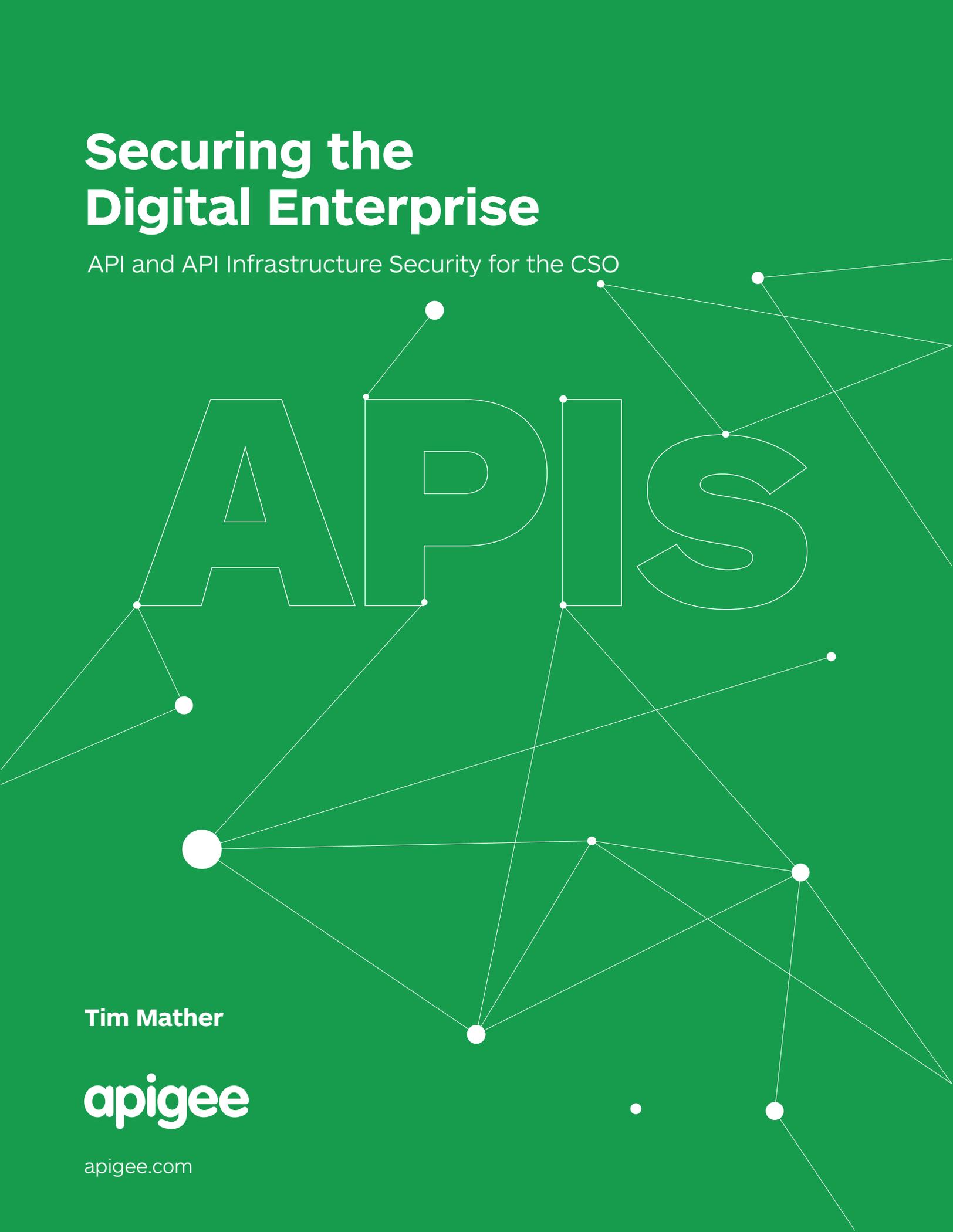


Securing the Digital Enterprise

API and API Infrastructure Security for the CSO



APIIS

Tim Mather

apigee

apigee.com

To a chief security officer (or a chief information security officer), “exposure” might not be a comforting term. But it’s a key tenet of application programming interfaces. APIs expose data or functionality for use by apps and the developers that create them. They make enterprise assets reachable by apps, and they’re the tool that enterprises use to add a digital layer to their interactions with customers, employees, and partners.

