How to Build an Effective API Security Strategy

Published 8 December 2017 - ID G00342236 - 20 min read

A security strategy that manages access and protects systems from attack while still engaging digital ecosystems is essential to any API program. Application leaders must design, execute and govern an effective API security strategy, including the use of API gateways.

Overview

Key Challenges

* Attacks and data breaches involving poorly protected application programming interfaces (APIs) are mounting.
* Protecting web APIs with traditional application security solutions alone is ineffective.
* New APIs are being added and consumed by organizations on an ongoing basis, meaning that API security is not a one-time exercise.
* Modern application architecture trends — including mobile devices, microservice design patterns and hybrid on-premises/cloud usage — complicate API security since there is rarely a single "gateway" at which protection can be enforced.

Recommendations

Application and application security leaders responsible for application strategies and governance should:

* Discover and assess the organization's APIs to ensure they are covered by an overall web application security architecture. You cannot secure what you cannot find or categorize.
* Take a capabilities view of API security before implementing it in infrastructure such as API gateways.
* Adopt a continuous approach to API security with ongoing discovery, monitoring and securing of APIs.
* Use a distributed enforcement model to protect APIs across your entire architecture, not just at the edge.

Strategic Planning Assumption

By 2022, API abuses will be the most-frequent attack vector resulting in data breaches for enterprise web applications.

Introduction

In their approach to API security, organizations exposing web APIs must balance ease of access (to ensure adoption of APIs) with control (to prevent abuse or attacks). Like the bank robber attacking banks because "that's where the money is,"[1](https://www.gartner.com/doc/reprints?id=1-6OFZ2S3&ct=190514&st=sb#dv_1_fbi_history) the use of APIs to provide access to applications and to business-critical data has naturally led to API security incidents. These have occurred particularly in the form of data breaches (see Note 1). The growth of open banking APIs, in particular, is set to make this picture worse.

API gateways, combined with web application firewalls and other application security infrastructure where necessary, are used to implement API security. However, a purely edge-based web application security defense strategy is not fully ready for the new challenges posed by APIs. The widespread use of internal APIs, combined with mobile access and increased reliance on cloud APIs, mean that defending from the edge is insufficient. New hybrid approaches highlight the fact that organizations should take a holistic view of API security (see "Defining Cloud Web Application and API Protection Services").

The best practices described in this research explain how an organization should use API security to enable its integration and digital business initiatives.

Analysis

Discover and Assess Your Organization's APIs to Ensure They Are Covered by Your Overall Web Security Architecture

Just as APIs are a part of application delivery side-by-side with the delivery of mobile and browser interfaces, API security should also be a part of your overall application security architecture. Begin by identifying the current capabilities of the vendors whose products are used in your organization. These could include web application firewall (WAF), application delivery controller (ADC), API management and content delivery network (CDN) software. Survey these vendors about their short-term roadmaps. Do they have, or are they adding, API security capabilities? If not, consider adding new infrastructure from a vendor which is stronger on API security. Consult research notes such as the "Magic Quadrant for Web Application Firewalls," which includes details of vendor API security capabilities.

Standards are key to ensuring that API security works with your organization's overall application security architecture. In particular, ensure that identity standards (such as OAuth 2.0, SAML and OpenID Connect) are supported, since these are used to interface with identity and access management solutions, so that a new identity silo is not required.

