Whitepaper

noname

API Security 101

Table of Contents

OVERVIEW

APIs are everywhere in the modern IT world, and the volume of API traffic is only expected to grow. However, despite the power and popularity of APIs, many organizations struggle with API security. This can be complex because a single application may be composed of multiple backend APIs and each API may be called in many different ways.

When it comes to securing APIs, traditional web application security controls alone don't provide a complete solution. Instead, organizations must complement additional API-specific controls that address API security posture, add threat blocking and remediation mechanisms, and provide continuous API-tailored security testing.

APIs are a powerful and widely adopted technology integration tool.

APIs enable programs to talk to another in an efficient, developer-friendly way, even if they weren't written in the same programming language. In most modern applications, APIs communicate using REST, webhooks, gRPC, or GraphQL.

Historically, organizations had to decide whether to build or buy new technologies. Now, APIs offer a third option: **integrating with a partner's technology applications**. As a result, APIs are everywhere in the modern IT world. They are used in cloud migrations, micro services, partner integrations, Kubernetes, DevSecOps, DevOps, and automation.

"APIs have become the core for modern business. They provide companies with technologies that would not be attainable if those firms had to build the functionality themselves. An ecosystem of APIs is a win-win for all partners involved."

Mark Campbell,
Senior Director of Product Marketing
Noname Security

Examples of APIs in action include:

- **Salesforce's AppExchange**. This is a huge platform of API-driven apps. Salesforce generates around half of its revenue through APIs. Over the next four years, Salesforce estimates it will generate more than $1 trillion in revenue from AppExchange.
- **Google Maps.** Rather than creating their own map applications, companies can use APIs to integrate Google Maps into their systems for a fraction of the cost.
- **Travel.** The web portals for booking flights, hotels, or cars leverage APIs to support that unified customer experience.
- **Shipping and Logistics.** Retail companies use APIs to automatically provide details on shipping costs and delivery time frames.

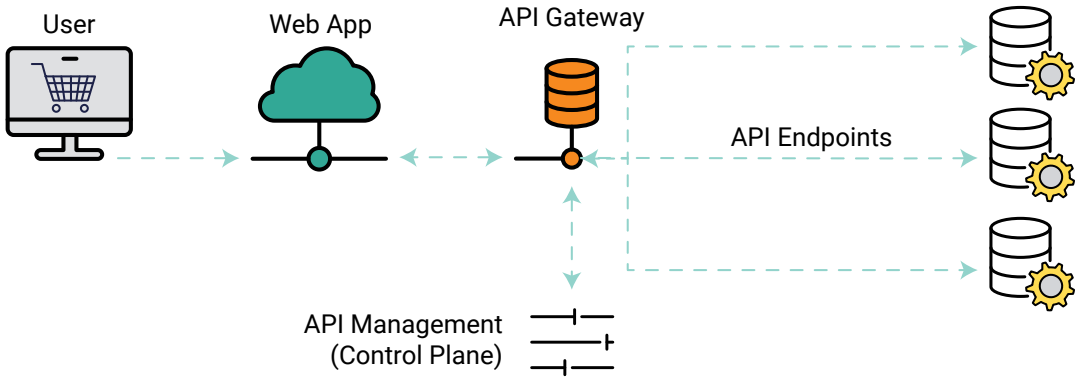
API Management and Gateways.

The two basic elements in the API delivery technology stack are API Management (control plane) and the API Gateway (data plane):

1. **API Management.** This includes versioning, publishing, and sharing a schema that describes the methods and data available from the API. Other aspects of API management include monitoring and dashboards to determine whether the APIs are performing correctly. If the API is open to the public or an organization is servicing clients with the API, onboarding and user documentation are also required.
2. **API Gateways.** Most deployments have an API gateway that offers a control point to handle authentication, authorization, and traffic management.

FIGURE 1

API management and gateways



API management and gateways provide critical capabilities for delivering APIs, but there are still significant security gaps. These gaps can be challenging to address. Making the problem more difficult is that often there is not a single, consistent stakeholder responsible for the API, including its securing. Depending on the organization, there can be several stakeholders involved. These can include API/Product managers, development teams, network teams, infrastructure teams, application security teams, and cloud teams.

API security challenges.

Some security controls needed for APIs are similar to those for traditional application security. However there are several security challenges that are specific to APIs and cannot be adequately addressed with traditional web application security controls.

