON to deliver 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 have become the de facto standard for any web API, including those used for B2B
GraphQL	GraphQL APIs are the newer, Facebook-developed standard that provides database access over a single POST endpoint (typically /graphql); it solves a common RESTful API problem — that of requiring multiple calls to populate a single user interface page
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 and are used mostly for east-west (within internal network) communication
OpenAPI	OpenAPI is a description and documentation specification for APIs. It may be helpful to know that the term Swagger refers to the original specification, and OpenAPI refers to the open standard developed by the OpenAPI Initiative

What is the difference between APIs and endpoints?

People often use “API” when they are really referring to a single API endpoint. APIs, sometimes called services or API products, are collections of endpoints that serve a business function. An individual endpoint, on the other hand, is a resource (or resource path, also known as a URI or uniform resource identifier) along with the operation performed on it (create, read, update, or delete). In RESTful APIs, operations 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 data 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 systems to travel agents or aggregators via APIs.**

APIs are the connective tissue that allows disparate organizations to exchange data. 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 is 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 business units or departments. It is possible for a developer to make a mistake that makes east-west APIs accessible by accident. These APIs are not meant to be accessible or even known by external entities, but breaches do happen when threat actors find east-west APIs accessible via the internet.

What are the differences between B2C APIs and B2B APIs?

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

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

B2B APIs help streamline how an enterprise works with its suppliers, resellers, and other partners and how it provides better experiences to its customers.

Examples of B2B APIs include:

Open banking APIs

Supply chain management APIs

Electronic invoicing and payments between trading partners

Since the consumers of the APIs differ greatly, the security controls available for protecting these APIs also vary. The industry has been focused on B2C use cases until fairly recently, but even there, the focus has not been on securing B2C APIs but rather on securing web applications. The security tools and controls typically employed for securing B2C web applications offer certain benefits (e.g., web application firewall [WAF]/ web application and API protection [WAAP]) but cannot provide the degree of visibility, real-time monitoring, and protection required for securing B2C APIs from attacks.

Protecting B2B APIs is becoming increasingly challenging. These APIs are often easier targets for attackers because they frequently lack essential protection mechanisms. Earlier API security tools had limited visibility into B2B APIs and struggled to secure APIs that facilitated bulk data access on behalf of shared users (as seen in open banking, where fintech companies and financial institutions consensually share customer data). However, newer API security solutions offer behavioral analytics and can recognize anomalous activities, effectively addressing these concerns.

What are the differences between private APIs and public APIs?

Private APIs, sometimes also called 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 each other's data efficiently and effectively.

By contrast, public APIs, also known as external APIs, are exposed to consumers from outside the company. In their most extreme manifestation, as open APIs, they can be freely consumed by anyone. In all cases, they require tight 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 in the strict sense of the word. For example, let's say ACME's B2C API is 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 unpublished to external audiences. Hackers attack such APIs regularly by intercepting traffic and by reverse engineering mobile apps to find their corresponding APIs.

API security explained

What is API security?

API security is a strategy for gaining visibility into, rigorously testing, and protecting every API across an enterprise. This includes APIs that are integral to applications, business processes, and cloud workloads. However, because both internal and external APIs are being produced so rapidly and in such large numbers, it can be difficult to have a complete understanding of your organization's entire API landscape. Many organizations lack visibility into how many APIs they actually have and which APIs return sensitive data when called. Identifying and mitigating API security risks require security controls that are sophisticated enough to provide this kind of visibility and data analysis. 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 building 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 (see sidebar)
- Identifying and remediating API vulnerabilities and misconfigurations that threat actors could potentially exploit
- Preventing instances of API misuse, such as business logic abuse and data scraping

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.
3. APIs are added to address completely new channels as well (e.g., partners, IoT, 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.

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

Published API (with Swagger documentation or similar)

Unsanctioned APIs

- Shadow API
- Rogue API
- Zombie API
- Hidden API

Out-of-date APIs

- Deprecated API
- Legacy API
- Zombie API
- Orphaned API

What are the best practices for protecting APIs?

