CEQUENCE® SECURITY

Ten Things Your API Security Solution Must Do

API security is ranked as a top priority in 2022 for enterprises and security leaders worldwide, but in a market crowded with vendors, how can you find the right fit for your organization? With new API security startups showing up on the horizon seemingly every third week, all with different messaging, it can be a struggle to decide which vendor will meet your API security needs.

The Recommended Approach: Shield Right While Shifting Left

The Shift Left movement was designed to improve API security by uncovering and remediating errors during development. Unlike some API security vendors who say Shift Left is failing, we believe strongly these efforts are working and should not be abandoned. However, many experts, including those at Cequence, do not think that Shift Left is a complete API security solution, nor is it a replacement for best-in-class API protection — even a perfectly coded API is susceptible to an attack.

Instead, a Shield Right While Shifting Left approach is recommended. This is where you deploy inline protection for your APIs complemented by visibility, inventory tracking, risk assessment, and remediation. Here's my reasoning: dynamic application security testing (DAST) and static application security testing (SAST) approaches use a battery of tests and techniques to break APIs. However, attackers can always come up with new and innovative ways of attacking your APIs not covered by your DAST tools. In these cases, you need ways to protect your APIs from an attack, but your options are somewhat limited to include:

- **Web Application Firewalls (WAFs)** that protect web applications from known runtime threats missed by static and dynamic code analysis. However, WAFs struggle to protect APIs and are ineffective in defending against unknown and automated attacks.
- **First-generation bot prevention tools** that rely heavily on client-side integration, which cannot be done effectively for an API, making these tools largely obsolete for API protection.
- **API-specific tools** that focus on helping you find and address coding errors before publication but are unable or have limited ability to detect AND prevent attacks natively.

With the Shift Left while Shielding Right approach in mind, here are the ten criteria your API security solution must meet to ensure comprehensive API protection.

One

Provide an Outside-in View of Your API Attack Surface Area and Risk Exposure

Guideline

Look for an API security solution that gives you an attacker's view, from the outside looking in, of your external API assets across all deployment locations including multiple clouds, data centers, service mesh, SaaS, and hybrid. It should do this without requiring the deployment of an agent or any other software. All APIs should be intelligently classified and assigned risk scores to help you prioritize remediation, protection, and deployment steps.

Importance

Modern enterprises have APIs hosted across data centers, multiple clouds, and service mesh environments, some of which may be unknown to security teams. At the same time, organizations adopting faster DevOps cycles will often publish APIs in the environment of their choosing without the knowledge of their security counterparts. Often referred to as shadow APIs, these can cause security gaps and were responsible for many of the API incidents seen in 2021 (e.g., Clubhouse, Experian, and Peloton). These APIs are often in staging mode with access to production data, API performance monitoring servers, third-party APIs, and other sensitive data, exposing them to attackers. Your API security solution must map all your APIs across any hosting environment with the same outside-in view that an attacker might see. Once discovered, every API must be classified and assessed for security risk to help you prioritize which APIs to secure first.

Pitfalls to Avoid

- Stay away from run-time deployment with a limited set of integration points. You want to avoid those that ONLY tie into edge devices, ONLY with load balancers, or ONLY API gateways. This limited visibility may lead to a false sense of security. Other hosting environments could be completely unknown to your security teams and therefore left unprotected from threats.
- Don't confuse this with more traditional Attack Surface Management tools as they are not API-centric and lack the classification and risk scoring capabilities modern enterprises need.

Two

Perform Runtime Discovery and Cataloging of APIs

Guideline

Your API security solution must provide an inside-out view that complements the outside-in view with continuous API discovery, cataloging, and risk assessment at runtime. It should support a variety of API types such as XML/SOAP, REST, and GraphQL. It should also group APIs by department, server, function, and location while allowing you to assign ownership with appropriate access control. Policy-based discovery alerts based on shadow API findings integrated into your SIEM/SOAR system to remove human involvement and improve efficiency are another must-have for an enterprise-grade API security program.