In recent years, apps have evolved from the means of content access and data entry to being the primary channels of interaction between a company and its customers and employees. This transformation has significant implications for the enterprise architecture, which now must move from delivering web applications as the primary interaction channel to powering interactions in a secure, performant, and data-leveraged way across multiple interactive touch points, of which the web is just one of many.

APIs expose corporate data in very deliberate and thoughtful ways, but, as with any technology that involves enterprise data, security should always be a prime concern. With the explosive growth in API use, business unit leaders at companies that are undergoing digital transformations are calling upon their CSOs to ensure the security of APIs.

Security must be built into the APIs themselves. But that’s not enough. Threat protection, identity services, infrastructure security, and compliance also must be top of mind for the CSO.

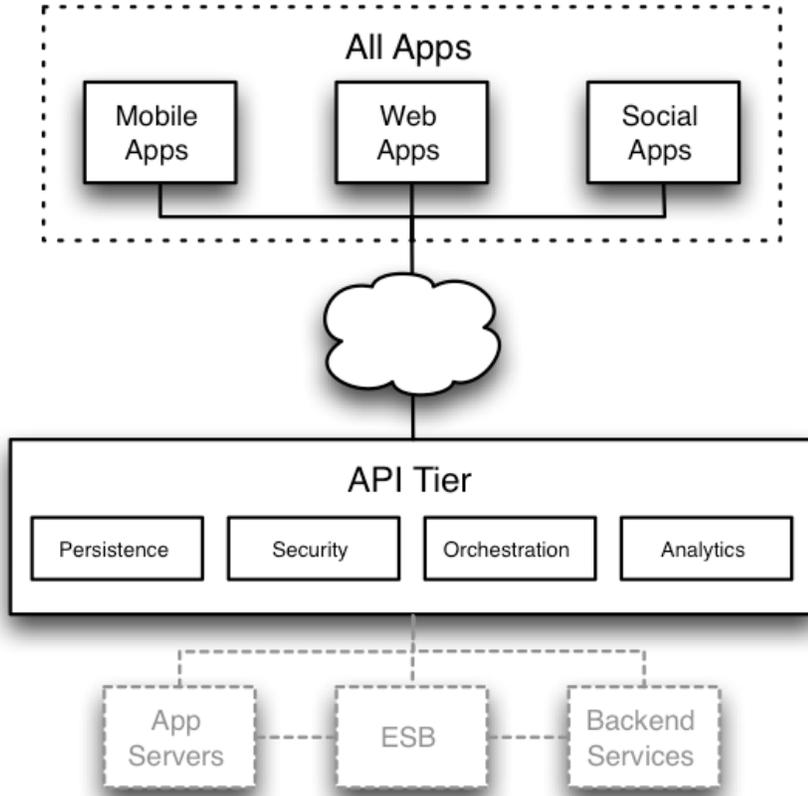
With that in mind, this eBook provides a framework to help CSOs and CISOs consider API security.

API security

API security, and the security of the infrastructures these APIs are running on, is of paramount importance to a CSO at an enterprise that is exposing digital assets.

Exposure entails enabling external access by apps. How do you do that? You build an API tier, either from scratch or with the help of a vendor like Apigee. The goal of today’s API tier is to allow a large number of apps, which are often mobile and created by your partner channels or new application teams, to access content and data from internal systems.

