Securing the Edge API and the Microservices Mesh

akana
by Perforce

Abstract

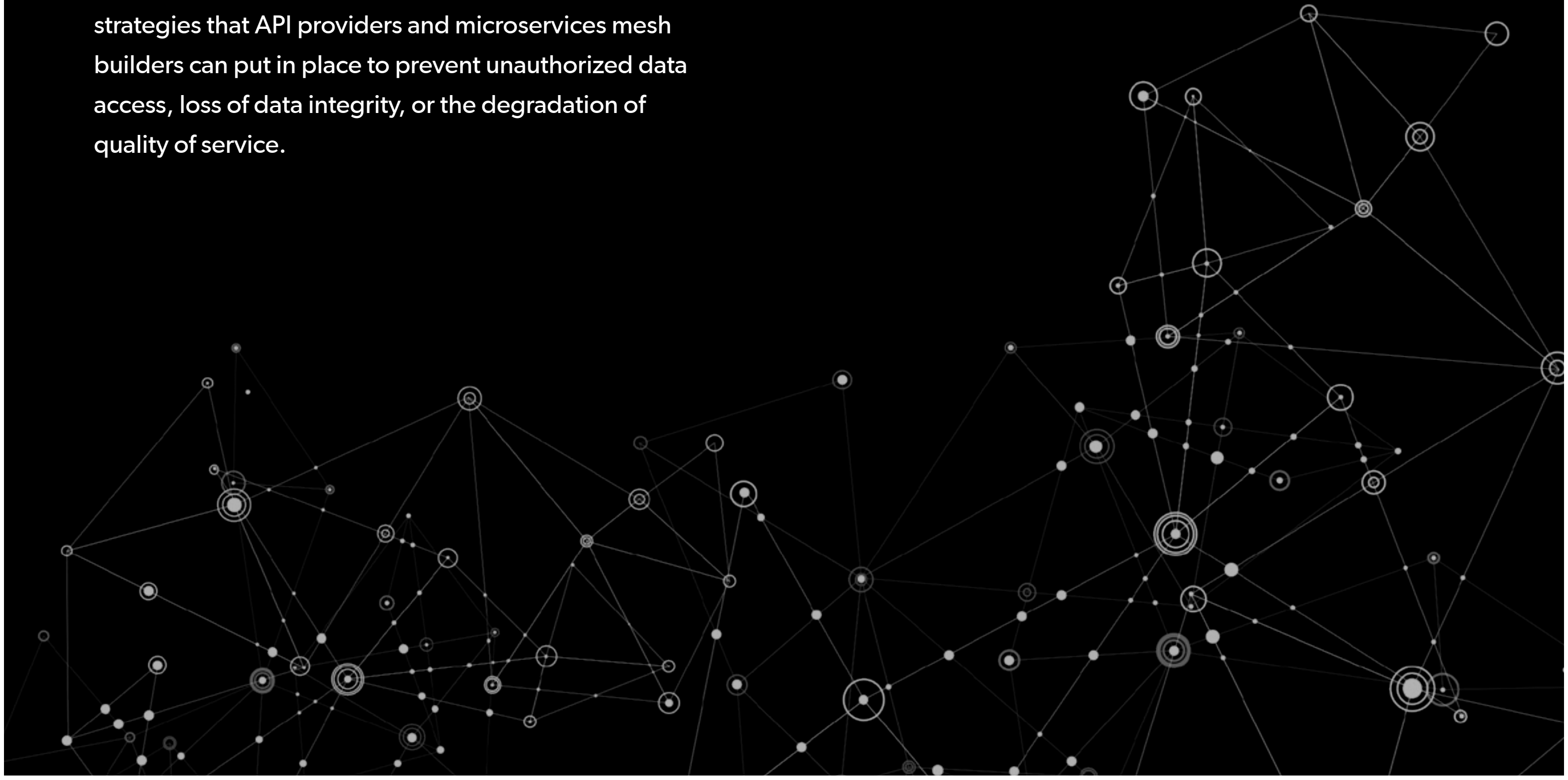
Malicious assaults and distributed denial-of-service (DDoS) attacks are increasingly targeting enterprise applications as back-end systems become more accessible and usable through cloud, mobile, and on-premise environments. APIs and microservices are major points of vulnerability, given their ability to offer programmatic access to data.

