Forrester Report Prepared For Hannah Hayes With F5 Networks, Inc.

API Insecurity: The Lurking Threat In Your Software

Design And Build API Security Top To Bottom, End To End Across The Software Lifecycle

August 6, 2021

By Sandy Carielli, David Mooter, Randy Heffner with Amy DeMartine, Chris Gardner, Melissa Bongarzone, Peggy Dostie

FORRESTER®

Summary

APIs provide a foundation for innovation and digital transformation at multiple levels — but at each, they open security holes and create privacy risks. API breaches are common, and securing your APIs requires holistic architecture across the software development lifecycle and especially at the define, build, and launch stages of the “secure what you sell” model. This report gives security pros a broad view of API security strategies, tools, and considerations as a working model for collaboration with digital channel executives, application developers, and enterprise, data, and infrastructure architects.

APIs Are The New Storefront, But Security Hasn't Kept Pace

APIs provide flexible network-based connections between applications, but they do much more than that. They transform business possibilities for customer engagement and ecosystem partnering, as well as create new direct revenue streams of their own. An API management solution, with its API user portal (aka developer portal), provides a storefront for establishing business relationships both inside an organization and across enterprise boundaries, but API security needs much more. This story has been building for a long while — external integration via SOAP APIs dates back to the early 2000s. However, though the range of usage scenarios and business relationships is greater today, so is attacker aggressiveness, and API security models haven't kept pace with the demands of a no-perimeter world. Security pros must consider that:

- **APIs are subject to the same vulnerabilities as traditional web apps.** Early external SOAP APIs were, more often than not, accessed over VPNs or two-way SSL connections between data centers. The comfortable, perceived security of a private network kept the API calls away from the prying eyes of external attackers. Today's REST APIs, on the other hand, when accessed through mobile apps and browsers, are open (by necessity) at the network and app levels to a raft of widely available client-side inspection and hacking tools. Teams that neglect long-understood security practices like server-side data validation and least-privilege data access controls will quickly learn that attackers can compromise their applications via API endpoints as easily as they could through direct HTTP calls.

- **APIs are also subject to vulnerabilities that traditional web apps rarely face.** Web apps have long had a form of back-end API access via asynchronous JavaScript and XML (AJAX) requests. These were often for less-sensitive UX-support and usability features, with transactional updates buried in the app's back-end code. Today, REST APIs may, without requiring a web app, provide direct access to powerful transaction updates and mass data access. Web apps — especially single page apps — may use APIs, but they may also have similarly powerful AJAX endpoints that, being buried within the app, are not recognized and managed with the same rigor as APIs. This proliferation of data endpoints introduces a new application risk, which the Open Web Application Security Project (OWASP) notes in its 2019 OWASP API Top 10. Firms that fail to track all types of endpoints risk unauthorized access and information leakage. Many APIs handle private data via opaque data identifiers without verifying the data owner, and this type of insufficient

data level authorization easily allows unauthorized access.

- **APIs may get blamed when really it's the app, infrastructure, or user at fault.** If you have an electronic lock on your home's front door and you give the code to a friend who promptly writes it down along with your address and forgets the paper on their car's dashboard, it is your home that's robbed. In the same way, apps with access to your API, whether the apps are built by you, a partner, or an open web developer, might open wide the front door to your data, leaving you to wonder whether you have any legal recourse.

- **API breaches are prevalent.** Breaches tied to poorly secured APIs, unprotected API endpoints, or API misuse are too common (see Figure 1). Data leakage is a frequent result, as attackers exploit these flaws to gain access to user account information, transaction details, or personal health status. API flaws have also helped attackers take control of and modify settings for personal IoT devices.