Enhancing your API security starts 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 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 Top 10 API Security Risks.
- Use signature-based threat detection and prevention as a baseline level of protection against known API attacks.
- Augment signature-based detection with AI and behavioral analytics to make API threat detection more scalable, accurate, business relevant, and resilient against novel threats.
- Ensure that the API security monitoring and analysis process 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.

Your ability to implement these API security best practices depends on where you are in your journey toward a mature API security strategy (see sidebar).

Stages of API security maturity

Stage 1: Visibility and discovery

You are in the process of discovering all your APIs and the microservices they support by using an automated approach. Breadth of coverage is critical, as overlooked APIs (such as those no longer in use) are a prime target for threat actors.

Stage 2: Testing

You test all your APIs to ensure that they are coded correctly and that they perform their intended function. Testing performed prior to deploying an API is the upper end of this maturity stage; risk is eliminated before the API goes into production, and any needed fix is exponentially less expensive.

Stage 3: Risk audit

You continually audit your entire API environment to identify misconfigured APIs or other errors. Your audit also ensures adequate documentation of every API and determines whether they contain sensitive data or lack appropriate security controls.

Stage 4: Runtime protection

You are using a solution with automated runtime protection, which can differentiate between normal and abnormal API activity. By monitoring API interactions this way, you're able to detect behaviors indicating a threat in real time.

Stage 5: Response

You have solutions in place to respond to suspicious API behavior, such as a WAF or API gateway that blocks suspicious traffic before it can access critical resources. Your solutions use customized, automated rules.

Stage 6: Hunt for threats

You regularly perform forensic analysis on past threat data to learn whether alerts correctly identified threats and whether patterns emerged that enable proactive threat hunting using a combination of sophisticated tools and human intelligence.

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 Top 10 API Security Risks offer 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 Top 10 API Security Risks?

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 the categories included in OWASP API Security Top 10 are quite broad, so it's important to drill down to the sub-areas for each one. API attackers frequently attempt to exploit authorization issues (covered by OWASP extensively), but there are also API risks that fall completely outside the OWASP API Security Top 10, such as the abuse of logic bugs.

How can APIs be abused?

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

- **Vulnerability exploitation:** Technical vulnerabilities in underlying infrastructure can lead to server compromise. Examples range from the Apache Struts vulnerabilities (CVE-2017-9791, CVE-2018-11776) to Log4j vulnerabilities (CVE-2021-44228).

- **Business logic abuse:** Logic abuse is when a threat actor exploits application design or implementation flaws to prompt unexpected and unsanctioned behavior. These scenarios cause stress for CISOs and their teams because legacy security controls are useless against them.

- **Unauthorized data access:** Another common form of API abuse is the exploitation of broken authorization mechanisms to access data that should not be accessible. These vulnerabilities carry many names, such as Broken Object Level Authorization (BOLA), insecure direct object reference (IDOR), and broken function-level authorization (BFLA).

- **Account takeover:** After a credential theft or even an XSS attack, an account can be taken over. Once that happens, abuse of even the most well-written and perfectly secured API is possible. Using an API security solution that offers behavior analysis allows you to differentiate authenticated activity from illegitimate 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.

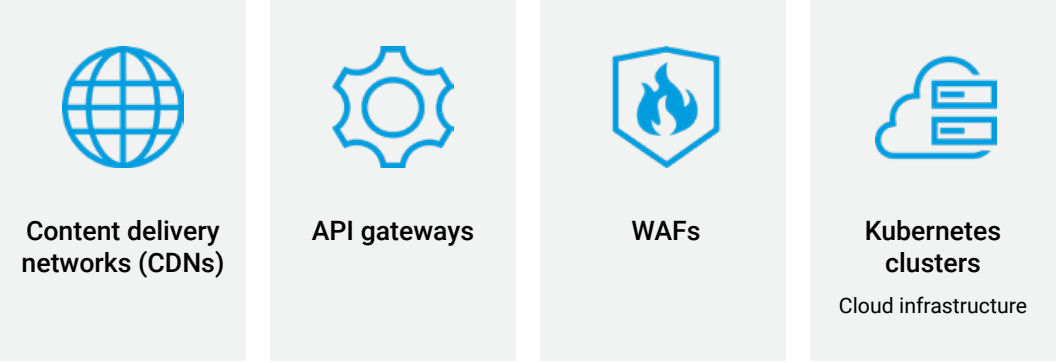
- **Business denial of service (DoS):** By asking the back end to perform heavy tasks, API attackers can cause erosion of service or a complete DoS at the application layer (a very common vulnerability in GraphQL but something that can happen with any resource-intensive API endpoint implementation).