Security, therefore, is an essential element of any organization's API strategy. While API security shares a lot of aspects that are common to both web site security and network security, it is also fundamentally different both in terms of usage patterns as well as the unique areas of additional risks that microservices and sidecars are susceptible to.

For instance, APIs move the boundary of interaction from the web tier to the backend applications, microservices, and data sources directly.

The purpose of this paper is to help you understand the necessary components of a well-constructed API security strategy. First, it takes you through an API strategy assessment discussing the various attack vectors that could potentially make your API vulnerable. Then the paper talks about risk mitigation solution architecture strategies that API providers and microservices mesh builders can put in place to prevent unauthorized data access, loss of data integrity, or the degradation of quality of service.

Contents



Introduction

APIs have become a means for accessing data through digital channels such as mobile applications, cloud, and the Internet of Things (IoT), making it easy for enterprises to make their information available to wider audiences, whether they are customers, partners, or employees. However, as APIs become a part of standard enterprise architecture, their security risk profile is emerging as an issue that potentially diminishes the appeal of this powerful integration technology. Malicious assaults and DDoS attacks are increasingly targeting enterprise applications as back-end systems become more accessible and usable through cloud, mobile and on-premise environments. APIs can be a major point of vulnerability, given their ability to offer programmatic access to external developers.

The API can be a major point of vulnerability, given its ability to offer programmatic access to external developers.

API security challenges are a natural successor to earlier waves of security concerns on the Web. When businesses first connected to the Internet in the early 1990s, they encountered the precursor to modern day hackers, malicious users that probed computers for open ports and platform vulnerabilities. To prevent breaches, organizations deployed firewalls and intrusion prevention systems (IPSs). However, when these same organizations opened up access to their Web applications, hackers quickly circumvented the firewalls, and they used evasion techniques like encoding and comments to evade IPS signature detection.

Web Application Firewalls that came onto the scene a few years later could identify the type of application traffic, such as HTTP or instant messaging. But, unfortunately, this application awareness provided little protection against preventing API attacks. Next generation firewalls cannot block attacks that exploit API specific vulnerabilities. They cannot detect various kinds of JSON or XML attacks, or parameter tampering attacks. They cannot stop fraudulent devices or business logic attacks. Organizations that rely solely on network security solutions to protect their APIs shouldn't be surprised when they suffer an API application breach.

Security should be an essential element of any organization's API strategy. Getting API security right, however, can be a challenge. While API security shares much with web application and network security, it is also fundamentally different. Usage patterns are not the same and APIs face other, unique risk factors. The purpose of this paper is to help you understand the necessary components of a well-constructed API security strategy, educate you on how potential hackers can try to compromise your APIs, the apps, or your back-end infrastructure, and provide a framework for using the right tools to create an API architecture that allows for maximum access, but with the greatest amount of security.

API Strategy Assessment

APIs are not exactly a new concept. However, a host of security risk factors parallel the emergence of the RESTful API, which uses HTTP protocols and JSON as the new, ubiquitous connection interface for billions of connected devices and hundreds of thousands of mobile apps and cloud applications. Because APIs use web technologies over the open Internet, an API developer is going to encounter the security threats commonplace in this ecosystem.

Most of the traditional risks for web sites and web applications apply equally to APIs, but the unique nature of APIs further expands the surface area for attack. While web applications are designed to provide a specific user interface and expose a specific functionality from the back-end, APIs provide a much more flexible conduit into the back-end, allowing an API consumer much more granularity and flexibility in terms of what and how much information it can fetch from the back-end servers. Also, the level boundary of interactions moves from the web-tier to applications and data repositories that sit behind your firewall. In essence, depending on how an API has been coded, it could dangerously expose back-end data, back-end architecture, and back-end applications to hacks and provide easy, low lying clues to identify potential attack vectors.

Risks posed by APIs include loss of integrity, confidentiality, and availability of data.