API Tier Architecture



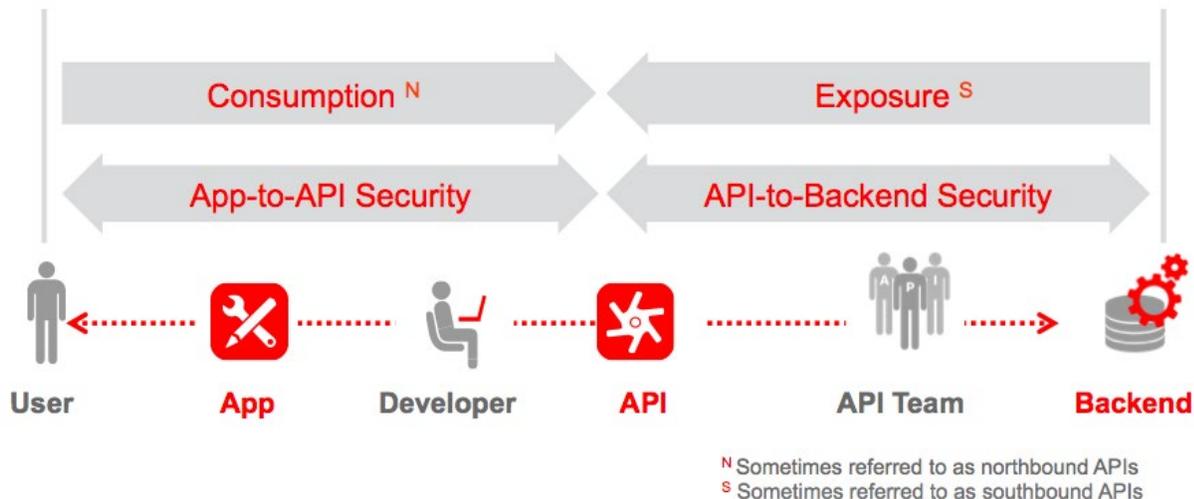
API security governance: One of the key tenets of enabling defense-in-depth security practices within an enterprise is separation of concerns. It requires support for separation of duties between the service providers (the IT architect, IT security, and business) and API service consumers (developers and end users).

A typical API centric architecture is composed of an exposure and a consumption side:

- Exposure** - Rather than having app developers consume your services directly, they access an API proxy. The API proxy functions as a mapping of a publicly available HTTP endpoint to your backend service. By creating an API proxy you let an API infrastructure like Apigee Edge handle the security and authorization tasks required to protect your services, manage data transformations and filtering, execute conditional logic or custom code, and perform many other actions.
- Consumption** - Includes capabilities that enable developers to build and deploy apps in a secure way; engage with a developer community; build compelling and feature rich apps (push notification, geolocation, social, and so on) and help manage application life cycles via self-service APIs and developer portals.



End-to-End Security for the Digital Value Chain



An API-centric security architecture demands industrial-grade security capabilities including RBAC (role-based access control), authentication for users, developers, and administrators as well as APIs for implementing authentication and authorization via OAuth, SAML, and LDAP, fine-grained policy management for authorization, and threat protection against XML, JSON, and DoS attacks.

Authentication

The two-sided nature of API authentication is an important aspect for CSOs to consider. For example, there's the authentication of the application to the API on the consumption side (sometimes referred to as northbound APIs). Then there is authentication of the API to back-end IT systems (sometimes referred to as southbound APIs). Additionally, both the consumption-side and exposure-side communications can be secured by one-way or two-way TLS or mutual authentication. That is, authentication of the server to the client, and authentication of the client to the server.

Besides authentication of the applications that are making the API calls, a key component is people: the authentication of API platform administrators, API developers, and sometimes the users of the apps that APIs are calling. This aspect of API authentication is done through LDAP, two-factor authentication, transport layer security (TLS), or security assertion mark-up language (SAML).

