apigee Google Cloud

Securing APIs in the Age of Connected Experiences

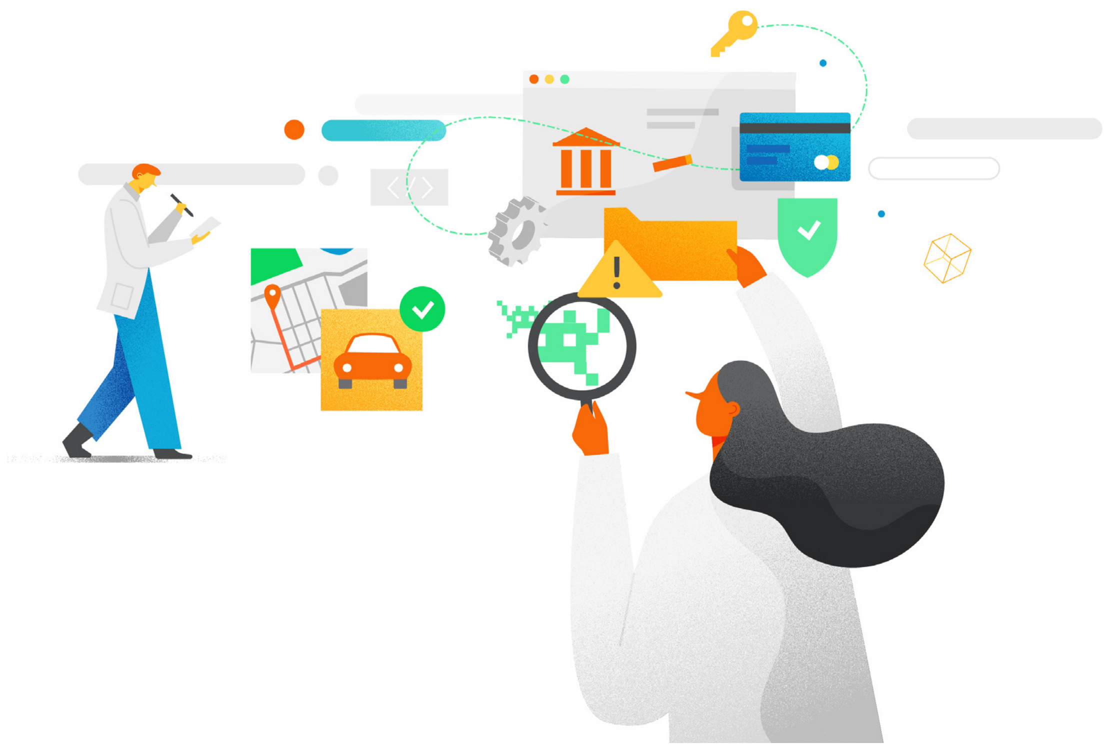
Table of contents

Executive Summary

The ways in which businesses attract, interact with, and serve customers have changed profoundly in recent years. Dedicated brick-and-mortar channels gave way to a combination of discrete legacy channels augmented by websites and apps, which in turn evolved into today's landscape, in which customers expect channels, whether physical or digital, to blend together into a cohesive connected experience.

Application programming interfaces (APIs) make these connections possible, and API-first development approaches have helped enterprises to not only enable connected customer experiences, but also participate in software ecosystems that may span billions of users and provide opportunities for unprecedented economies of scale. Yet the popularity of APIs and the varied ways in which enterprises leverage them are attracting hackers and bad actors. As business opportunities increasingly rely on digital connections, each point of interaction becomes both a potential source of business leverage and a potential source of risk.

The "walled garden" security techniques on which enterprises have long relied no longer offer sufficient protection. The web applications that fuel connected experiences may involve connecting services across multiple clouds or assembling software from multiple companies into a cohesive whole. The concept of a network perimeter simply no longer applies.



The growing number of breaches related to unsecured APIs has awakened many business and IT leaders to the risks. In the December 2017 report "How to Build an Effective API Security Strategy," Gartner analysts Mark O'Neill, Dionisio Zumerle, and Jeremy D'Hoinne predict that "[b]y 2022, API abuses will be the most-frequent attack vector resulting in data breaches for enterprise web applications."