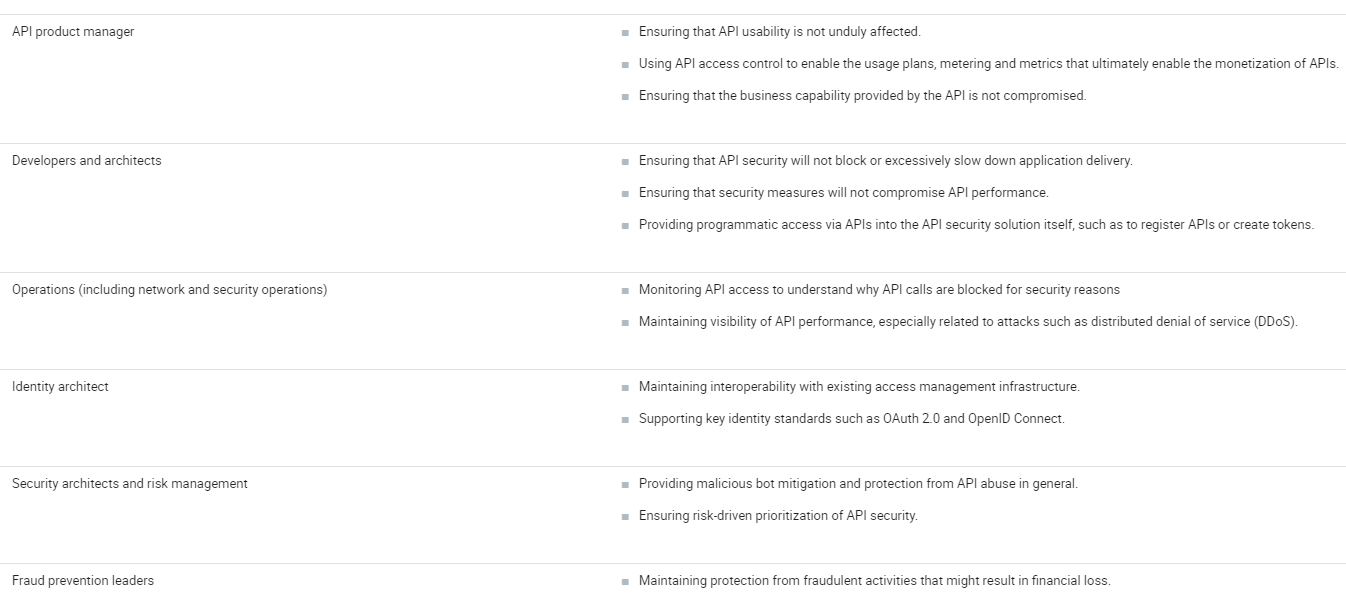
Also take into account the APIs that are consumed by your organization. These can introduce security concerns, such as data leakage (see "Establish Governance of External APIs to Avoid Unpleasant Surprises").

Involve Every API Security Stakeholder

Common stakeholders for API security are shown in Table 1 below. All of these stakeholders must be taken into account when formulating API security policies.

### Table 1: Key API Security Stakeholders

Stakeholders What is important for the stakeholder?



Source: Gartner (December 2017)

As shown in Table 1, a key API security stakeholder is the API product manager, who is also essential for the success of an API program. Gartner's CIO Survey for 2018 found that, among the top digital performers, 42% have created this role, compared to only 6% of trailing digital performers (see "The 2018 CIO Agenda: Mastering the New Job of the CIO"). API security enables API product managers work with business teams to identify where API access control can enable tiered levels of API access. In some cases, this may be linked to API monetization though most APIs will not be directly monetized (see "Choosing the Right API Pricing [and Funding] Model"). API product managers must also make sure that API security measures do not make APIs difficult to understand and use. As such, they act as a focal point for API security decisions (for more on the API product manager role, see "Create the Role of API Product Manager as Part of Treating APIs as Products"). As also shown in Table 1, security leaders should also be involved, as well as fraud prevention leaders, working in conjunction with the API product manager.

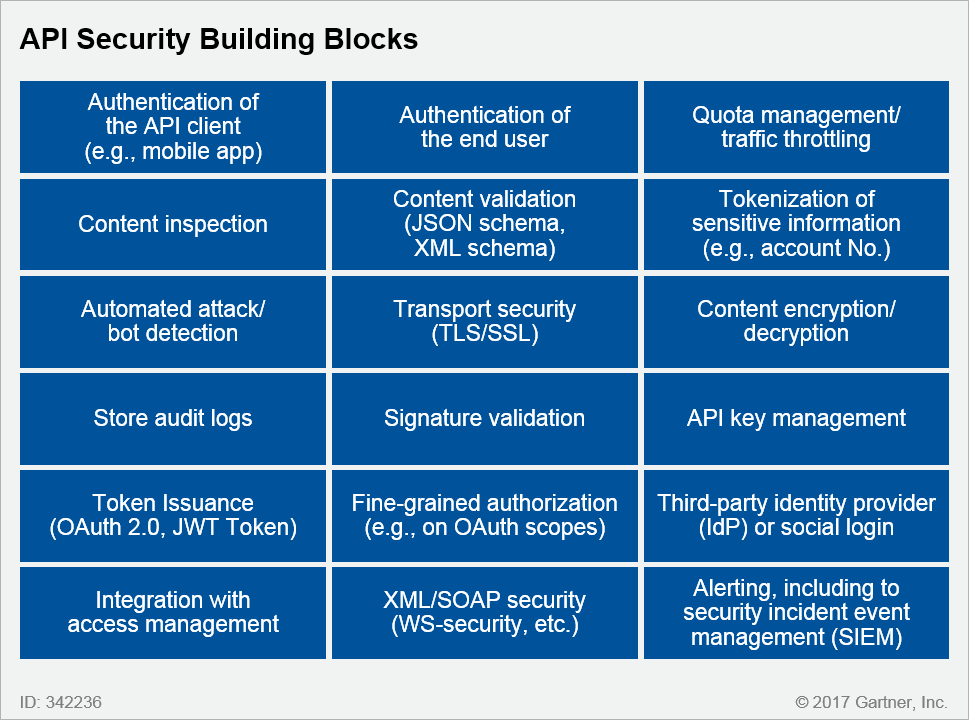
### Take a Capabilities View of API Security

API gateways in particular are powerful tools to provide API security because they support a wide range of features. However, these can be overwhelming in scope.