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?

One of the ways to conduct enterprise-wide shadow API discovery is to ingest and analyze API traffic on your network. Examples of API traffic sources include:



Once raw data from all available sources is collected, AI techniques can be used to transform it into a comprehensive 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 B2B APIs?

It really depends on the definition of "internal." Some teams 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 to company employees and contractors, hackers have become adept at analyzing apps and reverse engineering the APIs 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 an insider threat. Protect east-west APIs and your B2B APIs like most other APIs: Start by securing the software development lifecycle (SDLC) and continue by ensuring access is authenticated and authorized. You can also implement managing quotas, rate limits, and spike arrests. Additionally, you can protect your APIs against known threats by using WAFs/WAAPs. For B2B APIs, consider adding strict authentication mechanisms, such as mTLS, because of the sensitive and often bulk nature of the transactions.

And for both east-west and B2B APIs, 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 invoice numbers to steal account data?**

Protection of B2B APIs and east-west APIs requires business context that cannot be gained by analyzing technical elements like IP addresses and API tokens alone. Using machine learning and behavioral analytics to gain visibility into business-relevant entities is the only way to understand and manage 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 offer sufficient risk protection?

Many organizations taking 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 performing 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 the 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 they do not inspect payloads (as WAFs and WAAPs do), 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 large number of ways that APIs are used. However, there are some common themes in misconfiguration:

Broken or no authentication

Authentication is foundational to securing sensitive data that is made available via APIs. Step one is ensuring 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 allowing API consumers to bypass authentication mechanisms can sometimes happen, often around token management (for example, some notorious JWT 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 that 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

In addition to the authentication and authorization issues mentioned above, there are many possible types of security misconfigurations, including insecure communication (e.g., failure to use SSL/TLS or the use of vulnerable cipher suites), 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:

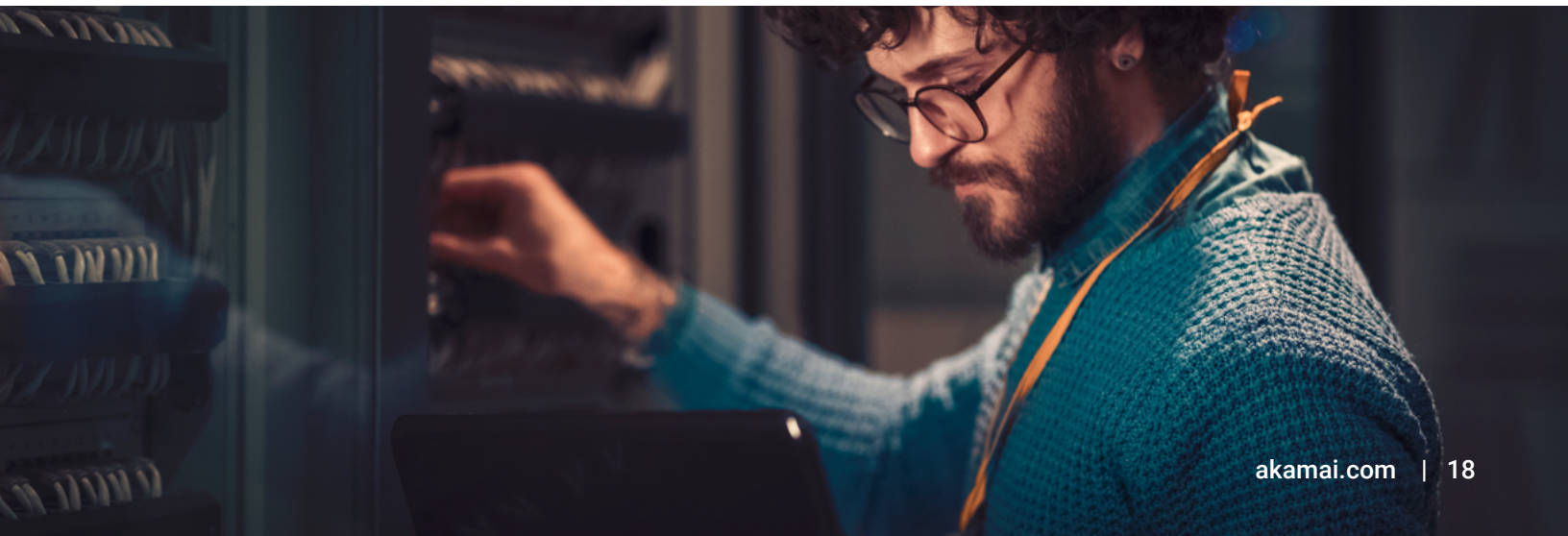
- Exploitation of technical vulnerabilities in API implementations
- Use of stolen credentials and other account takeover techniques to masquerade as a legitimate user
- Business logic abuse that enables use of APIs in unexpected ways