The issue is not with APIs intrinsically so much as how enterprises must manage APIs to both enable connected customer experiences and keep those customers and their data safe. Every digital connection—that is, every API—needs to be managed and secured without compromising the flexibility that enables developers to build connected experiences to begin with. The challenge is to maintain a "no trust" environment that recognizes threats could come from any interaction while also enabling interactions to occur with little friction.

This ebook explores how modern software development techniques and customer expectations have necessitated that businesses apply security and protection at every point of interaction within a connected experience. Because APIs drive virtually every interaction, a company's ability to enable, profit from, and protect connected experiences is very much a matter of its ability to create, manage, and secure APIs.

APIs and the Rise of the Connected Experience Mandate

Not long ago, when a person wanted to pay back a friend or loan someone money, that person either handed over cash face-to-face, wrote a check or perhaps went into a banking branch to arrange a transfer. The digital world has changed all that. A few years ago, people started to routinely use websites and apps to send money without any physical currency being exchanged, and today, millions of users don't even have to sign into an extra app or open a website to do the job, as they can frequently send money without leaving digital experiences in which they're already engaged, such as messaging services.

This dramatic and rapid evolution in how people share money is just one example of a much bigger phenomenon: connected experiences.

From digital marketplaces and content platforms to smart homes and wearable devices, across virtually all industries, connected experiences are redefining how customers interact with businesses, how businesses must engage with customers—and how both must be protected from threats in a landscape built through global connections rather than within corporate firewalls.

Customers don't care if a business has a website, an app, or both—they care about getting what they need, when they need it, effortlessly. If a consumer starts watching their favorite show on TV, then hops in the back of their rideshare car, they expect to be able to pick up the show where they left off on their mobile device. Customers want to consume products and services in the ways they find convenient, whether that's via an interaction with an in-store employee, a kiosk, a browser, a mobile app, a connected home device, a virtual assistant, a voice interface, or some free-flowing combination of all of them.

APIs enable these connections.

APIs allow enterprises to express parts of their businesses as software that developers can leverage for digital experiences and products. When someone orders movie tickets via a voice assistant,that transaction is only possible because the voice assistant can call APIs that express capabilities such as showtime lookup, location search, and transaction approval. Regardless of whether a developer is familiar with the underlying technology, such as what language was used to code the capability, she can still use an API to leverage the capability and to combine it with other APIs to create new connected experiences.

When businesses use APIs to share their digital assets for mutual benefit, they can also enable the creation of ecosystems of software, developers, and customers. Few enterprises have the resources or expertise to own all aspects of all connected experiences on which their businesses may rely, especially as those experiences have become more decentralized.

Ecosystems enable businesses to augment their proprietary strengths and expand their exposure to users via partnerships, using APIs to extend assets to new users at virtually zero marginal cost.

The most elaborate ecosystems often revolve around "gravitational" platform businesses, such as Android, iOS, various online marketplaces, and many social media and workplace productivity services. These businesses provide technology infrastructure that mediates interactions between suppliers of goods and services and consumers of goods and services. Interactions drive these ecosystems and the connected experiences and business models that rely on them—and those interactions are driven by APIs.