As shown in Figure 1, there are multiple "building blocks" which may be used in API security policies. Although many of these building blocks are self-explanatory, in some cases they have a specific purpose in the context of API security. For example, signature validation may be performed in the context of client authentication using API keys. This happens when a signature (or digest) has been created over a REST API call (using an API key) by the client in order to show "proof of possession" of that API key.

Figure 1. API Security "Building Blocks"

Source: Gartner (December 2017)



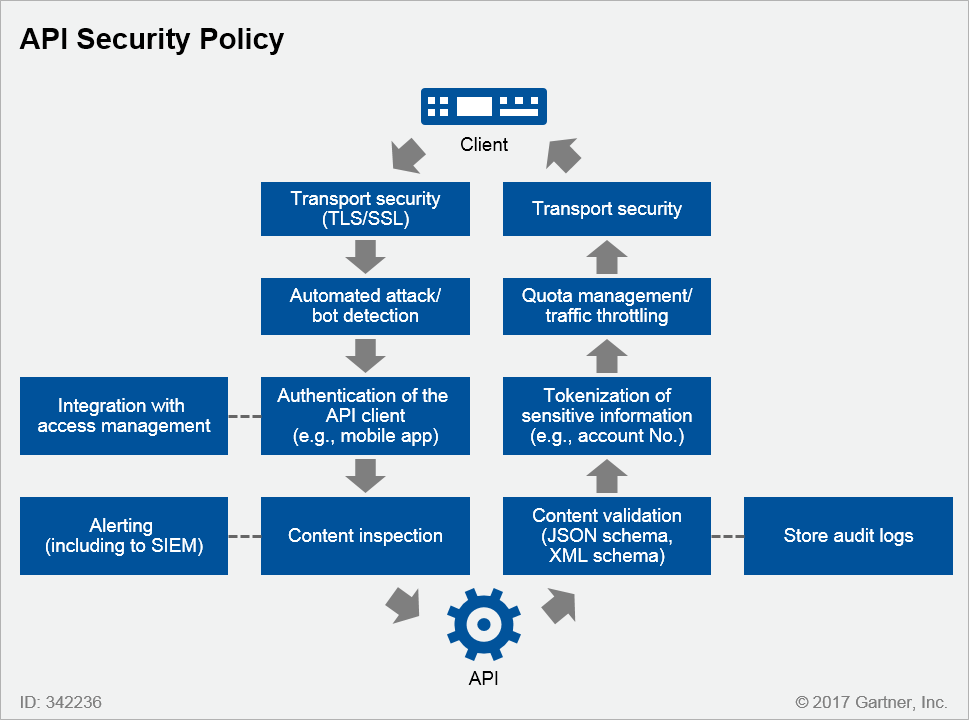
To avoid being overwhelmed by the features of API security products, and buying a product with the most features instead of the product you really require, instead take a capabilities approach to API security. The term "policy" is often used to describe the workflow involved in API security. Diagram your required API security policy as a sequence of steps, using some of the API security building blocks in Figure 1. You can then map this to product capabilities in your current infrastructure, or even use it as input to a proof of concept (by asking a vendor how this API security policy would be configured using its product, for example).

Figure 2 shows an example of such an API security policy. Notice that it brings in attack protection as well as authentication. The policy also acts not only on the request to the API, but also on the response. Sensitive data is tokenized before it is returned to the client, and the response is validated to ensure it is appropriate. The dotted lines show where links to other products are used, for example, to identity and access management infrastructure.