OAuth is an API authorization protocol that enables apps to access information on behalf of users without requiring them to divulge their usernames and passwords. The OAuth 2.0 specification defines different ways of creating access tokens for apps. The various mechanisms for distributing access tokens are referred to as grant types. The most basic grant type defined by OAuth 2.0 is client credentials. In this grant type, OAuth access tokens are generated in exchange for client credentials, which are consumer key/consumer secret pairs.

Alternatively, **API keys** provide a simple mechanism for authenticating apps to APIs (API keys are sometimes referred to as app keys, or consumer keys—they all mean the same thing). An API infrastructure generates API keys and enables developers or organizations to add API key-based authentication to their APIs' use policies. That app key is then embedded in the client app. The client app must present the app key for each request, and the API infrastructure validates the app key before permitting the app's request.

OAuth is the most commonly used method for authenticating APIs to applications. However, some organizations have standardized their authentication on **SAML**; in these cases, SAML support (instead of, or in addition to, TLS) might be an important consideration. Both TLS and SAML are used for broader authentication needs than just APIs, so most CSOs are already familiar with them.

Delegated authentication is another security capability that is often used on the API exposure, or south-bound, side. This enables organizations to retain full control over authentication and authorization data without needing to provide copies to an API infrastructure provider.

The OAuth protocol enables app end users to authorize apps to act on their behalf. Apps do so by obtaining access tokens from API providers. Apigee Edge enables you to configure and enforce OAuth using policies, without requiring you to write any code.

developers, and reports in their organizations. There should be at least two roles: the organization administrator and the user.

In Apigee Edge, an API key validation is the simplest form of app-based security that you can configure for an API. Apps simply present an API key, and Edge checks to see that the API key is in an approved state for the resource being requested. You can revoke an API key for an app at any time—you might need to do so if you find a serious error in an API resource or a security compromise of the key.

Authorization

Beyond authentication, another important consideration in API security is authorization. By far, the most commonly used method of authorization for APIs is OAuth, an open standard. Of course, with SAML assertions, authorization decision statements can also be included. Additionally, API keys can be used for simple authorization purposes. Successful presentation of a key to an API, for example, constitutes authorization to use that app's information.

Role-based access control (RBAC)

CSOs should ensure that an API infrastructure has RBAC, which governs actions taken by users on APIs, API products, apps, de-

For example, in Apigee Edge, organization administrators create and manage user roles, which define permissions for organization entities, such as development, testing, release, and operations. These roles have specific permissions assigned to them:

- GET enables a user to view a single RBAC resource, or a list of them
- PUT enables a user to create or modify an RBAC resource (encompassing both PUT and POST HTTP methods)
- DELETE is only supported on singletons (a single RBAC resource); users cannot delete collections of RBAC resources, regardless of their role

Organization administrators also create users, set up RBAC-protected resources, and assign appropriate roles to manage the API infrastructure.

Logs and audit trails

API security must also provide the ability to log all actions, and the ability to audit such logs. API management infrastructures must provide security auditors with auditing capabilities, including the ability to export audit trails and logs for offline analysis. API infrastructures should also support real-time troubleshooting to help detect API attacks that may be underway.

Advanced capabilities built into Apigee can capture messages sent to your API from apps or other clients (such as consoles and terminals).

Advanced capabilities built into Apigee can capture messages sent to your API from apps or other clients (such as consoles and terminals).

Analytics

API infrastructures like Apigee Edge enable enterprises to gain a deep understanding of customer behavior and interactions using real-time data from APIs. Much of this data is operational, which helps to guide business decisions, but it can also be relevant to monitoring the security of APIs. For example, run-time detection reports, such as temporal analysis of volume and traffic properties by the dimensions (/apis, /apiproducts, /apps, /devs, and /envs), should be correlated against threat detection capabilities, including:

- XML poisoning
- JSON injection
- SQL injection
- quota/spike arrest
- IP address-based access restrictions

Correlations of operational data with threat protection data enables security personnel to rapidly assess risks to their organization's API calls. For example, alerts or regular reporting on the frequency and origin of certain types of errors could alert a CSO

API infrastructures like Apigee Edge enable enterprises to gain a deep understanding of customer behavior and interactions using real-time data from APIs. Much of this data is operational, but it can also be relevant to monitoring the security of APIs.



to attacks involving XML poisoning or a JSON (JavaScript Object Notation) injection.