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, attackers are going straight for the APIs and targeting its authentication mechanisms. This enables attackers to automate the attack more easily since 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 an API attack. 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., WAFs) to mitigate risk, organizations are increasingly adding machine learning and behavioral analytics to view API activity in a 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 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 can analyze API activity over days and sometimes weeks, from completing basic automated WAF policy optimization to performing behavior analytics and detecting anomalies.

DevSecOps — embracing non-security stakeholders: One of the best ways to reduce API risks is by creating greater linkages between API security strategies and tools and the developers and systems involved in creating, implementing, and configuring APIs.

API-enabled API security: While detecting and mitigating active API attacks and instances of abuse are 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 then generate security alerts and other automated responses when matches are observed. This is typical of a WAF. The value: If an organization is informed about incoming API traffic that is compromised or behaving abnormally, it can use signature-based API security to block it immediately.

You'll want to find a WAF that is part of a larger WAAP solution that can offer advanced detections through machine learning that learns from attack signature patterns and can remain agile at scale. A WAAP that is integrated with an API security solution that offers behavioral analytics and customized responses will give you the best of both worlds. Together, these solutions offer complete API 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 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

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-wide (whether sanctioned or not)
- Analyze APIs and their usage 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 non-security stakeholders

What is an API security 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 organizations are aware of all active APIs and their potential risks, can detect instances of attacks and abuse, enable comprehensive security testing, 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?

Threat hunting involves actively searching for unknown or previously undetected threats. This proactive approach is critical for identifying new and emerging threats that may not have been seen before and for mitigating them before they can cause significant damage. One of the key techniques used in threat hunting is behavioral analysis. This involves analyzing the behavior of APIs to identify any suspicious or anomalous activity. For example, if an API suddenly requests thousands of records over a short period of time, it may indicate that the API business logic is compromised. Modern API security solutions provide specific threat hunting capabilities to allow security teams to identify possible threats early and respond with countermeasures.

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 protection solutions. It is an evolution of earlier industry coverage of the WAF market in response to the growing strategic importance of API security and the move by WAF platforms to the cloud as managed SaaS.

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 its documentation. To address this issue, some API security platforms 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 specific to an organization. However, the following is an API checklist that security teams can use as a starting point as they advance their API security:

- Does your API security approach include a mechanism for continuous enterprise-wide API discovery?
- Is API posture management integrated into the organization's broader security and risk management practices?
- 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 non-security stakeholders like developers into your API security tools and processes?

Akamai Security protects the applications that drive your business at every point of interaction, without compromising performance or customer experience. By leveraging the scale of our global platform and its visibility to threats, we partner with you to prevent, detect, and mitigate threats, so you can build brand trust and deliver on your vision. 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. Published 09/24.