Figure 2. Example API Security Policy Shown as a Sequence of Steps

*API = application programming interface; SIEM = security incident event management; SSL = Secure Sockets Layer; TLS = Transport Layer Security; XML = Extensible Markup Language*

Source: Gartner (December 2017)

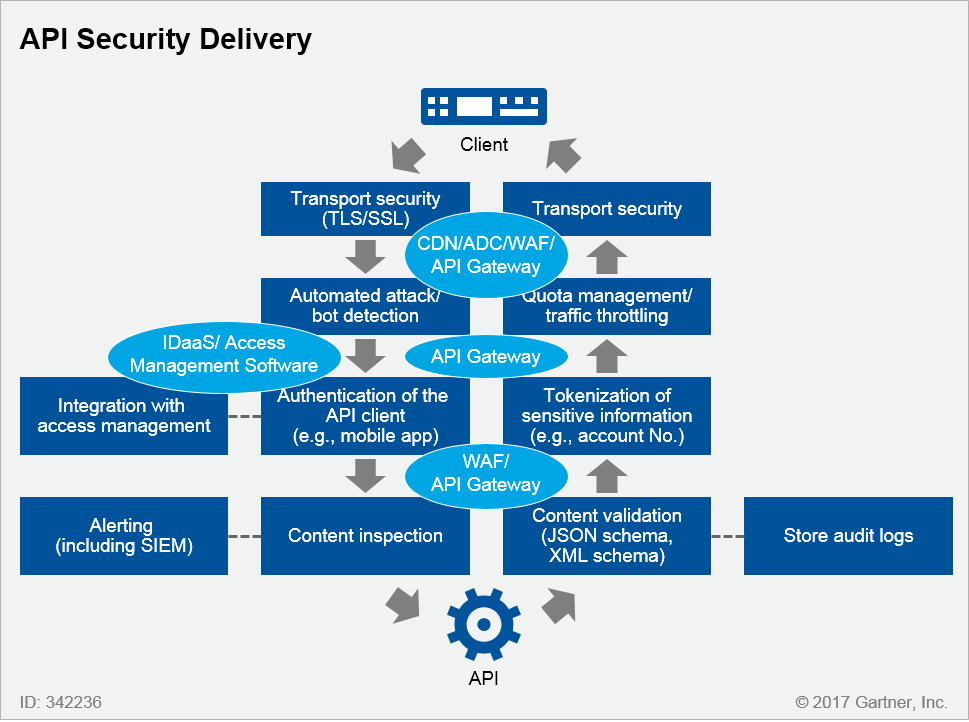


A policy such as that shown in Figure 2 is typically implemented using an API gateway, often as part of a full life cycle API management solution that combines API gateways with an API developer portal. But other products in your application security infrastructure can also be leveraged. Figure 3 overlays product categories onto the API security policy from Figure 2.

Figure 3. Products Delivering API Security

*ADC = application delivery controller; API = application programming interface; CDN = content delivery network; IDaaS = identity and access management as a service; SIEM = security incident event management; SSL = Secure-Sockets Layer; TLS = Transport Layer Security; WAF = web application firewall; XML = Extensible Markup Language*

Source: Gartner (December 2017)



An application delivery controller (ADC) is already in use in many organizations for Transport Layer Security (TLS) termination. It may also be used for transport security for APIs. Integration with access management may involve connections to access management services or software. Examples include identity as a Service (IDaaS), such as Okta, (which also offers API access control) or access management software from vendors such as ForgeRock and Ping Identity, both of which offer API access control. Content inspection may use a web application firewall (WAF) such as Imperva. WAF capabilities for APIs may also be included within an application control (ADC) or content delivery network (CDN) solution (such as Akamai, which also provides API protection).

Bot-based attacks are particularly concerning for APIs, because APIs are, by design, intended to be called programmatically. It is therefore important to distinguish "good" API calls from "bad" API calls that could be attempting to harvest data, for example. Bot mitigation may be provided by a solution that is already in place to perform bot mitigation for web traffic (for example, Distil Networks or ShieldSquare) or through an API management solution that includes bot mitigation (such as Apigee Sense from Google).

By using a capabilities approach, you may find that an API gateway is sufficient to deliver your required API security policy. But if a capability is required that is not provided by the API gateway, such as advanced bot mitigation, then check the capabilities of other products in your infrastructure to provide this.

Use Infrastructure Rather Than "Baked In" Security

Do not code API security policies into the APIs that you wish to protect. Coding security directly into APIs has the following disadvantages:

* Violates separation of duties.
* Makes code more complex and fragile.
* Adds extra maintenance burden.
* Is unlikely to cover all aspects that are required in a full API security policy.
* Not reusable.
* Not visible to security teams.

Furthermore, developers may propose a purely whitelist approach. However, these whitelists are typically too broad, and security teams will typically have no visibility of their usage or management.