Figure 1

Recent Breaches Due To Improperly Secured APIs

Organization	Description	No. of users affected
Panera Bread	In the spring of 2018, Panera revealed that 37 million customers had had their data exposed. Some reports attributed the breach to an unauthenticated API endpoint.[1]	37 million customers
Venmo	In July 2018, Venmo revealed that user transactions were set to public by default and available to anyone through Venmo's public API.[2]	Number of customers unknown; data from over 200 million transactions was available
T-Mobile	In 2018, attackers exploited a "leaky" API and exposed 2.3 million customers' personal data.[3]	2.3 million customers
United States Postal Service	In late 2018, the USPS divulged that an API tied to its tracking service had broken access controls and allowed logged-in users to access account information for other users.[4]	60 million customers
Capital One	In mid-2019, Capital One announced a breach that had given an attacker access to personal information of those who had applied for various credit products. Through a server-side request forgery, an attacker compromised an application and gained access to Capital One's AWS-based infrastructure configuration API. The compromised role had broad-based entitlements so an attacker could access a wide range of AWS-based resources.[5]	106 million customers (100 million in US; 6 million in Canada)

1. Source: Torsten George, "The Next Big Cyber-Attack Vector: APIs," SecurityWeek, June 28, 2018
2. Source: Catalin Cimpanu, "PayPal's Venmo App Exposes Most Transactions via Its API," Bleeping Computer, July 18, 2018
3. Source: Tom Spring, "T-Mobile Alerts 2.3 Million Customers of Data Breach Tied to Leaky API," Threatpost, August 24, 2018
4. Source: "USPS Site Exposed Data on 60 Million Users," Krebs on Security, November 21, 2018
5. Source: Riyaz Wlikar, "An SSRF, privileged AWS keys and the Capital One breach," Appsecco, August 5, 2019

Source: Forrester Research, Inc. Unauthorized reproduction, citation, or distribution prohibited.

Organization	Description	No. of users affected
Xiaomi	In October 2019, a researcher discovered that unauthenticated API endpoints in Xiaomi smart pet food dispensers allowed anyone to change a pet's feeding schedule.[6]	Approximately 10,000 customers with active devices
Aarogya Setu	India's COVID-19 contact tracing application had authorization weaknesses and validated parameters at client side rather than server side. In May 2020, a researcher reported that they could make direct API calls and manipulate parameters to get the COVID-19 status of any neighborhood or location in India.[7]	All residents of India (approximately 1.3 billion)
Starbucks	In June 2020, a researcher was able to traverse API calls to find internal endpoints and discovered 100 million customer records.[8]	100 million customers

6. Source: Pierluigi Paganini, "Thousands of Xiaomi FURRYTAIL pet feeders exposed to hack," Security Affairs, October 30, 2019

7. Source: "Issue 83: India's COVID-19 tracing app, OAuth2 API attacks," APISecurity.io, May 14, 2020

8. Source: Sam Curry, "Hacking Starbucks and Accessing Nearly 100 Million Customer Records," Sam Curry, June 20, 2020

Security Pros Don't Know What APIs Exist Or Who Should Be Using Them

Think of an API-based application as a school of fish: There are many individual members, and the group together looks like a unified coalition, but a number of fish may be hidden or flit in and out of the group. Without some clear definitions and measurements, it's difficult to know exactly how many and which fish are in the school — or how many API endpoints are buried in an application (see Figure 2). Security pros struggle to:

- **Identify and catalog APIs and endpoints.** Vendors in the API security space often cite rogue endpoint discovery as their customers' most urgent API challenge. Many have only a rough idea of how many APIs they have but no accurate inventory — especially since APIs can be buried inside a mobile app or web app, and they can masquerade as quasi-API endpoints like AJAX requests, webhooks, and more. A large network and web application security vendor noted that customer APIs get deployed without a clear understanding of what security endpoints are available and that sometimes endpoints meant for internal testing have been accidentally