Importance

API breaches commonly occur due to unknown and unprotected APIs. Therefore, discovering shadow APIs in various hosting environments is a critical step in starting your API security program.

Pitfalls to Avoid

- Avoid API security tools that can't handle certain API types.
- Avoid API security tools that compromise privacy and governance policies due to lack of proper access control and grouping of APIs in their dashboard.
- Avoid API security tools that don't have SIEM/SOAR integrations and automated responses to shadow APIs.

Three

Provide Continuous Data Governance and Risk Assessment

Guideline

Your API security solution must continuously assess API transactions for data governance violations. Based on the definition of sensitive data, governance violations can mean different things to those in finance, retail, healthcare, and other types of industries. Regardless, the solution must look for unintentional leakage of sensitive data elements through insecure, unauthenticated, or unauthorized APIs. Another must-have data governance feature is to deny access to certain entities as mandated by your business, regulators, and law enforcement. Most enterprises are banned from doing business with entities located in the OFAC list of countries. Your API security solution must be able to enforce all such restrictions as part of the data governance function.

Importance

Unintentional sensitive data leakage or theft via an attack can result in significant financial and regulatory violations. Your API security solution must ensure that you can continuously assess data usage by APIs and initiate remediation and prevention steps when violations are discovered.

Pitfalls to Avoid

- Avoid Inflexible and opaque solutions that result in high false positive rates due to lack of tunable detection based on industry, use-case, and context.
- Be wary of complete reliance on ML/AI with no manual override possible, which can lead to a large number of false positives.
- Stay away from solutions that rely on third-party enforcement and can result in extended data exposure and governance violations.

Four

Assess and Enforce API Specifications Conformance

Guideline

Your API security solution must use API specifications to protect APIs in three different ways:

1. Integrate with CI/CD pipelines to consume API specifications in pre-production environments, looking for potential vulnerabilities before APIs go live.
2. Validate runtime API behavior against the specifications, flagging any deviations and initiating alerts for remediation.
3. For legacy APIs where no specifications exist, the solution must generate API specifications based on the runtime behavior of APIs and repeat steps one and two for those.

Importance

Organizations are rapidly adopting API specification frameworks to improve quality, consistency, and security at runtime. Sometimes though, developers will deviate from specifications or forget to update them as new APIs are published. Specification non-conformance can introduce weaknesses and risks that the security team doesn't know about and can't address. The same is true for legacy APIs that are live with no specifications associated with them. Therefore, runtime specification validation, conformance enforcement, and generation are a must-have for your API security solution.

Pitfalls to Avoid

- Avoid API security tools that do not integrate with your choice of CI/CD pipeline.
- Don't get locked into an API security tool that cannot generate specifications in the format you have standardized – Swagger, OpenAPI, etc.
- Stay away from API security tools that can only initiate alerts on schema non-conformance and are unable to enforce schema conformance.

Five

Natively Protect Against All OWASP API Security Top 10 Threats

Guideline

Your API security solution must include comprehensive and native protection against all the threats listed in the *OWASP API Security Top 10*.

Importance

APIs and the environment hosting them are vulnerable to similar threats that targeted web applications in the late 90s. As developers migrate business logic from web applications to APIs, the attackers migrate there as well. APIs are easier to attack as they are often well documented via published specifications.

Pitfalls to Avoid

- Avoid signature-based tools like WAFs that attackers can easily bypass.
- Stay away from solely DevOps-focused tools and lack a balanced approach to preventing threats while code is being fixed.
- Avoid using a vendor's own test suite to validate this functionality. Create a battery of tests based on your APIs and the threats that can target them.
- Avoid API security tools that cannot natively mitigate threats and rely solely on third-party tools such as WAFs. Such tools cause a time delay between detection and response and higher than average false positives, which leave APIs vulnerable to compromise.