API Security Challenges	
Creating an API or application inventory	Few businesses have a good handle on their API or application inventory. This information helps application security teams identify high-priority security risks. In an environment of limited resources, teams shouldn't spend time securing the least risky apps or APIs Without an accurate API inventory with context-based threat details , it is impossible to know where to effectively apply security efforts.
Imbalanced ratio of security staff to APIs or applications	Security efforts can become overwhelmed when each AppSec team member is responsible for an unmanageable number of APIs or applications. API sprawl exacerbates this issue.
Deploying the right tools to evaluate security across applications	In general, application security teams lack good preventive and detective controls to identify problems. Often, AppSec teams discover problems only when an app goes down or the CPU usage spikes to 100%. A proactive approach is preferable and can prevent costly outages
Handling diverse API communication patterns	APIs are software-to-software communication, which can be easier to manage than an application with a complete user interface. However, many different API communication patterns exist. An API may be called by a web app, a command line client, cURL, or another API.
Deploying robust tooling	API test tooling is only starting to mature. Although the Swagger and OpenAPI specifications have made tooling better, it's still fairly weak. It is early days, particularly for dynamic API tooling for testing.

Traditional application security controls offer only partial protection for APIs.

API gateways provide some visibility into security issues, since they serve as a central point for traffic and policy enforcement. However, not all API calls go through the gateway. Organizations are often blind to these API calls, as well as to microservice API calls that are not routed through the API gateway.

Logging and monitoring usually aren't very effective. Organizations typically discover authentication threats in APIs only after a breach occurs. In addition, API testing is different from application testing. Pointing a suite of normal application security tools at APIs doesn't provide proper coverage.

Legacy or zombie APIs present another concern. These could predate an organization's API security initiatives. They could also be APIs that were supposed to be decommissioned but were left active as an oversight. These APIs typically lack both ownership and can function without any visibility or security controls, just waiting to be exploited.

API security requires common controls, in conjunction with dynamic and static testing.

To protect APIs, teams must implement several common controls. These include encryption of network traffic, authentication to identify who is calling the API, authorization to determine whether the caller's request should be permitted, rate limiting as a blunt filter for abuse, as well as audit logging to capture a picture of what normal operations look like.

Dynamic and static testing are also essential, since the earlier that security vulnerabilities and misconfigurations are caught in the development cycle, the less expensive they are to fix.

Static application security testing (SAST) is required for API source code. Software composition analysis (SCA) is also recommended to understand the quality and test coverage of the source code. One caveat is that SAST doesn't have a great track record for finding API specific flaws, such as authorization issues.

"Unless you have a ridiculously big staff, you won't have enough people to do manual continuous testing. This is where automation helps, especially if you can tie it into CI/CD early in the development pipeline."

Matt Tesauro,
Director Security Evangelist
Noname Security

On the dynamic side, teams may use manual and/or automated testing. Manual testing can be extremely thorough, but this does not scale well or fit in with most agile development practices. If teams are using REST APIs with an OpenAPI Specifications, the automated test tooling is getting better. In general, however, automated testing designed for web applications often does not provide adequate security testing for APIs and their unique business logic functions.

Organizations must address security across the API ecosystem, from code to production.

Securing APIs is complex and requires an approach that encompasses all aspects of the API from its development, deployment configuration, and run-time operations. Three recommended strategies include:

1. **API security posture.** Assess every API, including legacy and shadow APIs, with data classification. Determine which ones are critical to the business. Based on that inventory, identify misconfigurations and vulnerabilities in the source code, network configuration, and policy. Focus security interventions on the highest-risk areas.
2. **Detection and response.** Deploy behavioral-based models for runtime API threat detection. Implement automated and semi-automated for blocking and remediation of threats.
3. **Continuous testing.** Continuously test API endpoints to identify API risks before they emerge. Supplement DevOps DAST, SAST, SCA and other existing tools with API-specific testing that can be automated and incorporated into CI/CD pipelines.

Noname Security is the only API security platform that solves API security posture management, provides API run-time security, and adds security into the API development pipeline.

About Noname Security

Noname Security is the only company taking a complete, proactive approach to API Security. Noname works with 20% of the Fortune 500 and covers the entire API security scope across three pillars — Posture Management, Runtime Security, and Secure API SDLC. Noname Security is privately held, remote first with headquarters in Palo Alto, California, and an office in Tel Aviv and Amsterdam.