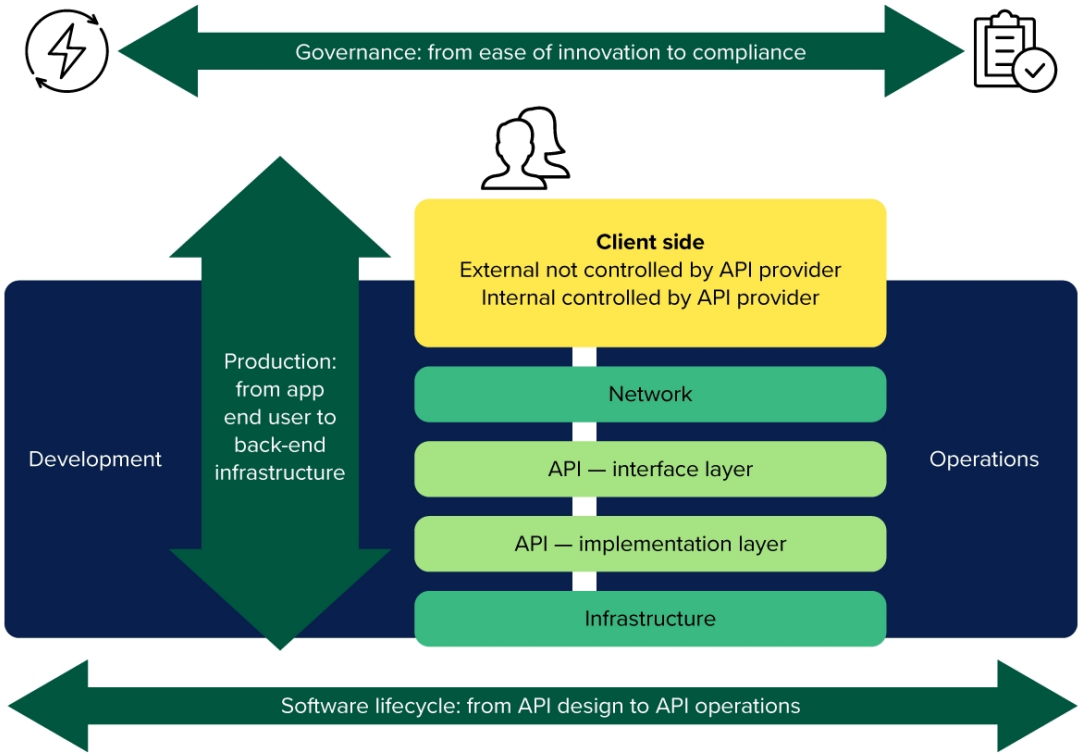
deployed to production.

- **Assure and manage API user identities.** Offering public APIs means that API calls come from a wide array of customers, partners, and applications. Yet many API security models authenticate and authorize only the initial API user's identity and let the API's implementation run under a trusted shared identity with broad data entitlements, which opens the API to breaches. Firms struggle to properly assess data entitlements, prioritize calls from key stakeholders, propagate identity information throughout the application, or flag impersonation attempts without properly identifying and designing for various identities.

- **Meet regulatory and compliance requirements.** Open banking regulations require banks to make their customer account information and services available to third-party providers: In the EU, the second payment services directive (PSD2) went into effect in 2018, and Australia has required all authorized deposit-taking institutions (ADIs) to comply with open banking by 2021. APIs are the most common approach to meeting open banking regulations, meaning that security pros do not have the option to restrict API availability. Therefore, security pros must ensure that their firms do not violate customers' security and privacy while addressing open banking requirements.

Figure 2

Holistic API Security Strategy Must Balance Ease Of Innovation And Compliance



Source: Forrester Research, Inc. Unauthorized reproduction, citation, or distribution prohibited.

Govern The Tension Between Innovative Design And Compliance

API design too easily centers on innovation and business benefits, overrunning critical considerations for security, privacy, and compliance such as default settings that make all transactions accessible. A proper framework of design guidance and governance ensures innovative functional capabilities have a built-in full range of risk and compliance measures. Key concerns are that:

•

API product management must incorporate security and compliance. API product managers may be external facing, leading the charge on new API-enabled revenue streams, market channels, and business models; or internal facing, enabling innovation via adaptive API-infused business processes and platforms. Collaborate with these product managers during the define stage of the secure-what-you-sell model to understand the answers to key questions: Whose data does the API work with and what are their rights to it? What regulatory or business requirements demand reporting on all access to the data? How might different business models for API access affect risk and compliance? How will the API's security model ensure a match between data accessible via an API call and the entitlements of the person using an app that uses the API?

- **API policies must ensure the right API-level trust enablement and attack protection.** Clearly, financial APIs need strong trust, but APIs for public data might reasonably have wide open trust models — and there is a broad continuum of requirements in between. Which are the appropriate policies to apply? In some cases, there is specific industry guidance, such as the OpenID Foundation's Financial-grade API (FAPI) for open banking. In other cases, the answer may be much simpler. Either way, the design must include attack protection along with an appropriate trust model. Each of these may have elements that operate in front of the API, at the interface-level entry point to the API, or within the implementation of the API (which may encompass calls to other APIs). Some policies may be declaratively applied; others may require custom coding — for example, to enable business logic within an API implementation to make fine-grained determinations of access entitlement.