APIs could easily allow bulk data transfers much more easily than is possible with web applications. An API could be called repeatedly to compromise data or generate a DDoS attacks. Risks posed by APIs include loss of integrity, confidentiality, and availability of data. In some cases, the business impact of a security incident resulting from an attack on an API might be extreme. The following sections highlight some of the more serious threats, vulnerabilities, risk exposures, and business impacts.

API Threats and Vulnerabilities

Hackers today exploit business logic flaws in your APIs and weaknesses in emerging technologies such as containers and microservices sidecar platforms. They perform repeated brute force attacks. They use wildcards in search fields to shut down APIs and applications. They will screen your APIs to find loopholes to extract valuable business information. These attacks have frustrated many organizations because traditional web security cannot catch these attacks. APIs face numerous threats, some of which are slightly new versions of very familiar types of attacks:

- **DDoS attacks** – APIs are potentially open to flooding and other types of attacks that can bring back-end systems to a halt. A DDoS attack cripples an API by overwhelming it with requests (e.g. a "Request Burst").
- **Cross-site scripting (XSS)** is a hacking technique that takes advantage of known vulnerabilities in a web-based application, the servers that support it, or related plug-ins. The XSS attack places malicious code into content that is delivered from the compromised site. The reason that XSS is so dangerous is that the tainted content arrives at the API from a trusted source. It has all the permissions granted to that system. By placing malicious code on web pages, the hacker can get high-level access-privileges to confidential content, cookies, and other browser-based information about the user. Several high-profile sites, such as Twitter, have suffered from XSS attacks, and many other prominent sites have been described as vulnerable to them.
- **SQL injections** attack database-driven web applications. In this type of attack, the hacker places malicious SQL statements into an entry field for execution. For example, the hacker could instruct the database to dump the content of the database to the attacker. SQL injections exploit security vulnerabilities such as incorrect filtering for string literal escape characters embedded in SQL statements.

- **Parameter attacks** in general are one of the most vexing threats in the API world. They threaten APIs by modifying the parameters of the API call. For example, HTTP Parameter Pollution (HPP) changes the HTTP parameters of a web application in order to perform a malicious task differently from the intended behavior of the application. This hacking technique is considered to be simple but quite effective. Furthermore, these attacks can be realized because the input is not sanitized properly. HPP injects encoded query string delimiters in existing or other HTTP parameters (i.e. GET/ POST/ Cookie), which make it feasible to supersede parameter values that already exist to inject a new parameter or exploit variables from direct access. This attack affects all web technologies, whether running client-side or server-side.
- **Malicious code injection** – Manipulates security design flaws in technologies to send valid code to services using SQL, LDAP, XPATH, or XQuery statements to open up the interface for any user to take control or to cause harm.
- **Business logic attacks (BLA) –** Because the APIs makes business-related operations available as procedure calls, a hacker can attack the business logic of a company via an API. As opposed to traditional technical application attacks (e.g. XSS or SQL injection), business logic attacks do not contain malformed requests and include legitimate input values making this sort of attack difficult to detect. Furthermore, BLAs abuse the functionality of the application, attacking the business directly. A BLA is further enhanced when combined with automation where botnets are used to challenge the business application.
- **Tampering with API requests and responses –** This is an attack that manipulates the API request and response parameters exchanged between client and services with the goal of modification of data. An example of this is the man-in-the-middle type of attack where eavesdropping can occur when non-secure API communications expose the data to access and tampering while in transit.
- **Identity and session threats** – APIs can be exploited to allow for session fixation, wherein a hacker uses a real user's session ID to gain access to the user's account. APIs are vulnerable to misuse and unauthorized use if they lack adequate Authentication/Authorization (Au/Az) controls. This is true for direct app-to-API calls, but the risks can be worse with multi-party authentication schemes, such as where a user of an application wants to grant permission to a website to access his or her private data from a third application.
- **Lack of rate limiting or quality of service (QoS)** – In its natural state, an API has no way to limit the number of calls it can receive. An API in this condition can be flooded and shut down or made so busy that it affects QoS for legitimate users.
- **Service information leakage** – An API inadvertently leaking data about its configuration, resulting in the ability to take control or expose private data. Broken Session IDs, Keys, and Authentication create exposure to unauthorized access through authentication factors that are not functioning because of poor security design or technology bugs.