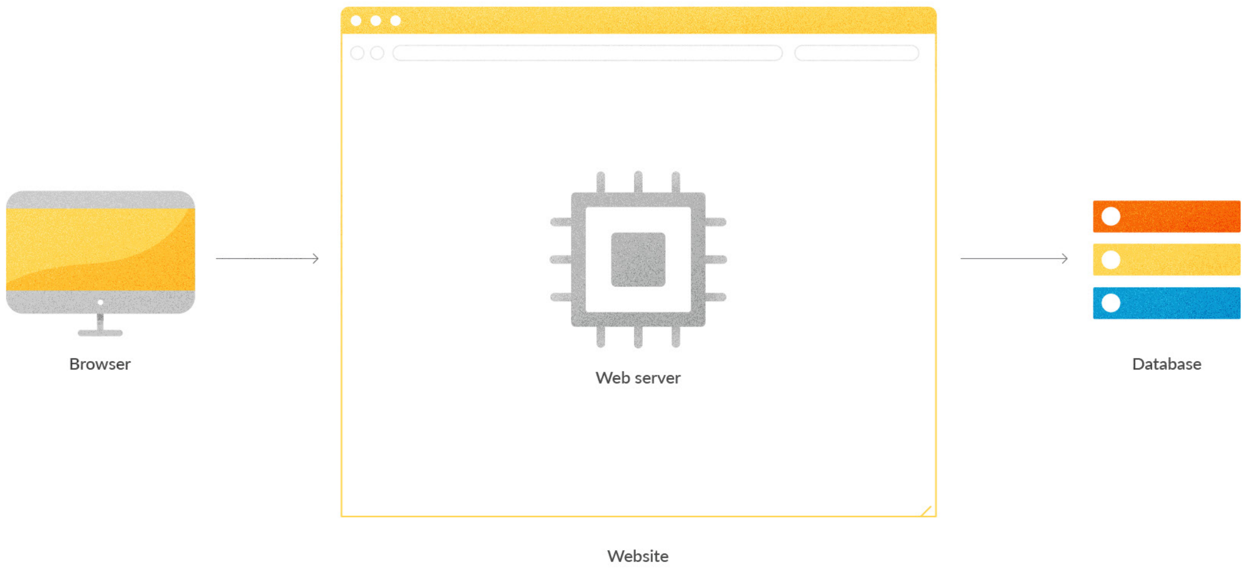
Consequently, enabling connected experiences, participating in the ecosystems that customers value, and protecting both customers and the business are all in large part matters of managing, leveraging, and securing APIs.

Though the new business opportunities that APIs enable are exciting for enterprise leaders (just as the connected experiences that APIs enable are compelling to consumers), these fundamental changes in how user experiences are constructed have also changed the enterprise security picture.

APIs *can* be leveraged to create a robust security layer—but only if enterprises execute it well. Aging IT security methodologies don't translate well to today's technology landscape, and attackers are taking advantage. Inconsistent or nonexistent precautions are a big reason that APIs are among the biggest emerging attack vectors for bad actors.

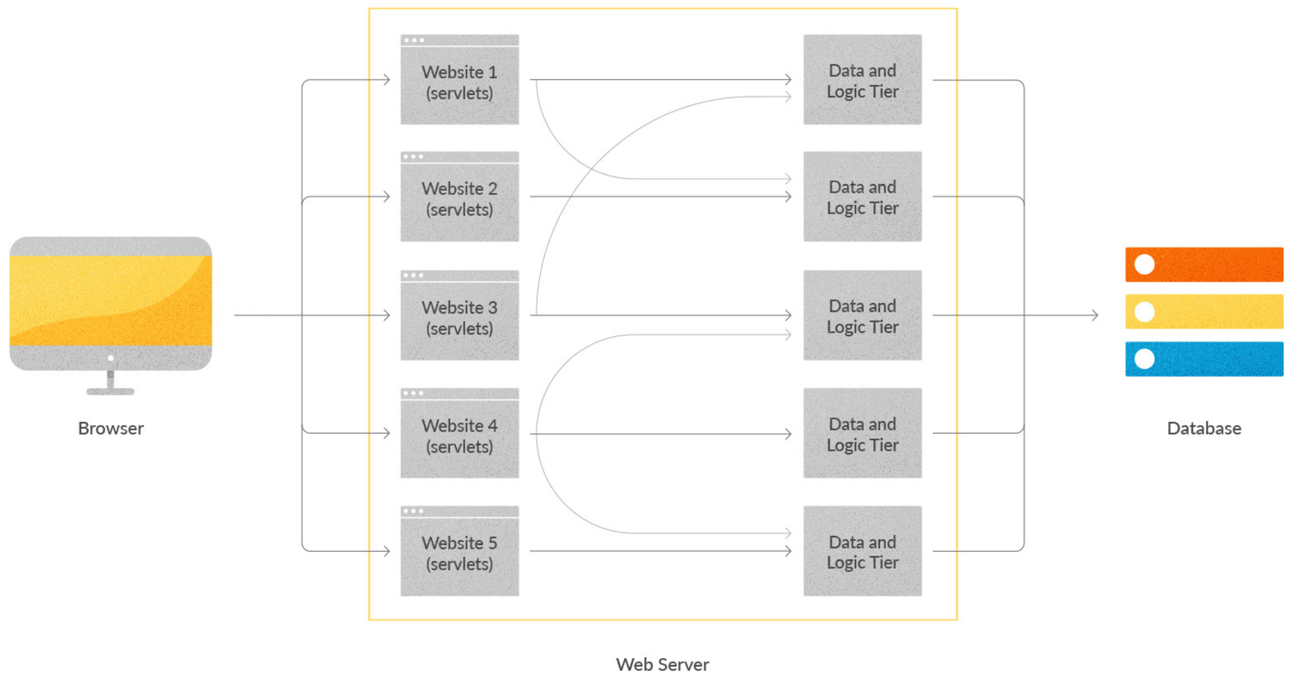
From Security at the Perimeter to Security at Every Interaction