- **API design governance sets security context for each API type.** Context, usage, and purpose vary across APIs, creating different security demands. One of the big mistakes Forrester commonly finds when reviewing API strategies, products, and services is insufficient attention to API taxonomy. Good classification of API types is critical for understanding risks and value-add for different API types and then setting common security patterns for each type. Categorizing and tagging APIs early in the identification and design process helps ensure you involve the right teams and apply appropriate policies as design and implementation proceed.

- **API platform governance supports coordinated security models.** If every API creator custom-builds identity propagation, entitlements management, risk assessment checking, and the rest into every API, it's a recipe for mistakes, misconfigurations, and security holes. It is better to focus attention and governance on the API platform as a whole — that is, everything from the network entry point on

down — to ensure the proper integration and configuration of the various tools, infrastructure elements, and policy specifications. Handle as much as possible on behalf of API creators, and foster collaboration between multiple roles (e.g., security, application architects, data architects, and others) to create clear guidance, implementation support, and patterns for what developers must do on their own.

Manage API Risks And Business Goals Throughout The Development Lifecycle

Testing an API for security is a core activity in the build stage of the secure-what-you-sell model, but it is not a once-and-done activity — it must be done continuously in launch as well. Remember that:

- **API security testing — before, during, and after deployment — is the safety net.** Prerelease testing tools like static application security testing (SAST) and dynamic application security testing (DAST) have long been staples of mature application security programs, and firms continue to implement security testing earlier in the software development lifecycle (SDLC). Take the same approach to APIs, and build testing into all stages of development, looking for weaknesses in API definitions and in running APIs. As with traditional application security testing, embed API security testing into the CI/CD pipeline where possible, and use stage gates to enforce good security practices.

- **Security-oriented functional testing helps identify data flow and trust level issues.** Security-specific testing tools add great value, but security-oriented functional testing is also important. Use this approach to ensure, for example, that a request from user A for user B's data will fail or that that failure messages look the same no matter which element of back-end infrastructure catches an invalid request.

- **Hardened combined configuration of APIs, apps, and infrastructure is the foundation.** OpenAPI specification (OAS) files define an API's interface, including the functions it provides, parameters to refine requests, payload schemas, and security requirements. However, without proper controls and maintenance, OpenAPI specs may be incomplete or outdated. Beyond OpenAPI, there are any number of ways that the actual business logic and data of an API may be implemented, with integration platforms, SaaS apps, mainframe apps, cloud

platforms, Java, .NET, JavaScript, Python, Go, Kubernetes, low-code, AMQP, and Kafka being just a few of the technologies that may be behind an API's interface. Work with the dev team to set policy around API specification updates and implement stage gates in the SDLC to ensure that policy is followed. Also, collaborate to ensure each technology has implementation patterns that ensure adequate brokering of identity and authorization from API façade through to application logic, data, and platforms.

Layer Production Protections From End User To Back-End Infrastructure

An API provider may make some APIs for use only by the firm's own developers (i.e., internal APIs) and others for use by external developers (i.e., open web APIs or B2B APIs). With external developers/APIs, the API provider typically has little or no control over the application, but with internal APIs, the right coordination between app and API security models can strengthen API security (see Figure 3). Make sure to include production protections for APIs at the launch stage of the secure-what-you-sell model. In addition to standard network controls, hardened infrastructure, and data security, remember that:

- **External, uncontrolled apps require guardrails tuned to API sensitivity.** For external APIs, external developers, not the API provider, are in control of security at the application layer. And this includes whether or not the app maintains appropriate protection for security credentials and tokens. Lack of control does not mean powerlessness for the API provider; it means that API protections must focus on 1) narrowing channels for compromising security through mechanisms like session length limits, message signatures, and token-channel binding (i.e., accepting tokens only when they arrive through the same network path on which they were generated); 2) observing normal and abnormal usage context and patterns (e.g., approximation of end-user location or time-of-day usage); and 3) requiring end users to go through known apps (e.g., the API provider's website) when authorizing an app to access their data.