As Gartner's "2018 Planning Guide for Security and Risk Management" notes, externalized capabilities such as API gateways, WAFs and CDNs, are "ideally suited for certain security functions that are difficult to implement and maintain, that are less effective, or that lack flexibility when part of application code." We are starting to see a number of innovative startups appearing in the API security space, including 42Crunch, Elastic Beam and Secful among others.

Because mobile clients are often used to call APIs, attackers may target an API by reverse-engineering a mobile app to discover how it calls the API. If API keys are "baked in" to the app, this can result in an API breach. Indeed, API keys should not be used alone for client authentication, without another form of authentication or client IP restriction. See the Protect your APIs section of "Securing Mobile App Back Ends" for recommendations on protecting mobile API access.

In addition to authentication of the API client, many enterprises will bind the client to a specific app or device. This can be achieved using application authentication solutions (such as Approov from Critical Blue or the CA Mobile API Gateway), or by using an application shielding solution (see "Market Guide for Application Shielding"). Access management tools and services are also increasingly able to use device context to render an access decision. Client IP address, geolocation or other context gathered from the accessing device can be used by the access management policy.

Truly "Open" APIs Need Security Too

APIs are increasingly used by governments to deliver data or to integrate with service providers (see "Add Full Life Cycle API Management to Your Digital Government Platform"). These APIs often take the form of "open data" APIs, delivered as an alternative to providing datasets as files, and providing real-time data such as transit information. It may be tempting to think that these APIs do not need security. However, it is important to use traffic throttling so that clients downloading data in bulk via an open API (a common occurrence) do not have an adverse impact on other clients or increase cloud consumption (in the case of cloud-delivered APIs). In addition, clients may be directed to sign up for the APIs through a developer portal, and then use API keys to authenticate (note that API keys alone should not be used for API security, since attackers will reverse-engineer client applications to obtain API keys).

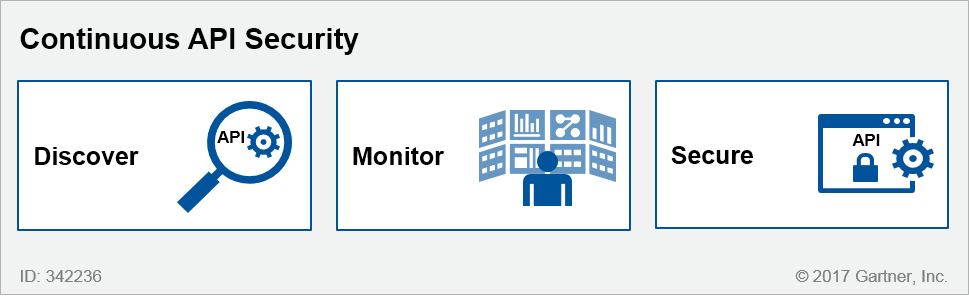
Adopt a Continuous Approach to API Security

API security is not a one-time project. Take the three steps illustrated in Figure 4 to ensure a continuous approach to API security.

Figure 4. The Three-Step Approach to Continuous API Security

*API = application programming interface*

Source: Gartner (December 2017)



Discover:

* Ensure visibility across teams by making sure that security, quality and development teams get access to API reports (from API management software, for example). Determine how your organization's change management process should be updated to inform relevant stakeholders when implementing a new API, or modifying an existing API.
* Continuously inventory APIs that are delivered by the organization, or that are in development. APIs that the organization consumes from third parties should be included in this inventory. An API catalog, typically provided by a full life cycle API management solution, can be used for this purpose (see "A Guidance Framework for Evaluating API Management Solutions"). CDN products, which have visibility into web traffic that may include API traffic, can be used to dynamically discover APIs.
* Integration with the software development life cycle, and in particular application development life cycle management, allows planned development efforts for new APIs, or changes to existing APIs, to be accounted for upfront rather than inventorying them later (see "Managing Digital Trust in the Software Development Life Cycle"). This becomes critical for DevOps and DevSecOps so that appropriate API security policies can be applied as APIs are developed.

Monitor:

* Quiz vendors on their ability to identity normal behavior and anomalies in API usage, especially for cloud-delivered APIs.
* Monitor external third-party APIs consumed by the organization. This may be performed by the "reverse gateway" capability provided by some API management vendors, to apply policies to API consumption (not just API delivery).
* Categorize APIs based on monitoring the data and applications they can access, whether they are business critical, and their client usage profiles.

Secure:

* Apply policies to APIs (for example, using an API gateway) but avoid situations where each API has a unique security policy. Instead, leverage a reusable set of policies that are applied to APIs based on their categorization. Abstract any specific API characteristics (such as URL path) from the policies themselves.
* Apply API security throughout the API life cycle, including application security testing (AST) of new versions (see "How to Integrate Application Security Testing Into a Software Development Life Cycle").