Years before mobile apps, the data flow for creating a digital experience was very straightforward: a browser called a website attached to a database. In this approach to application development, the network perimeter served as protection, creating a "walled garden" around the data with a single "guard" mediating access. Specifically, credentials were exchanged for a cookie, and the cookie remained within the walled garden.



Early, straightforward web applications relied on cookies and the network perimeter for security.

Many active parts of the Internet still largely rely on this structure, but web applications quickly grew more complicated. Developers started leveraging reusable components, all deployed in the same container, in their applications. Concepts such as sticky sessions were introduced to help manage scale. Independently scalable web and data tiers enabled new, more immersive digital experiences. But the security paradigm remained essentially the same: a reliance on the network perimeter.

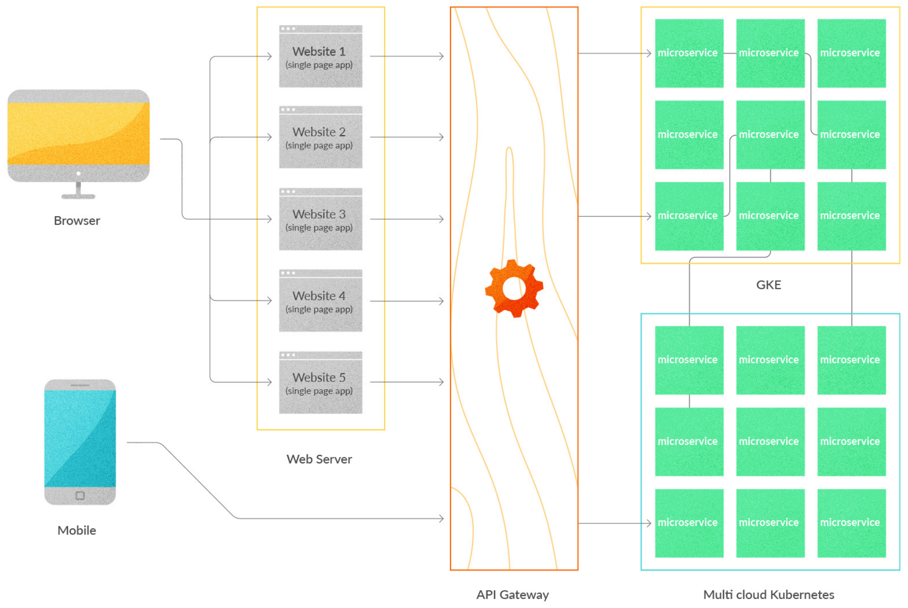


Walled gardens remained the dominant security technique as web applications became more complicated.