- **Known, controlled enterprise apps can go deeper on integrated app-API security.** For internal APIs, where the organization that provides the API also controls the client app (e.g., a customer self-service mobile app), app-API security models can use all of the mechanisms for external apps and more. Sample

mechanisms include obfuscating client-side code and algorithms, app integrity self-checking, environment risk assessment (e.g., checking for a rooted mobile device), minimizing the number of API keys resident on client side, and using client-side software development kits (SDKs) that encode best practices into standard components.

- **API interface and policy governance coordinates with but doesn't trust app security.** The first and best assumption for requests arriving at the door of an API interface is that all input is evil — even if it came from an enterprise-written app. An API gateway, web application firewall (WAF), or both can handle schema validation and attack protection at layer 4 and layer 7 as the first line of defense. Gateways also handle rate limits either as attack protection or API product management measures. Other incoming signals from an app, such as risk assessments, request context, and user identities, may be handled by a gateway or passed through to the API implementation. Pay special attention to tight coordination of the multiple parts of OAuth2 (e.g., scope names, scope descriptions, association of scopes and API operations, and end-user authorization dialogs) to avoid breaches due to confusion and misconfiguration.

- **API implementation security borrows heavily from server-side app security.** Once a request is past an API gateway, it may be handed off to any number of alternative environments that comprise the business logic and data of API's implementation such as legacy monoliths, microservices, integration servers, SaaS apps, and many others. Each will have its own specific security concerns across three major themes: environment and/or runtime hardening, secrets management, and identity handling. If an API implementation runs under an identity shared across client apps, authorizing the end user's entitlement to specific data must be done either by the API code itself, by an entitlement manager, or some other type of framework on the API's behalf. And, if the API calls another downstream API, the same concerns get pushed one level deeper.

- **Intelligent monitoring catches anomalies in production.** Whether it's a string of malformed requests coming from a given IP address (or IP range), a request for large quantities of data, a methodical stream of data-siphoning requests, unusual sequences of requests, or any number of other anomalies, you can't anticipate and set policies to catch every complex attack pattern. However, you can use various AI-based API traffic analysis tools (such as Apigee Sense and Ping Intelligence for APIs) to learn what normal traffic looks like and identify departures from normal — potentially supplementing with additional AI models to ensure coverage across technical anomalies (e.g., failure and data egress patterns), user anomalies (e.g.,

unusual API request sources or times for a given end user), or business anomalies (e.g., unusual business activity patterns across users).

- **Penetration testing finds vulnerabilities that preventive tools miss.** While application security prerelease testing tools identify known flaws during development and testing, application penetration testing tools and services provide an additional layer of assurance by finding unknowns: discovering flaws in how the application runs that could lead to a compromise. For applications built on APIs, penetration testing is a valuable backstop for prerelease testing tools that are just beginning to support APIs. Penetration testing may come from in-house teams or be outsourced to service providers or bug bounty programs. In all cases, make sure that APIs are in scope for the testing.

Figure 3

Sample Tools, Patterns, And Mitigations Across End-To-End API Security Strategy

Source: Forrester Research, Inc. Unauthorized reproduction, citation, or distribution prohibited.

On the client side, there is an external component, which notes the following technologies: request-reply signatures, token-channel binding, and context policies/risk assessment. There is also an enterprise component for client side, which notes the following technologies: app integrity checking, environment risk management, and minimize client-side API keys. The network layer must coordinate and integrate

standard network security with API security. The API interface layer includes API trust and usage policy, rigorous OAuth2 configuration, and layer 7 attack protection. The API implementation layer includes: microsegmentation, identity propagation and brokering, and multilevel authorization. The infrastructure layer must coordinate and integrate standard infrastructure and application platform security with API security.