Good API security also requires change control throughout the entire application life cycle. For example, only certain developers should be allowed to make changes to the API, and application security testing is performed whenever an API change is made during development. Undocumented changes to APIs can also have serious implications to the availability and security of the overall system (for example, some protections may stop working as expected if the API changes).

Use a Distributed Enforcement Model to Protect APIs Across Your Entire Architecture, Not Just at the Edge

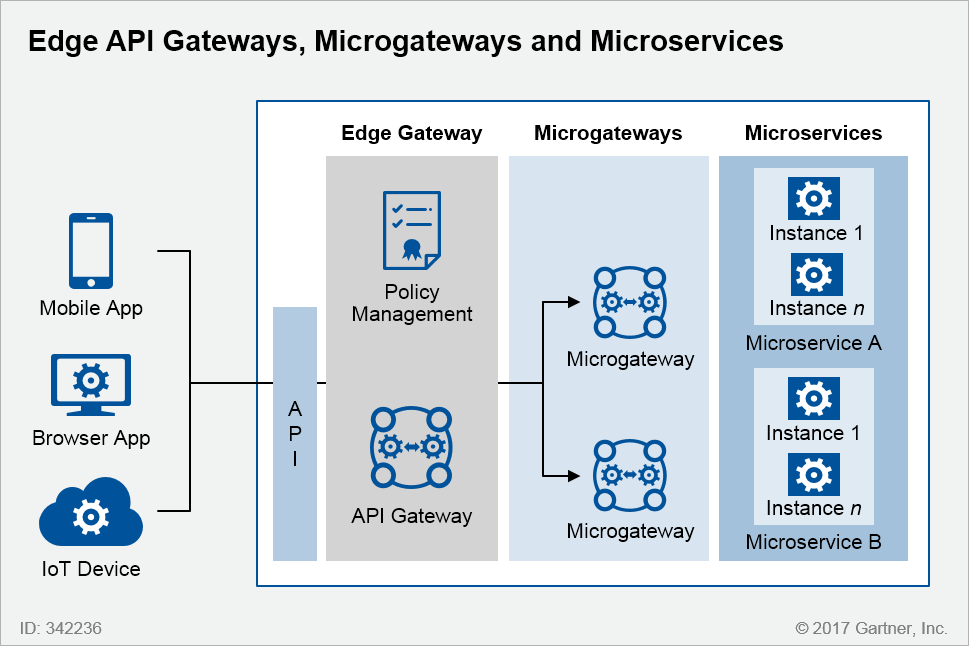
API security has traditionally focused on the "north-south" traffic between an API client and the API itself. However, the use of microservices, in particular, brings a requirement for "east-west" security between microservices, or APIs consuming other APIs and services, which typically will not call each other through an edge API gateway. This has led to the development of "microgateways," which can enforce policies closer to microservices themselves (see "Emerging Technology Analysis: Full Life Cycle API Management" for more on the impact of microgateways on API management solutions).

In the Figure 5, we see that security must be applied not only at the edge API gateway, but also for the traffic within the microservice architecture itself (see "Innovation Insight for Microservice Infrastructure"). Note also that, even though microservices may be called via APIs, they may also be event-driven (see "The Impact of Event-Driven IT on API Management").

Figure 5. Edge API Gateways, Microgateways and Microservices

*API = application programming interface; IoT = Internet of Things*

Source: Gartner (December 2017)