Risk Exposure/Business Impact

What is the actual exposure faced by businesses with API security vulnerabilities? Specific exposure will vary from company to company, but in many cases, the business impact of an API-related security incident can be quite serious. The following table summarizes the major exposure categories and their potential impacts on business.

Risk Exposure	Potential Business Impact	Drivers of Business Impact Level
Loss of service	• Lost revenue • Negative impact on customers • Loss of trust from business partners • Contract breach • Employee stress	• Importance of service quality and QoS to business (i.e. online commerce would be a high level of importance)
Compromise of personal identifiable information (PII)	• Legal liability • Loss of reputation/ damage to brand • Compliance liability • Loss of trust from business partners	• Regulatory environment • Requirement to be PCI or other regulation compliant • Contractual obligations to protect PII
Improper access to data/theft/leak of private data	• Loss of intellectual property • Competitive disadvantage • Legal liability • Loss of reputation/ damage to brand • Compliance liability • Loss of trust from business partners	• Value of data exposed by APIs • Regulatory environment • Contractual obligations to protect data

Risk Mitigation Best Practices

It is possible to put together an effective, comprehensive API security program that mitigates the most serious risks to back-end systems and data. The architecture, tools, and controls vary, but the fundamental requirements are to defend against the major attack vectors that can exploit vulnerabilities in APIs. At a high level, it is necessary to control access to APIs, monitor API usage, and limit API usage both in terms of absolute number of API calls and the rate of API calls.

It is also essential that IT departments think about API security in the context of the complete API development lifecycle. This practice is recommended for application security in general, but with APIs in particular, the pace of changes to API code and the number of disconnected parties involved in their use make security management through the lifecycle absolutely imperative.

Validate User and App Identity

Controlling access to APIs is critical to mitigating the risks of identity and session threats. It is essential to separate the identity of the user and the app that is accessing the API. API providers should be able to identify an app uniquely and control the operations that the app itself can perform. An API key gives the API provider a way to verify the identity of each app or caller. The API provider can use this information to maintain a log and establish quotas by user. The API key validation is something that should be controlled by the API management tier. The end user's identity also needs to be verified to check if the end user has access to the resource he or she is requesting. This can be done either at the API management tier or delegated to a more authoritative source like Identity and Access Management systems that perform and facilitate single-sign-on. In some consumer-facing apps, the API provider might instead allow the user to use their social login provided by Google, Facebook, Twitter, and so forth.

The challenge is to make APIs part of that broader Identity and Access management apparatus. The API management tier needs to both provide basic built-in authentication and authorization capabilities as well as integrate with existing enterprise identity and access management systems. API providers should also take into account the context in which the API or the app is being used by considering authorization based on details such as user, application, geo-location, device type, and time. Applying flexible run-time policies and managing these policies from a centralized management console increases the flexibility and the control the API provider can have on these parameters.