API Security Requires Multiple Tools And Strategies

Security pros have a range of security tools with which to protect APIs during API design, built into API platforms and implementations, and in production (see Figure 4). API security tools fall into three buckets: API-focused tools with security features, security tools that extend their protections to or may be integrated with APIs, and emerging API security-specific tooling. Some of these tools have overlapping functionality, so examine your particular mix of products to see where the deepest capabilities lie. When building out your API security strategy, consider that:

- **API management and API gateways are the centerpiece of API access management.** API gateways are the primary enforcement point (PEP) for controlling an internal or external app's access to an API. API management solutions, which embed a gateway (or may use other vendors' gateways or service mesh proxies as PEPs), serve three primary roles: 1) API policy definition point; 2) API entitlement registration and credential issuance (along with a broader array of functions for managing relationships between API providers and API users); and 3) governance and collaboration for API lifecycle processes across multiple roles involved in publishing APIs. For development teams that create their own internal APIs, not accessible to others, an API gateway may run stand-alone with local policy configuration (i.e., without API management). Sample API management vendors include Axway, Google, IBM, Software AG, and WSO2.

- **Application security testing tools catalog APIs and identify vulnerabilities.** SAST and DAST vendors are expanding their portfolios to include API testing. Application security testing tools analyze OpenAPI specifications and monitor network traffic to find rogue (unregistered) APIs (so they can be brought under governance), scan APIs for security weaknesses, or test how the running APIs respond to a variety of inputs. Sample vendors include Checkmarx, GitLab, and Synopsys.

- **WAFs create rules based on API configs.** API gateways vary widely in their attack protection features, but a WAF can fill holes left by a specific gateway product. Some WAF vendors have API-specific security features, such as support for OAuth and the ability to configure protections based on OpenAPI specification files. Generating WAF rules from OpenAPI specifications requires that users have an updated specification file to upload, so look for WAFs to build in or integrate with API discovery tools to meet that need. WAFs also offer features like zero-day attack protection, data leakage protection, and rule updates based on vulnerabilities discovered in the application. Sample WAF vendors with API security features include Akamai, Barracuda Networks, F5, and Imperva. Don't be surprised to see overlaps in WAF and API gateway features, and expect that you'll need a deeper dive to understand the depth to which each product addresses each API security requirement.

- **Bot management tools detect and defend against all bots, including API-based bots.** We often think about bot attacks as coming through browsers, but bots also come through APIs. With 10% to 15% of API traffic coming from malicious sources, bot management vendors are extending their detection capabilities to determine the origination and intent of API traffic and defend against API-based bots. Bot management tools apply signal collection and machine learning to identify basic and sophisticated bots, and they offer a range of responses, including honeypots, fake data, open connections, and challenges. Sample bot management vendors include F5 (Shape Security), Imperva, and Perimeter X. Some API gateways have bot detection features; work with your API gateway vendor to determine the extent of their bot detection features and whether it can supplement, support, or feed into your bot management solution — or if you need to rely on a bot management vendor.

- **Application microsegmentation tools evaluate north-south and east-west traffic.** Microsegmentation, at both the network and application layer, is emerging as an architectural best practice for Zero Trust implementations. At the application layer, microsegmentation tools control traffic between application components, whether those components are containers, serverless functions, or APIs. Sample vendors include Avocado Systems, ColorTokens, Guardicore, and Illumio. For container-based applications, the direct use of Kubernetes network policy provides a basis for DIY microsegmentation.

- **API security-specific tools include discovery, scanning, and gateways.** While WAF, bot management, and gateway vendors have all expanded their core use cases to address APIs, other vendors have emerged that focus specifically on the API

security problem. These tools focus on rogue API discovery, monitoring, and scanning. Some tools look at how specification files comply with standards and identify security issues or calculate risk scores. A few API security vendors also offer microgateways. Sample vendors include 42Crunch, CloudVector, and Salt Security.

- **Other tools round out specific niches.** Because API security is a multidimensional problem, and because API implementations and usage scenarios vary quite widely, other tools fill important niches. To judge client-side risks, application shielding and other assessment tools, such as Google Apigee's integration with Google ReCAPTCHA, warn back-end infrastructure to take extra care with an API call (e.g., initiate step-up authentication). An entitlements server handles tricky authorization and data entitlements needs. Service mesh infrastructure, along with microgateways, offer protections for low-level APIs that are typically buried deep within microservice applications. Other adaptations, such as custom policies in an API gateway, fill important holes identified through holistic API security analysis.