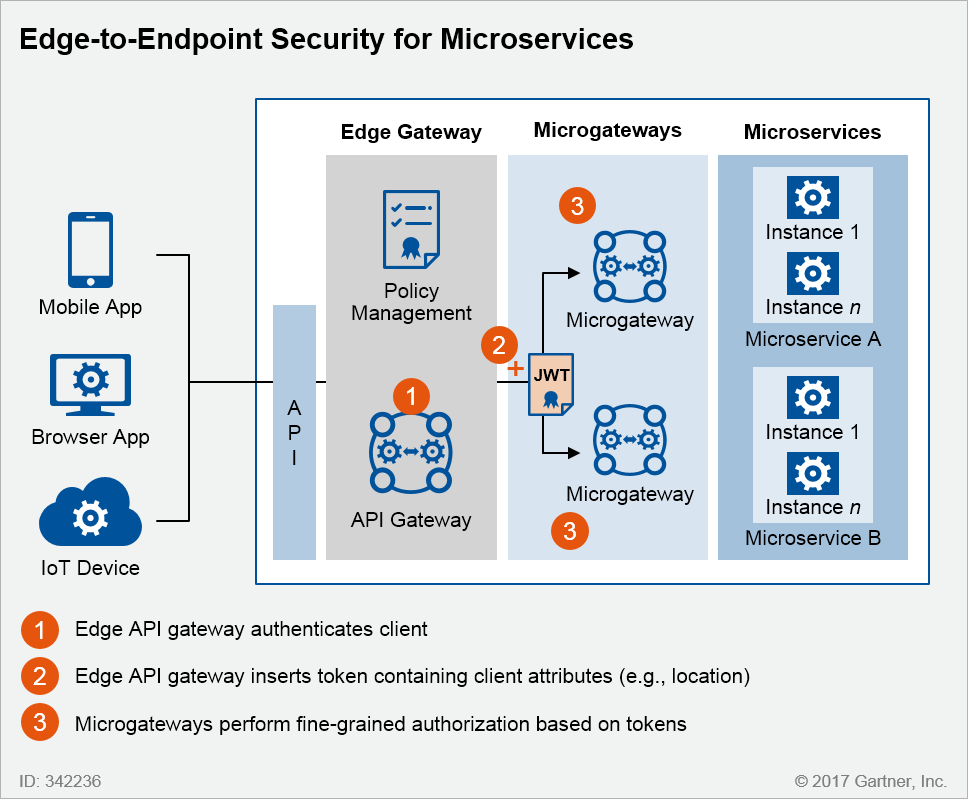
Edge-to-Endpoint Security

To account for microservice architecture, consider an "edge to endpoint" approach, as shown in Figure 6 below.

Figure 6. Edge-to-Endpoint Security for Microservices

*API = application programming interface; IoT = Internet of Things*

Source: Gartner (December 2017)



As shown in Figure 6, the external API Gateway performs authentication (as well as other capabilities such as content inspection), then "injects" security context into the API calls using standards such as JSON web tokens. Microgateways can then use this context information, including attributes of the API client such as their location, to perform fine-grained authorization. Optionally, this fine-grained authorization can make use of externalized access management products such as Axiomatics Policy Server. If the API calls are authorized, then the microgateway passes them through to the microservices themselves. This provides the benefit of edge-to-endpoint security. Another benefit of this architecture is that microservice-to-microservice traffic does not need to go through the external gateway layer, and can also make use of JSON web tokens.

Next Steps

As we saw in Figure 4, the first step to API security is to understand the APIs that are created and consumed by your organization. To choose the appropriate tools to secure these APIs, use a capabilities approach to identify the steps in your API security policy, and then map these to the products that deliver these capabilities (as shown in Figures 1, 2 and 3).

Ensure that all of the stakeholders shown in Table 1 have visibility to this process. If necessary, use examples of breaches (we have listed a selection in Note 1) to demonstrate the importance of API security to stakeholders. Finally, resist the temptation to build API security yourself, and instead use off-the-shelf products (some of which may already be deployed by your organization) to obtain the benefits of API security.

Case Study

In the banking industry, third-party technology companies (often financial technology startups known as "fintechs") offer apps that provide services to end users such as personal financial management or account aggregation. These apps often interface with banks through what is also commonly called a "screen scraping" approach. Note that this use of the term screen scraping is not the same as the older usage for optical character recognition (OCR) of a user interface. Instead, it works in this way:

1. Users enter their online banking credentials into the third-party app.
2. The app provider connects to the online banking systems using those credentials.

Because these apps can connect to multiple banking and investment services on behalf of the user, the user's credentials are typically stored by the app provider. This naturally raises security concerns for banks.

By providing an API, a bank can control this third-party access.