Today, the situation is significantly different. Legacy application development techniques generally cannot support the experiences that customers demand, and modern approaches to application development (and the agility and feature richness that these approaches enable) do not easily support aging security methods.

Modern applications are composed of many pieces connected via APIs. The UI is detached from the business logic. Components within the user experience are run as independently deployable services, some originating inside the enterprise and some hosted elsewhere. The applications are built differently and consumed differently, making for richer experiences and faster, more responsive development—but also obviating many legacy security approaches.

Many businesses are investing in microservices, for example, to enable faster, more efficient application development. But whereas in traditional models, applications are deployed to application servers, in a microservices-based architecture, servers are deployed to the application. One consequence is that tasks previously handled by the application server—such as authentication, authorization, and session management—are shifted to each microservice. If a business has thousands of such microservices powering their applications across multiple clouds, how can its IT leaders even begin to think of a perimeter?


Browser
Mobile
Website 1 (single page app)
Website 2 (single page app)
Website 3 (single page app)
Website 4 (single page app)
Website 5 (single page app)
Web Server
API Gateway
microservice
GKE
Multi cloud Kubernetes

Modern distributed architectures mean enterprises cannot rely on the network perimeter—the concept may be meaningless in multicloud scenarios. An API layer can apply security at every point of interaction between users and backends.

The customer's end experience relies on interactions among numerous components—and each needs to be secured.

When a user orders a ridesharing service, for instance, many services combine via APIs for the end experience—services to pinpoint the user's location and match the user with nearby drivers, services to call the user's profile and customer information, services to enable the user to purchase a ride, services to chart routes, etc. Some of these services originate within the ridesharing business and some are external services that the ridesharing business leverages via digital ecosystems. To the end user, these services must produce a seamless experience, with robust security and low friction at each point in the process. Effectively securing all of these interactions is in many ways an exercise in moving outside the network perimeter to manage and secure APIs wherever they are.

Likewise, government agencies as well as large corporations have similarly exposed their systems—including customer data in many cases—due to mismanaged APIs. It's dangerous when even a single user's credentials are compromised, as might happen in the case a phishing attack—and it can be exponentially worse if a misconfigured API compromises data for millions.

Historically, many enterprises applied management and security to only a subset of APIs—e.g., those shared with internal partners and hosted behind the corporate firewall (within a walled garden, for example). But because network perimeters no longer contain the experiences that drive business, enterprises should think of each API as a possible point of business leverage and a possible point of vulnerability. All APIs should be managed and secured, regardless of where they are located—there are no gardens or firewalls, just billions of interactions that need to be protected.

Leveraging APIs to Apply Security to Every Interaction

At too many enterprises, APIs are not uniformly designed, cataloged or managed, meaning that one team sometimes duplicates APIs already created by another team, that APIs created by one team cannot be easily understood and leveraged by developers elsewhere, and so on. Well-designed APIs should be consistent, intuitive, and well-documented in order to make them easy for developers to consume.

For this reason, APIs should be thought of not as "just" middleware but rather as products that empower developers to work agilely and responsively, leverage digital assets more easily, and experiment and iterate more quickly. Many successful enterprises have formed API teams led by an API product manager to help ensure that APIs are consistently designed for intuitive developer consumption, well-supported with documentation and other resources, updated and maintained based on user patterns and changing business needs, and protected against threats.

Don't neglect the basics

Following some basic best practices will help to establish a strong software architecture from the start and to mitigate future complications. Beyond rudimentary testing, it can be important to conduct code and security reviews, for example, both to increase the likelihood of finding vulnerabilities before they affect customers and to ensure that security defects produced in one part of an API program are documented and don't reoccur in another part.

IT professionals should also be aware that many security problems are found by helpful API users—which means it is important to establish channels for users to report issues, to develop fixes and roll them into production, and to check with the person who filed the bug in order to confirm the issue has truly been resolved.

Balance protection and ease of use

From customer data to inventory information and from legacy applications to microservices, APIs expose a company's valuable data and functionality, making them analogous to the doors and windows of the enterprise. Just as putting a gate around a house doesn't diminish the necessity of locks on those doors and windows, each API is a potential access point that needs to be secured.