Figure 4

Multiple Tools Provide Important API Security Features And Capabilities

Tool/component	Role	Sample vendors
API admin portal (part of an API management solution)	• Enforce authentication, authorization, rate limits, and other security mechanisms • Some provide structures to enable separation of policy specification duties • Some provide rigorous governance controls such as requiring approvals before API go-live	Axway, Google, IBM, Software AG, WSO2
API gateway (stand-alone or part of an API management solution)	• Enforce API trust and attack protection policy • Some provide advanced policy authoring mechanisms such as applying policy based on API metadata • Some have attack protection and bot detection features that overlap with WAF and bot management functions • Some provide value-added features for service mesh, which opens new options for deep, custom API security	Akamai, Axway, Broadcom, Google, IBM, Kong, Software AG, WSO2
API user portal (part of an API management solution)	• As appropriate to specific API policies, allow API users to maintain, e.g., OAuth2 callback URIs or client-side certificates • If needed for custom security models or extensions, provide a channel for additional policy-related attributes or agreements	Axway, Google, IBM, Software AG, WSO2
API traffic anomaly detection	• Apply machine learning models to API traffic to identify abnormal activity	Google Apigee, Ping Identity, Salt Security
Application security testing tools	• Scan APIs for flaws in implementations and definitions • Build API security testing into the development process	Checkmarx, GitLab, Synopsys
Web application firewall (WAF)	• Leverage API specifications to generate rules and block traffic that does not conform to specifications • Enforce OAuth for API authentication	Akamai, Barracuda Networks, F5, Imperva
Bot management	• Detect and defend against basic and sophisticated bot attacks launched via APIs	Akamai, F5, Imperva, PerimeterX
Application microsegmentation	• Set policy for and evaluate application traffic between components, including APIs, containers, and serverless functions	Avocado Systems, ColorTokens, Guardicore, Illumio
API security specialty tools	• Conduct API scanning, discovery, monitoring, and threat protection	42Crunch, CloudVector, Salt Security

Source: Forrester Research, Inc. Unauthorized reproduction, citation, or distribution prohibited.

Tool/component	Role	Sample vendors
Application shielding	• Use authentication, environment assessment, and tamper-resistant tooling to protect applications, particularly mobile applications, from reverse engineering and API exploits at the endpoint	Arxan, OneSpan, Promon
Service mesh	• Enforce policy for APIs and API implementations that are backed by microservice-based components	Consul Connect, Istio, Linkerd
Entitlements management	• Can be integrated via callout mechanisms (either from API code or gateways) to handle complex authorization rules and scenarios	Broadcom, Oracle

Source: Forrester Research, Inc. Unauthorized reproduction, citation, or distribution prohibited.

Build Security Into API Development And Deployment Pipelines

The biggest problem with API security is that the diversity of technologies, layers, designs, and contexts in which they are used means that there is no single answer to API security. The biggest mistake is to address only part of the problem, such as focusing on API façades and ignoring implementations, or attack protection and ignoring identity and entitlements. Two key questions to ask, which drive the analysis of and answers to the greatest number of architecture choices, are: 1) If the API is breached, whether or not through an authorized identity, how quickly will you detect it, how widespread can the damage be, and how will you remediate it? and 2) if some unauthorized action is taken via an API (e.g., a nefarious account credit is issued), whom should you hold accountable to what legal standard, and what forensics will be available to you for that purpose? To work your way through these questions and derive an API security strategy that appropriately balances cost and risk:

- **Start with the big picture: Secure APIs are about more than APIs.** Poor data handling and configuration will come back to haunt you regardless of how you've architected your application. Whether your application is API-first, a classic client/server model, or a combination of both, follow the tried-and-true rules: Default deny, and don't trust client-supplied data. During the design phase, work with dev to define and document data usage, authentication requirements, and access control rules.
-

Put all APIs through secure design and dev, with rogue identification as a failsafe. Successful APIs start with a cultural change: Teams must see APIs as assets for the enterprise, not as their own private assets. You must create a formal API lifecycle that begins with candidate API identification, a system of major business and technical domains (for organizing and classifying APIs), and stage definitions through which an API moves on its way from candidate to production to deprecation and retirement. This ensures that all APIs are known and managed — but implement rogue API identification to ensure any that bypass the lifecycle are brought into governance.