How Akana Fits Into Your Existing Architecture

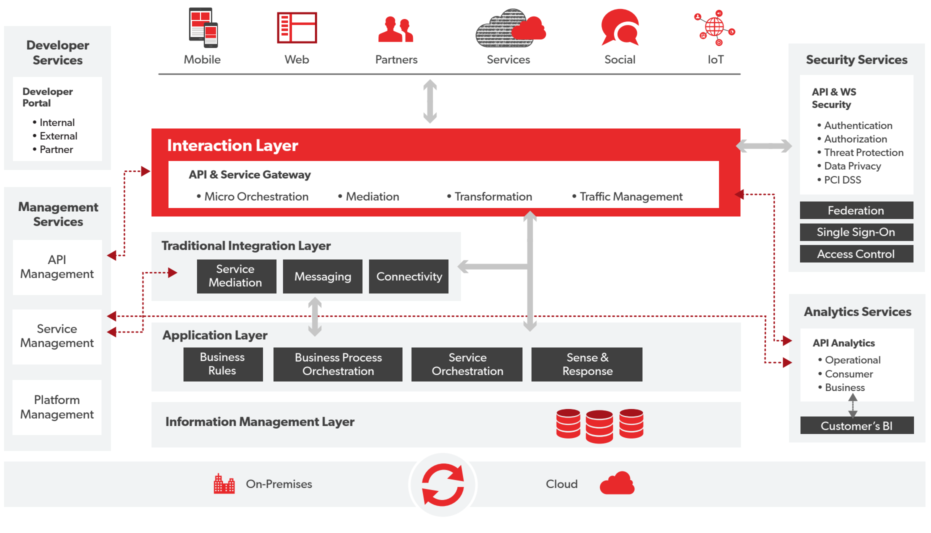


Figure 1: Secure edge APIs and the core microservices mesh

The Akana API Gateway offers multiple modes of authentication and authorization, including a built-in access management system. The API Gateway integrates with other enterprise identity and access management systems, such as LDAP, Ping Federate, CA Siteminder, Microsoft ADFS, and others. In addition, the API Gateway is capable of issuing and consuming X509 certificates.

The API Gateway lets security managers stay on top of API keys, which are needed to grant apps access to APIs. However, as noted above, it is not a wise practice to use API keys for end user access control. API keys are not tied to end user identity. They can be easily copied because they are stored on the app itself, outside the security perimeter of the API owner. API keys should be used for rate limiting, monitoring, and QoS. The Akana API Gateway supports HMAC based encryption of API keys so that the keys can be securely exchanged and authenticated between the app and the API Gateway.

It is not a wise practice to use API keys for end user access control.

OAuth is the recommended technology to manage the authorization in cases where an app user needs to authorize an API to access data held by a third party. OAuth is an open standard for authorization that enables an application to request access to third-party systems on behalf of its users. For instance, an app user might want to grant a bank system authorization to get personal account data from a stock market trading system controlled by another entity. OAuth can facilitate this authorization grant. OAuth makes the authorization possible without the need to share sensitive personal login information. Rather, it creates a secure token that allows one system to access specific information and functionality from another system. OAuth can be difficult to tackle in its native, open standard form; however, Akana OAuth Server eliminates the complexities of implementing OAuth and integrating with existing enterprise identity and access management systems.

Preventing Attacks

A well-constructed API security toolset offers defense in depth against an array of threats. The Akana API Gateway includes a content firewall that can detect malicious content, such as virus, SQL Injection, or malformed JSON or XML data structures. By detecting and blocking these problematic API calls, the Gateway mitigates the risk of parameter attacks, business logic attacks, SQL injection, and XSS attacks. The Gateway can also establish whitelists to reduce the risk of attack from untrusted sources

In the case of DDoS, attack prevention occurs on two levels. API providers should limit the number and rate of API calls by any specific app. They should monitor usage and send alerts to system administrators if the API is getting overloaded with requests or is subject to suspicious patterns of API calls from the network. The Akana API Gateway provides these capabilities but also adds a licensing capability that makes it possible for API owners to establish contractual relationships with apps, including pay for API use terms. Licensing, combined with rate limiting and QoS monitoring, greatly reduces the risk of DDoS attacks.

Encrypt the Message Channel

Encryption for API security must be pervasive and flexible. API security providers should enable SSL/TLS encryption for all APIs by default. Enabling SSL is an essential and basic step for all API providers and offers an extremely effective defense against man-in-the-middle attacks. A built-in PKI and key distribution model can ensure the privacy of customer data with sophisticated encryption and signature capabilities. It can also provide a mechanism for client-side authentication using certificates. In order for an API security toolset to mitigate the risk of the man-in-the-middle attack, it must provide message encryption.