The optimal amount of security isn't always easily struck. Not so long ago, "needs to be secured" was synonymous with "lock it down" in many IT circles, but this rigid control often isn't tenable anymore; an enterprise's developers need to move quickly and responsively, leaving little room for heavy-handed security policies and restrictive access processes that may take days or weeks to complete. Now that business is fueled by digital ecosystems and trillions of interactions across billions of endpoints, an enterprise can't protect itself by impeding developers—that defeats the whole point.

Fortunately, many enterprises have moved on from this approach, using API management to mediate access to APIs, monitor their use, and automate developer sign up, onboarding, authentication, and education.

Even so, it is still common for businesses to apply these precautions to only some of their APIs, whether due to uneven governance, shortcuts taken to reach aggressive sprint deadlines, the belief that bespoke APIs not intended for reuse do not require management—there are many other possible reasons. This partial approach is akin to putting locks on the windows that provide a glimpse of one's most impressive valuables—while leaving doors unprotected.

Authenticate the right users

Properly-secured APIs should provide authentication for both end users and applications.

OAuth is the de facto open standard for API security, enabling token-based authentication and authorization. It enables end users and applications to gain limited access to a protected resource without requiring the user to reveal their login credentials. For APIs specifically, OAuth allows a client that makes an API call to exchange some credentials for a token, and that token gives the client access to the API. If an end user allows her social media account to log into a newspaper subscription, for example, OAuth is likely being used to facilitate the transaction, using a token while keeping passwords secret.

A token uniquely identifies a single application on a single device, making its scope limited compared to a password's. Though OAuth is straightforward in principle, correctly applying it can be difficult, so API teams should familiarize themselves with OAuth's capabilities and the current authentication best practices. API teams should consider building OAuth authentication of end users into all critical applications.

In addition to authenticating users, OAuth also supports the concept of application credentials, which requires an API to verify both the user and the application before returning a call, thereby facilitating server-to-server transactions. This does not provide absolute protection, as anyone with application credentials can use and potentially abuse them—which is one reason API monitoring and other management capabilities are also needed to keep APIs safe. Even so, application credentials provide an important layer of security, fortifying the flow of trust connecting the end user, the application, the OAuth authorization server, and the protected asset the application is requesting via an API.

OAuth is made up of a complex family of specs, and there are many ways to use it. Many businesses rely on API management platforms to generate OAuth tokens, granularly control what a token is allowed to do, and apply concepts such as role-based access control (RBAC), which assigns different levels of API access and privileges based on built-in user types.

Use input validation to thwart injections and other common attacks

Input validation helps an application to verify (before sending any data) that user input matches expected parameters. Applying input validation to APIs is one of the best defenses against SQL injections, cross-site scripting, and other common threats—and improperly configured validation has played a role in some of the noteworthy breaches mentioned above.

Well-designed and managed APIs should leverage an OpenAPI specification or a Swagger spec to clearly define input and output and to deploy APIs with uniform input validation processes. Applying inconsistent input validation to APIs is akin to putting different types of doors on a single building—some with push handles, some with rotating knobs, some with deadbolts, some with simple latches, some with redundant locks, some without, and so on. Just as a thief casing such a building would systematically assess the different door types for weaknesses, hackers and other bad actors will look for differences in input validation in hopes of sniffing out a vulnerability.

Deploying shared libraries can be an effective way to achieve consistency, and some API management platforms offer policies that enable easy application of input validation. Robust API management capabilities monitor validation behavior and offer reporting so that when an attack happens, the enterprise can quickly get an understanding of the culprit's methods. Legitimate applications and well-meaning API users are unlikely to violate input validation policies—so by watching the people whose behavior can't be validated, API teams can more quickly understand when and how someone is trying to penetrate their systems.