As traditional security incident and event management (SIEM) platforms reinvent themselves to address big data security analytics (the practice of analyzing data to monitor API security), vendors such as HP ArcSight, RSA Security Analytics, Splunk, and others face a common challenge: collecting all of the data that those platforms must analyze from the multitude of security and operational sensors that organizations use. It's important that this SIEM-related security analytics capability is included in enterprise API infrastructures roadmaps.

There are literally hundreds of these security sensor vendors, some of whom make their security point-product data available via a common log format such as syslog, and others who make the data available via an API. That still leaves vast amounts of security data that must be laboriously ingested into SIEM platforms through the use of connectors or forwarders.

This “forced data ingestion” is inflexible, very costly, and inefficient. There is no common API format for security vendors to write to; in other words, every vendor with an API uses a proprietary format. The problem is compounded by the lack of a common log format. Even syslog, which is supposed to be a common log format, has multiple variations.

What if this problem could be brought under control by your API infrastructure? What if it could be the common clearinghouse of security APIs? This would add tremendous value to the SIEM platforms and the numerous security sensor products available today. With your API infrastructure “clearing” all your security service APIs, the API infrastructure becomes your first point of analysis for all big data security analytics.

Of course, integrating the exposure side of APIs into the SIEM platforms themselves would enable these platforms to better analyze the massive quantities of data that they are purposefully designed to handle. However, analyzing and correlating those data flows challenges a SIEM platform's ability to provide real-time analysis. Your “first alert” capability could and should be your API infrastructure instead, as it can serve as a sensor to cue the SIEM on potential security threats.

What if the API infrastructure could be the common clearinghouse of security APIs? With your API infrastructure “clearing” all your security service APIs, the API infrastructure becomes your first point of analysis for all big data security analytics.

Threat protection

While TLS should be used for message (API call) confidentiality, CSOs also should consider which API infrastructure capabilities help to provide message integrity and availability.

According to the Open Web Application Security Project (OWASP), injection attacks remain the number-one risk for web applications. CSOs should ensure message integrity by seeking protection against XML poisoning, which is the ability to manipulate a schema either by replacing or modifying it to compromise the programs that process documents using the schema. For example, corrupt or extremely complex XML documents can cause servers to allocate more memory than is available, tying up CPU and memory resources, crashing parsers, and generally dis-



abling message processing and creating application-level denial-of-service attacks. To protect against this, an API infrastructure should validate messages against an XML schema, evaluate message content for specific black-listed keywords or patterns, and detect corrupt or malformed messages before those messages are parsed.

Also important are policies that fend off injection attacks against SQL (Structured Query Language) and JSON, the open standard format used primarily to transmit data between a server and web application, as an alternative to XML. Apigee, for example, offers these built-in threat protection policies to protect your API calls.

Regarding message availability, CSOs should have the ability to set temporal quotas on the number of API calls to prevent an inadvertent or deliberate denial of service (DoS) attack. For example, a quota that is based on a calendar period (per day, per week, or per month) might be appropriate. Alternatively, it might be useful to set a quota based on a rolling window, in which a start time isn't specified; instead, the counter for a defined interval starts when the first message is received from the client. A flexible quota, which is based on when the first request message is received from an app, could also be appropriate.

Traffic control

A significant challenge to the maintenance of healthy APIs is traffic control and CSOs need a spike arrest capability to manage availability. APIs consumed by heterogeneous client apps are vulnerable to performance lags and downtime caused by the sudden influx of request messages, whether caused by malicious attacks, buggy apps, or seasonality.