- **Architect to support encryption and performance.** API traffic, like all application traffic, must be encrypted. Firms must carefully select TLS termination points to provide data secrecy according to the needs of the API while avoiding unnecessary performance hits from multiple rounds of decrypting and re-encrypting traffic. To save on performance costs, do it once; a common approach is to decrypt traffic as it enters the API interface layer to enable the API gateway, WAF, and bot management tools to analyze, block, and redirect traffic and then re-encrypt traffic when it leaves the interface layer. For highly sensitive requests, message-level encryption of the payload (or parts of it) can maintain secrecy from the client app deep into the API implementation.

- **Authenticate everywhere; design explicit chains of trust.** From the time an end user first logs into an app, through to data retrieval and update, examine the solution's handoffs from one layer to the next. Design based on clear knowledge of the effective identity at each layer. For example, if a web app knows that Alice is the user but it calls an API using the web app's credentials, the API doesn't know about Alice and can't properly assess data entitlements. If the web app provides Alice's identity, the API may use a pooled, shared identity to call a mainframe service, pushing the problem down a layer. Some business scenarios will have multiple identities in play at the same time (e.g., employer and employee identities when accessing a benefits service). Connect the dots to ensure that the relevant PEP knows and has access to the proper identity or identities to use for authorization (not the identity in effect at some low level of the call chain). If achieving this is too expensive (from a latency or app maintenance point of view), design a monitoring or forensic strategy to mitigate the risk.

- **Remember, the API encompasses the full implementation behind it, not just the façade.** It's a mistake to look at the API as a superficial "interface layer" on top of an application. Everything that goes on behind the API call is both part of the API and part of the application. When securing the API, understand how the API functions

throughout the application (and calls to downstream apps), and ensure that everything it touches and interacts with is protected. Whether the API ultimately touches data, legacy, or modern application architectures, or even hardware or firmware, work through those touchpoints collaboratively with the security owners of the relevant systems.

- **Allow variations in tooling; pursue unified, not singular, API security architecture.** API security requires multiple tools, frameworks, and strategies. It would be nice if you could pick among them, integrate them as needed, and build a single approach to API security that addresses all scenarios. But given the wide diversity of API technologies, layers, designs, and contexts of usage, you will have no such luck. That's OK. Focus first on the overall structure of the API security architecture, create a reference architecture that comprises the multiple tools, and then have alternative elements ready to plug, as necessary, into each spot in the architecture — not willy-nilly, however the implementer wants — but according to prescribed patterns that unify the architecture into a cohesive whole.

API Security Is The Next Big Opportunity For Security Pros

Security pros adjusting to the differences between APIs and classic web applications need to know that it's not all doom and gloom. A properly secured API offers an opportunity: the long-hoped-for positive security model. By only allowing application traffic that conforms to a fully fleshed out API specification, APIs offer a more secure experience than a classic web application. Reaching this summit requires agile API management and smooth integration with deployment-stage security tools.

Supplemental Material

Companies We Interviewed For This Report

We would like to thank the individuals from the following companies who generously gave their time during the research for this report.

42 Crunch

Akamai

Arxan

Barracuda

Cisco

Imperva

Software AG

WSO2

FORRESTER®

We help business and technology leaders use customer obsession to accelerate growth.

FORRESTER.COM

Obsessed With Customer Obsession

At Forrester, customer obsession is at the core of everything we do. We're on your side and by your side to help you become more customer obsessed.

Research

Accelerate your impact on the market with a proven path to growth.

- Customer and market dynamics
- Curated tools and frameworks
- Objective advice
- Hands-on guidance

Learn more.

Consulting

Implement modern strategies that align and empower teams.

- In-depth strategic projects
- Webinars, speeches, and workshops
- Custom content

Learn more.

Events

Develop fresh perspectives, draw inspiration from leaders, and network with peers.

- Thought leadership, frameworks, and models
- One-on-ones with peers and analysts
- In-person and virtual experiences

Learn more.

FOLLOW FORRESTER

Contact Us

Contact Forrester at www.forrester.com/contactus. For information on hard-copy or electronic reprints, please contact your Account Team or reprints@forrester.com. We offer quantity discounts and special pricing for academic and nonprofit institutions.

Forrester Research, Inc., 60 Acorn Park Drive, Cambridge, MA 02140 USA
Tel: +1 617-613-6000 | Fax: +1 617-613-5000 | forrester.com