Encrypt sensitive data with TLS

The cryptographic protocol that ensures API calls use `https` instead of `http` is called "transport layer encryption," or TLS. It establishes an encrypted link between server and client—that is, the link that secured APIs mediate to form connected experiences. Protecting network traffic using an encrypted channel is the easiest way to ensure sensitive information is not susceptible to attacks while in transit—which means TLS is in many ways the foundation of API security.

Monitor and analyze API traffic

An enterprise cannot secure what it cannot see—so monitoring is an essential part of protecting APIs and the connected experiences they support. Many security issues aren't detected by security professionals but rather when the operations team notices something peculiar in the course of monitoring traffic to keep services healthy.

Similarly, while an API team can protect digital assets by simply watching for signs of abuse and revoking access from users that break the rules, it can take its security (as well as its ability to keep APIs online) to another level by mining API traffic for patterns and effectively conveying findings to key audiences.

Monitoring capabilities focus on the present tense, looking after streams of continually updated data and enabling IT operations teams to make near real-time decisions. They are analogous to a heart rate monitor that lets users distinguish when their heart is healthy from when they need to call an ambulance, or a security system that lets the user distinguish when a cat has triggered a motion detector light from when a break-in is occurring.

Analytics, in contrast, are more backward-facing. The goal is less to understand the present tense than to understand what is trending and to use that understanding to improve decision-making. In terms of maintaining API availability, analytics capabilities are less like a heart monitor than a team of physicians, personal trainers, and dieticians making personalized, data-driven lifestyle and fitness recommendations. In terms of security, analytics capabilities aren't so much a home security system as a team of security consultants helping a business to understand why thieves have had so little difficulty with previous security systems.

These two distinct treatments of the same traffic data mean that API monitoring and analytics solutions need to support a range of use cases, including alerting, diagnostics, predictive analysis, and actionability. It's not useful, for example, if the operations team gets an alert that traffic is spiking but can't quickly identify whether an API is overloaded because of a DDoS attack, a surge in legitimate traffic or a systemic problem. It's certainly not useful to make hundreds or thousands of APIs available to internal and external developers without keeping track of which ones are being adopted, which ones might consequently represent new business opportunities, and whether any are being used in unsanctioned ways that could threaten the business and its customers.

An API management platform can help businesses implement this range of monitoring and analytics capabilities because it enables the same platform to handle not only monitoring and analytics but also actions taken based on that data and analysis. Some third-party monitoring tools can be bolted onto an API gateway, for example, but these tools generally issue periodic API calls to check for availability, without any of the more granular visibility, detailed analysis, intelligent automation, and powerful actionability that can be achieved when monitoring is a core part of the management platform.

Business relies on a company's ability to unite many discrete digital assets into a cohesive connected experience—and partial visibility and hamstrung actionability aren't a sufficient approach to keeping those experiences online and secure.

Use rate limits to maintain control

API teams should always consider using rate limits for additional API security. Any API could be subject to a brute-force attack: a technique in which bad actors use automated software to generate a large number of consecutive guesses in an attempt to gain access to protected data. Such attacks may be inevitable for successful digital enterprises—in some ways, it's a cost of doing business. Rate limits help these businesses to keep control of their digital assets and protect customer experiences and privacy when the attacks come.

API management platforms that support rate limits enable an API team to establish thresholds at which spike arrests are triggered, helping to keep backends from being impacted by unexpected activities. An API team might establish that no one should call an API more than 500 times per second, for example, or that an application be allotted only a certain number of API calls per day.

Detect and block bad bots

Like brute force attacks, attacks from bad bots are in some ways the cost of being successful in a connected, digital ecosystem. The more businesses use software to automate connections, the more bad actors will attempt to leverage automation themselves. For example, an attacker who has compromised an API key may be able to deploy thousands of bots that pose as "users" trying to buy products, log into loyalty accounts, or impact marketplace rankings and prices.

Though "bad bots" are a threat, many automated programs are "good bots" that are critical to API-powered connected experiences. It is essential that businesses not only manage API keys carefully but also be able to discern from traffic patterns whether API calls are coming from legitimate users or criminals.