APIs consumed by heterogeneous client apps are vulnerable to performance lags and downtime caused by the sudden influx of request messages, whether caused by malicious attacks, buggy apps, or seasonality. CSOs need spike arrest capabilities to manage availability.

In Apigee Edge, spike arrest policies throttle the number of requests forwarded from the point in the processing flow where a policy is attached as processing step. You can attach a spike arrest policy at the proxy endpoint to request that flow will be limited to inbound requests on the consumption side. You can also attach the spike arrest policy at the target endpoint on the request flow to limit requests forwarded to the back-end service on the exposure side. For example, if you define the spike arrest policy to enforce a limit of 100 messages per second, the API infrastructure enforces that policy by throttling any requests that exceed the one-message-per-10-milliseconds limit.

Any server that receives online data is subject to attack, whether malicious or unintentional. Apigee Edge enables you to enforce XML threat protection policies that screen against XML vulnerabilities and minimize attacks on your API using the following approaches: validate messages against an XML schema (.xsd); evaluate message content for specific blacklisted keywords or patterns; detect corrupt or malformed messages before those messages are parsed

Identity

Identity is another critical area of interest for CSOs when it comes to an API infrastructure. How is user or developer provisioning handled, and how scalable is it? What does the infrastructure's RBAC model look like for delegated provisioning and user or developer management? Almost certainly, CSOs won't want to be responsible for these tasks. Instead, they'll want to leverage the infrastructure's ability to create a flexible RBAC model that allows these tasks to be delegated appropriately and securely to other business units or groups.

What flexibility exists in being able to create those groups? For example, can your existing lightweight access directory protocol (LDAP) or active directory groups be leveraged so that you do not need to recreate groups solely for the API platform? Apigee Edge allows you to customize policies that integrate OAuth with your existing user store, be it active directory, an LDAP directory, or some other user store. Since every organization authenticates users in different ways, some policy customization or code is required to integrate OAuth with your user store. Having to recreate that hierarchy could be extremely cumbersome. Apigee provides a sample API proxy and login app to simplify this integration.

Your organization may also want the ability to act as an identity provider, especially with regard to the use of SAML. If using SAML, your API infrastructure can function as a service provider or an identity provider. When the infrastructure validates SAML tokens on inbound requests from apps, it acts as a service provider. When generating SAML tokens to be used while communicating with back-end services, the API infrastructure also takes on the identity provider role.

Infrastructure security and compliance

Whether an organization runs an API infrastructure on-premises or uses a cloud-based service, the infrastructure security that the infrastructure operates in should be a prime consideration. While CSOs have been dealing with these concerns for years, the point does not become moot with APIs. The physical security of the data center; power and environmental considerations (HVAC, for example); background checks on key personnel (such as organizational administrators); business continuity planning and disaster recovery (BCP/DR); and other considerations are still critical for ensuring the secure operation of your API infrastructure.

Regardless of where an organization's API infrastructure is hosted, compliance—the frameworks your API program must adhere to, and how they are validated—remains a prime consideration. At a minimum, an organization should ensure that service organization control (SOC) 2 type II controls are being addressed. Very likely, you have a PCI DSS (Payment Card Industry Data Security Standard) requirement, and possibly HIPAA (Health Insurance Portability and Accountability Act) requirements as well. If an organization is dealing with the federal government, it may need to comply with FedRAMP (Federal Risk and Authorization Management Program) rules.

Conclusion

As enterprises continue to transform themselves digitally to meet the demands of customers, employees, and partners, APIs facilitate this evolution, and are spreading far and wide. Consider Amazon.com, a leading example of a digital enterprise. Its Web Services Elastic Cloud Compute (EC2) alone employs hundreds of APIs. [Programmable Web](#) tracks over 10,500 APIs publicly available to developers. And that's the tip of the iceberg. Hundreds of thousands of internal and partner APIs round out the total API count. Twitter handles more than 13 billion API calls per day, and Expedia's affiliate network produces \$2 billion worth of business annually via APIs alone.