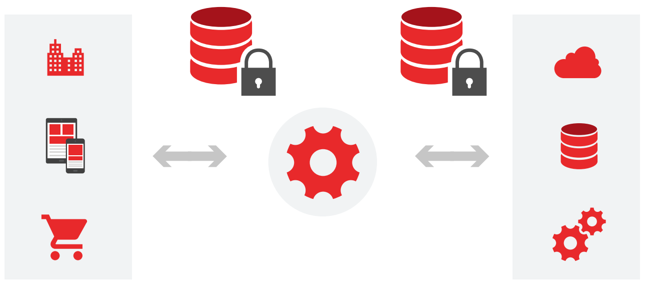


Figure 2: API message security

The Akana API Platform supports SSL/TLS as well as message-based encryption and decryption using the XML-Encryption and JOSE standards. The API Gateway can sign and verify messages and headers to provide non-repudiation. It uses built-in PKI services to simplify key and certificate distribution and management.

Monitor, Audit, Log, and Analyze Your API Traffic

Authentication and authorization enable an enterprise to resolve who can use what in the API. However, what about how many times your API is being used by a specific app or a user? More importantly, how is it being used? Excessive API use can be even more damaging and costly to the API provider than no use at all. It is not possible to manage an API's QoS without the ability to monitor the activities and usage of the API. Your API management solution should enable you to monitor the performance and availability of an API and its consumers. It is also a best practice to monitor aspects of the usage of the API, such as most popular consumers, most popular operation, or operation consumption per consumer. Analytics from running the API can be very useful when planning extensions and updates or in understanding the usage of your APIs to further harden it against attacks.

Furthermore, if your API is involved in financial transactions, you might be required to make the API metrics part of your audit process. Country or industry specific laws or other compliance policies could require you to adhere to specific security practices. For instance, you might need to provide auditors with verifiable logs of API requests and responses to enable the detection of unauthorized users.

Lastly, root cause analysis relies heavily on logging as a means of providing a window into the actions of the API at the time of errors or security breaches. Without some deep, but easily variable, logging capability, API issue resolution would become a horribly hit-and-miss activity.

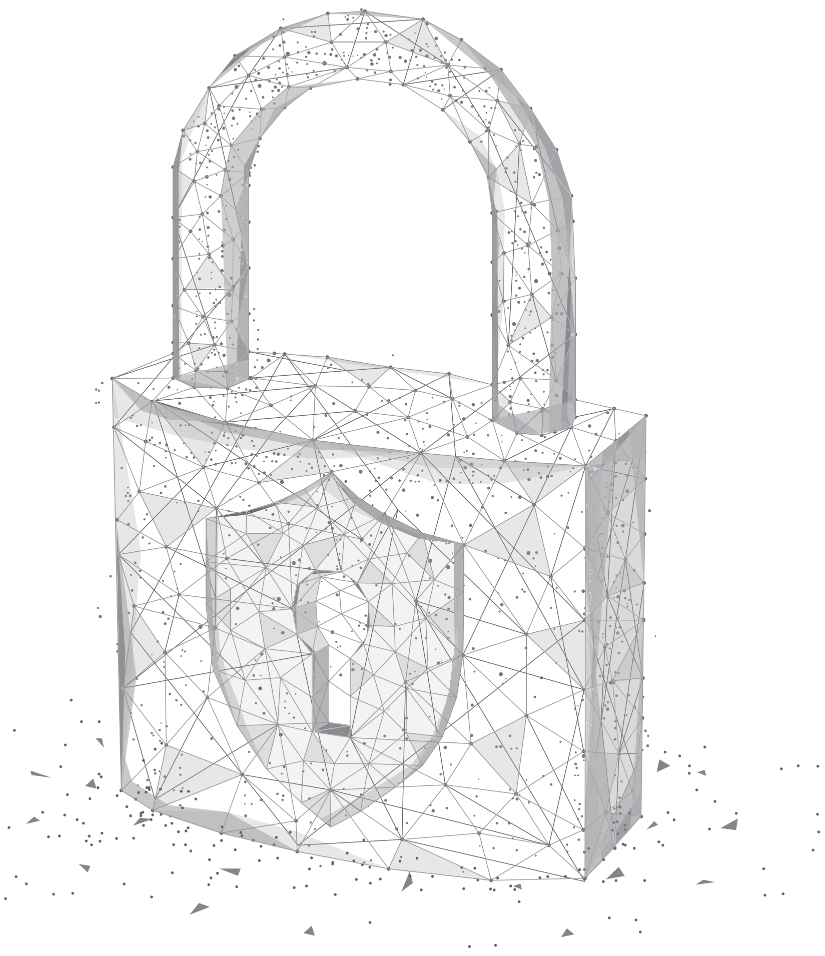
Monitoring your APIs and capturing detailed logs and audit records is essential to managing the overall security of your APIs. The API management platform should provide the ability to export log and audit records for offline analysis.