Prioritize usability when assessing dashboards and reporting capabilities

Connected experiences unite digital assets that may be distributed across a range of geographies, public and private clouds, and API providers. One goal of effective API management and security is to control this distributed architecture in a unified way. Many solutions provide monitoring and analytics capabilities that can be triaged together into reports—but any gaps between can leave the business at risk. As noted above, the health of an API program often relies on effective handoffs between security and operations teams, and if reporting capabilities can't make clear when a situation calls for one team versus the other or can't facilitate easy, quick collaboration, the data and insights in those reports isn't worth much.

Consequently, it is important to assess not only whether an API management platform provides capabilities such as monitoring and analytics but also whether the integration of these capabilities will actually accelerate solutions to business problems. Effective reporting and dashboard capabilities should provide at-a-glance insights as well as the ability to drill down for more granular detail. It should be simple to see changes in traffic day-to-day or week over week, to view traffic patterns across chosen time periods, to access information about change management and governance data, to view policy configuration data on a per-API and per-proxy basis, and much more. Having a capability is one thing—being enabled by the capability to solve problems is another.

Protect the Business—Don't Block It

Protecting APIs and the connected experiences that they deliver isn't a matter of just throwing a single switch; rather, API security is a holistic effort that relies on many different capabilities working in concert and involves many different roles within the API team. Businesses should consider a range of capabilities when assessing the strength of their API security:

- **Governance and compliance:**
  - ☐ Does the security meet regulatory standards, apply encryption, and avoid basic complications, such as exposing user data?
- **Authentication:**
  - ☐ Does the enterprise control who can access APIs?
- **Automated bot protection:**
  - ☐ Does the enterprise control not only who can access APIs but also what the user is allowed to do with them?

- **Detection of anomalous behavior:**

  - ☐ Can the enterprise automatically detect suspicious behavior, such as a North American user who has never logged in from a mobile device suddenly logging in from a smartphone in another country and attempting to make a large purchase?

- **Protection of backends:**

  - ☐ Can the enterprise detect problems and automatically apply solutions, such as spike arrests, before customer experiences deteriorate?

- **Optionality and at-a-glance actionality:**

  - ☐ Can the enterprise not only isolate problems but also quickly remediate them?

  - ☐ Are dashboards and reporting capabilities equipped for handoffs between operations and security teams and to provide at-a-glance insights that can inform near real-time decisions?

Beyond simply checking off the above requirements, businesses should aim to apply light-touch management and security processes. The goal is to protect APIs without compromising the agility and freedom that APIs provide developers and that have made APIs, and connected experiences, popular in the first place.

A business's digital ecosystem goals may not advance much, for example, if it applies OAuth but continues using manual approval processes that require developers to wait weeks or months between requesting an API and actually accessing it. Security is an important part of enabling connected experiences—but so is empowering the developers who build those experiences.

Many enterprises attempt to establish the right mix by establishing a self-service developer portal where developers can quickly access APIs but that still gives the enterprise control over and visibility into that access. The portal should include documentation and sample code to help developers get started, testing environments in which developers can begin experimenting, and potentially even monetization options that enable developers to pay for API access or to share profits from their apps with the API provider whose technology they're leveraging.

Security and agility must be balanced, in other words, and businesses that want to profit from connected experiences should not prioritize one to the exclusion of the other. Each interaction point should be secured, but each point must also leave participants the flexibility to interact. Businesses that look as APIs as exercises in both developer enablement *and* enterprise security put themselves in the best position to grow in today's connected landscape.

About Apigee API Management

Google Cloud's Apigee API management platform delivers full lifecycle API management to help businesses unlock the value of data and securely deliver modern applications and digital experiences. Apigee offers a rich set of capabilities to enable enterprises to gain control over and visibility into API traffic, including the ability to automate troubleshooting and problem resolution and to derive insights from API usage. Learn more about Apigee's API security capabilities.

Now that you've finished reading, why stop learning?
Visit the Apigee website for more.

apigee

Share this eBook

on social

with a colleague

Google Cloud