The [very public hacks against the popular Snapchat mobile app's API](#) demonstrate that CSOs need to be cognizant and knowledgeable about how to provide API and API infrastructure security. For example, the use of [rate limiting would have thwarted at least one of the hacks that hit Snapchat](#). In other words, the ability to manage API security is far easier when using a complete API infrastructure that provides these kinds of capabilities, such as [Apigee Edge](#).

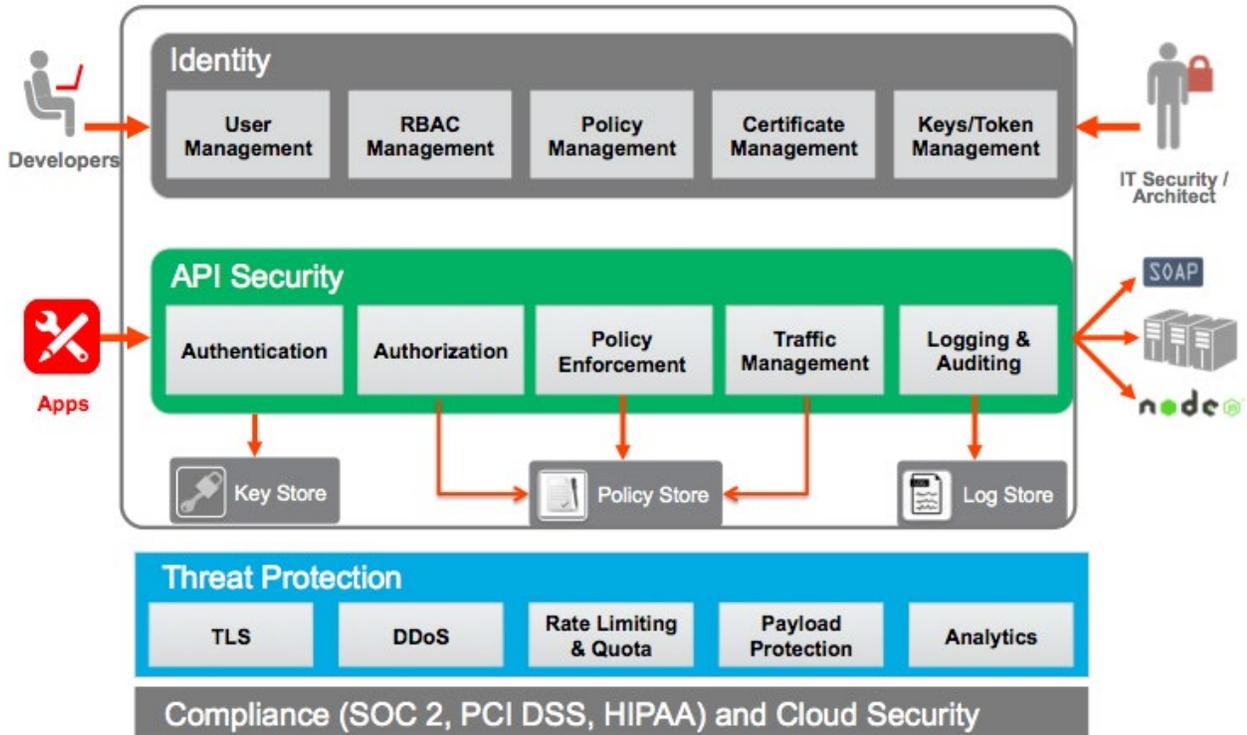
Any API-centric architecture must meet the security requirements from each of the stakeholders:

- **the developer** needs the ability to create and deploy apps and configure security (not code) via the API and self-service management portal
- **the API architect** needs the ability to securely expose the back-end services with necessary authentication, authorization, and threat protection
- **the security architect** needs the ability to protect high value assets (intellectual property and sensitive data) that are being exposed via APIs, while meeting enterprise security standards for authentication, authorization, and auditing (AAA)
- **the application/business owner** needs the ability to manage availability, risk, and compliance when delivering an API service to end users accessing from any device at anytime



Appendix A: Apigee Edge Security Architecture

Apigee Edge supports all the stakeholders in the value chain—and all the use cases from the consumption of APIs by client apps to the exposure of back-end services.