Akana's API Management platform provides extensive monitoring, auditing, QoS, and licensing capabilities, along with the ability to export these log and audit records for offline analysis. Leveraging real-time monitoring, customizable alerts, and offline analysis, API providers can ensure that their microservices mesh of sidecars only accepts mutual TLS connections from trusted API Gateway endpoints with valid certificates.

Excessive API use can be even more damaging and costly to the API provider than no use at all.

Building API Security Into Software Development and Deployment Processes

One thing that should be clear to infosec professionals is that API security will be inadequate without a comprehensive, policy-based approach. Using a scattered toolset and ad-hoc security rules will almost certainly lead to gaps in security and exposure to unnecessary risks. APIs that your organization develops and the apps that connect to them should be governed by a coherent set of security policies that span the complete API lifecycle. At the planning stage, architects and developers should think through the dependencies, authentication issues, data integrity challenges, and so forth that will affect the API once it is developed and deployed. In development, the policies established for the API should be implemented. For example, if an API requires an OAuth token for third-party authentication, the capability to provision that token should be built into the API code. Before deployment, APIs should be subject to penetration testing. At runtime, the API should be monitored for threats and performance issues that might indicate a looming security incident. Rates and QoS policies should be established to mitigate against flooding and DDoS attacks.



API security will be inadequate without a comprehensive, policy-based approach.

	Microservices Mesh of Istio Sidecars For more information: https://istio.io/docs	**Akana API Gateway Platform** For more information: http://docs.akana.com
Deny Checker	1.0.1 Stable	**v2019.0 Mature** Allow and deny rules configurable with a policy administration declarative UI
List Checker	1.0.1 Stable	**v2019.0 Mature** Allow and deny rules configurable with a policy administration declarative UI
Pluggable Key/Cert Support for Istio CA	1.0.1 Stable	**v2019.0 Mature** Integrated Java PKI and HSM keystores
Service-to-service mutual TLS	1.0.1 Stable	**v2019.0 Mature** With the ability to enforce mutual TLS 1.2
Kubernetes: Service Credential Distribution	1.0.1 Stable	**v2019.0 Mature** Policy decision point (DB) and policy enforcement point (Gateway) contract management architecture

Security managers have many options they can employ to devise and implement API security policies.

The Akana API Gateway is one such tool, with comprehensive out-of-the-box policies. The Gateway allows security managers to choose from different authentication schemes, standards and token types to ensure that only valid users and applications get access to your APIs. The following recommended API security counter measures are contained as features in the Gateway, though they can also be realized through alternative means. The advantage of the Gateway, and tools like it, is that it can make a complete API security regime possible within a single platform.

Use a PCI Compliant Infrastructure

API security must be part of the PCI compliant controls that ensure that Personal Identifiable Information (PII) is protected. APIs in PCI complaint businesses have to be hosted on infrastructure that meets the strict criteria of the PCI regulations. Akana's API Management solution and cloud offering have been validated for compliance with version 3.0 of the Payment Card Industry Data Security Standard (PCI DSS). Akana recently underwent a series of rigorous audits by an independent Quality Security Assessor (QSA) to ensure that it met best practices and security controls needed to keep sensitive data secure during transit, processing and storage. Akana is one of the few 'Approved Service Providers' for major credit card brands.