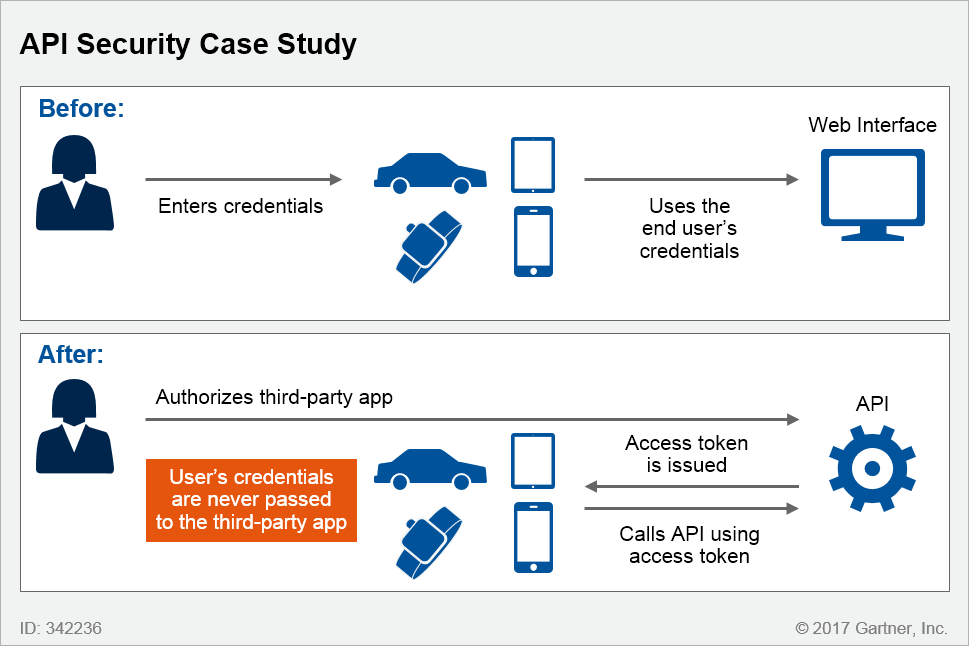
This is done in two ways:

1. App providers, such as fintechs, must sign up to use the bank's API, using an API developer portal.
2. The customer logs into the bank's online systems, and authorizes the app provider to access banking systems on their behalf. This typically uses OAuth 2.0.

This is shown in Figure 7:

Figure 7. API Security Case Study

Source: Gartner (December 2017)



From the bank's point of view, the API approach has an advantage because customer credentials are never passed to or stored by third parties, limiting fraud concerns. The bank can also control access, since the API may not provide the same functionality provided by their online banking systems, and can be limited in its scope. API security, typically using an API Gateway, can be applied. But from the app provider's point of view, the ability to access data may be constrained since the bank is in control of the API. In addition, fintechs may argue that complexity is introduced by requiring users to authorize third-party apps, thus putting disruptors at a disadvantage to incumbents.

Banks are reaching agreement with third parties in order to allow access to data via APIs. An example is Capital One Financial Corporation, which has announced an agreement with the financial management firm Intuit that allows Capital One customers who use Intuit's suite of services (including QuickBooks Online, Mint and TurboTax) to securely import their financial data, through APIs, without sharing login credentials.[2](https://www.gartner.com/doc/reprints?id=1-6OFZ2S3&ct=190514&st=sb#dv_2_capital_one)

A number of banking regulations, in particular PSD2 in the European Union, require banks to open APIs for access to accounts and for payment status (see "Six Steps to PSD2 — Digital Banking Reimagined in Europe and Beyond"). API security policies will be key to protecting the APIs delivered as part of these banking APIs, including supporting different levels of access.

Evidence

Client inquiries, vendor briefings

1FBI History, Famous Cases and Criminals,  [Willie Sutton](https://www.fbi.gov/history/famous-cases/willie-sutton).

2 ["Capital One Financial (COF), Intuit (INTU) Enter Data Sharing Agreement,"](https://www.streetinsider.com/Corporate+News/Capital+One+Financial+(COF),+Intuit+(INTU)+Enter+Data+Sharing+Agreement/13442492.html)StreetInsider.com

Note 1Examples of API Security Incidents in 2017

* ["T-Mobile Customer Data Plundered Thanks to Bad API."](https://arstechnica.com/information-technology/2017/10/t-mobile-website-bug-apparently-exploited-to-mine-sensitive-account-data/)Ars Technica reported that an unsecure API allowed third parties to query account information by simply providing a phone number.
* ["Instagram's Leaky API Exposed Celebrities' Contact details."](https://www.theregister.co.uk/2017/08/31/instagram_leaks_verified_members_contacts_via_api_bug/)The Register reported that Instagram blamed a bug in its API for the partial breach of verified user accounts.
* ["McShame: McDonald's API Leaks Data for 2.2 Million Users."](https://www.bankinfosecurity.com/blogs/mcshame-mcdonalds-api-leaks-data-on-22-million-p-2426)BankInfoSecurity reported that McDonald's acknowledged that an unsecure API exposed personal information for users of its McDelivery mobile app in India.
* ["API Flaws Said to Have Left Symantec SSL Certificates Vulnerable to Compromise."](https://www.csoonline.com/article/3184897/security/api-flaws-said-to-have-left-symantec-ssl-certificates-vulnerable-to-compromise.html)CSO Online reported that unsecure APIs allowed third parties to obtain access to public and private keys, as well as the ability to reissue or revoke certificates.