Appendix B: Checklist for Assessment

API security, and the security of the infrastructures these APIs are running on, is of paramount importance to a CSO at an enterprise that is exposing digital assets.

As with any other technologies that an organization's business units consider, CSOs must conduct due diligence, learn about the world of APIs, and consider end-to-end security needs. Consider how to support the API consumption and exposure sides of the equation and the separation of duties between the service providers (the IT architect, IT security, and business) and the service consumers (developers and end users).

	Capability . . .	Consider . . .
API Security		
	App to API (Consumption)	
✓	Authentication	Can your infrastructure support multiple authentication methods (e.g., TLS, OAuth, API keys)?
✓	TLS authentication	Can your infrastructure handle TLS authentication both one-way and two-way?
✓	Authorization	Permission management
✓	API key and token management	Can your infrastructure handle scale to handle massive number of API key and tokens?
✓	Runtime policy	Can your infrastructure handle runtime policies?
✓	SLA enforcement	Can your infrastructure help to enforce SLA provisions with your partners?
✓	Logging and audit	Does your infrastructure provide logging and audit capabilities?
	API to Backend (Exposure)	
✓	Authentication	Can your infrastructure support multiple authentication methods (e.g., TLS, OAuth, SAML)?
✓	TLS authentication	Can your infrastructure handle TLS authentication both one-way and two-way?
✓	Delegated authentication	Can your infrastructure support delegated authentication
✓	Identity integration	Can your infrastructure integrate with custom identity providers?
✓	Authorization	Can your infrastructure support fine-grained authorization?
✓	Logging and audit	Does your infrastructure provide logging and audit capabilities?



	Capability . . .	Consider . . .
Threat Protection		
✓	Poisoning	Does your infrastructure protect against XML poisoning?
✓	Injection	Does your infrastructure protect against JSON and SQL injection?
✓	DoS / DDoS	Does your infrastructure protect against app DoS attacks and DDoS attacks?
✓	Rate limiting	Does your infrastructure quota and spike arrest capabilities
✓	Address restrictions	Does your infrastructure provide IP address restriction capabilities?
Identity		
✓	User provisioning	Does your infrastructure provide scalable user provisioning?
✓	RBAC management	Does your infrastructure provide robust RBAC capabilities
✓	Groups	Does your infrastructure identity capabilities support groups?
✓	Identity provider	Can your infrastructure act as an identity provider (e.g., SAML assertions)?
Infrastructure Security & Compliance		
✓	Location flexibility	Can your infrastructure operate either or both in the cloud and on-premise?
✓	Cloud security	If your infrastructure operates in the cloud, does that include robust cloud-based security?
✓	Compliance	Does your provider comply with SOC 2, PCI-DSS, and HIPAA?
✓	Support	Does your provider have 24 x 7 support?

About Apigee

Apigee empowers enterprises to thrive in today's digital world by transforming digital assets into innovation engines. Results are delivered as apps, optimized by data and catalyzed by APIs. Hundreds of companies including Walgreens, eBay, Shell, Bechtel, Marks & Spencer & Vodafone partner with Apigee to accelerate their digital transformations. To learn more, see apigee.com.

[Accelerate Your Digital Strategy with Best Practices](#)

[Why Apigee?](#)

[Apigee Products](#)

About the author

Tim Mather brings 20+ years of experience as an information security executive to his chief security officer role at Apigee. He is a frequent speaker and commentator on information security issues, and serves as an advisor to several security-related start-ups.

Share this