	Microservices Mesh of Istio Sidecars For more information: https://istio.io/docs	**Akana API Gateway Platform** For more information: http://docs.akana.com
VM: Service Credential Distribution	1.0.1 Beta	**v2019.0 Mature** Policy decision point (DB) and policy enforcement point (Gateway) contract management architecture
Mutual TLS Migration	1.0.1 Beta	**v2019.0 Mature** Client certificate management is self-service for apps consuming mutual TLS APIs
Traffic Control: Label/ content based routing, traffic shifting	1.0.1 Beta	**v2019.0 Mature** Visual Process Designer to create custom traffic flows using the branch, split and join process activities
Resilience features: timeouts, retries, connection pools, outlier detection	1.0.1 Beta	**v2019.0 Mature** Resilience features with many QoS policy templates and integrated health status monitoring monitoring of a container's Outgoing HTTP connection pool statistics, Incoming HTTP thread pools, database connection pools, container memory usage, usage monitoring queues, JMS connections, container configuration state, container lifecycle.
Gateway: Ingress, egress for all protocols	1.0.1 Beta	**v2019.0 Mature** Protect the microservices mesh layer of the network with a tier of edge DMZ API Gateways, rather than connecting the mesh controller directly to a cloud load balancer.
TLS termination and SNI support in Gateways	1.0.1 Beta	**v2019.0 Mature** The API platform's support of SNI means that multiple keys/certificates can be used for one HTTPS endpoint. You can have individual identity keys/certificates per API implementation. Each implementation can use its own key/certificate for its own clients.

	Microservices Mesh of Istio Sidecars For more information: https://istio.io/docs	**Akana API Gateway Platform** For more information: http://docs.akana.com
Authentication policy	1.0.1 Alpha Operators specify Istio authorization policies using .yaml files. Once de-ployed, Istio saves the policies in the Istio Config Store.	**v2019.0 Mature** Easily link different identity providers and policies to different APIs and Application Contracts with an integrated Akana OAuth server, Bearer, MAC, JWT, JOSE token support and easy integration with user directories and OpenID Connect providers.
End User (JWT) Authentication	1.0.1 Alpha Istio only supports JWT origin authentication.	**v2019.0 Mature** A key advantage of the Bearer token is that the Resource Server can validate the token, without having to go to the Authorization Server. This is more efficient in terms of performance, especially when the Resource Server and OAuth Provider are different vendors. Signed and Encrypted JWT Tokens are also supported by the Akana API Gateway.
OPA Checker	1.0.1 Alpha All policies in OPA are written in Rego policy language (V1).	**v2019.0 Mature** Create authentication domains and policies declaratively with a web browser.
Authorization (RBAC)	1.0.1 Alpha The RbacConfig object is a mesh-wide singleton with a fixed name value of default. You can only use one RbacConfig instance in the mesh. Like other Istio configuration objects, RbacConfig is defined as a Kubernetes CustomResourceDefini-tion (CRD) object.	**v2019.0 Mature** The Akana OAuth Authorization service includes such activities as initiating a resource owner grant, authen-ticating the resource owner with the corresponding resource owner domain, and obtaining the resource owner's authorization for the application's access to the resources, with the specific scopes request-ed. Calls to this service are always initiated by the resource owner, never by the application. Since the authorization endpoint is only used in three-legged scenarios, these operations are only used by three-legged grant types. (Authorization Code and Implicit). Additionally, Licenses and Scopes provide authoriza-tion functionality to apps and APIs with or without the use of an OAuth token.
Enabling custom filters in Envoy	1.0.1 Alpha	**v2019.0 Mature** Filtering of protocol headers, path, query parameters, and XML or JSON message parts with XPath, JSON-Path, RegEx using policies or custom processes.

akana
by Perforce

Conclusion

Despite posing a potentially serious impact on your business, API threats can be mitigated. Protecting APIs and microservices from threats is a manageable prospect for organizations that have a commitment to information security. In most cases, API security will be an extension of existing security policies and controls, with some new elements added in. APIs do present some unusual security challenges due to the programmatic nature of the access they provide to outside users. With the right API security toolset, even these risks can be mitigated. When organizational attention is focused on API security, security managers can realize effective API risk mitigation by controlling access to APIs, limiting API usage, monitoring and controlling quality of service, and encrypting API calls and responses. It is a process that should encompass the entire API and microservices lifecycle, from planning through development and runtime.

LEARN MORE AT AKANA.COM