Six

Stop Automated Business Logic Abuse

Guideline

Look for an API security solution that can protect your APIs from all forms of automated business logic abuse. Common examples of business logic abuse include:

- The enumeration of repeated alpha-numeric fields in API calls.
- Adding extra parameters to API calls.
- Setting unsupported parameter values in API calls.
- Forcing APIs to return verbose errors and leak sensitive information.

Pick appropriate test cases to evaluate vendors' ability to stop business logic abuse.

Importance

Attackers often use enumeration techniques to perform API reconnaissance, looking for potential targets. Once they find a potential business logic abuse target, new forms of automation are used to execute an account takeover attack (ATO) or fake account creation. Therefore, stopping business logic abuse early in the attack cycle is key to success.

Pitfalls to Avoid

- Avoid signature-based tools, like WAFs, that are easily bypassed by attackers.
- Avoid offerings that rely solely on sending signals to third-party systems such as WAFs. Such tools cause a time delay between detection and response, which leaves APIs vulnerable to compromise.

Seven

Detect Automated Bot Attacks and Fraud

Guideline

APIs enable attackers to execute all the attacks listed in the OWASP top automated threats, often resulting in fraud and other business-related losses. Your API security solution must be able to detect signs of malicious automated intent hiding in plain sight by analyzing behavior in real time using ML/AI.

Importance

Bot attacks directly impact an organization's bottom line, resulting in fraud, increased infrastructure cost, poor customer experience, and inefficient marketing campaigns. APIs allow bots to send high volumes of requests that appear legitimate and pass most security controls, but as a group, they cause significant damage. You need an API security solution that prevents bots out-of-the-box with high efficacy rates.

Pitfalls to Avoid

- Steer away from first-generation bot mitigation solutions that require application changes and integrations to collect attack telemetry.
- Avoid API security upstarts that don't have depth in their bot detection capabilities and don't cover all types of bot attacks.

Eight

Natively Mitigate Threats in Real Time

Guideline

Select an API security solution that can natively detect AND mitigate threats without relying on any third-party solution integrations. The vendor should offer a variety of response options, including deception that is configurable by users based on APIs and threats detected. The solution should also support configurable automated responses for threats based on pre-defined policies to reduce administrative overhead and exposure time.

Importance

Native mitigation as compared to reliance on third-party solutions is critical for the following reasons:

- Native mitigation reduces the time window between threat detection and response. It should be possible to eliminate this time lag completely by having predefined threat policies and responses in your API security solution.
- The third-party solution response options may be limited to simple IP addresses and basic WAF rules, which attackers can easily bypass.
- Depending on the network architecture, the third-party solution may not see the same API traffic as your API Security solution, making it useless from a response point of view.
- The use of a third-party solution for mitigation creates dependencies and challenges with upgrades and vendor changes.

Having flexibility in your response options is critically important to your API security strategy. Simply blocking API requests is not always an ideal response as it may encourage attackers to try different methods and may not be feasible from a business point of view. For example, a business may detect a data governance issue with a business-critical API, so blocking access to that API may not be an option. Therefore, the API security solution must ensure the weakness is not exploited while also allowing legitimate API use during the time it takes the DevOps team to roll out a fix.

Pitfalls to Avoid

- Avoid tools that rely on third-party solutions like WAFs for their mitigation response.
- Stay away from tools with blocking as the only response option.
- Avoid deployment architectures where detection and mitigation components don't see the same API traffic.

Nine

Deliver Enterprise-Grade Performance, Availability, and Scale

Guideline

Make sure your API security solution can handle your business's scale and performance requirements. If you are a large enterprise with many hosting environments, with many APIs generating large volumes of API transactions, the solution should scale with no impact on performance and availability. Establish a success criterion for scale and performance based on your current and future needs and develop a test suite that evaluates solutions against it. Your API security solution must be highly available (H/A) and support H/A as a deployment option. Test for fail-over scenarios with a H/A set up to ensure the solution does not fail open or fail close when experiencing downtime.

Importance

Often enterprises choose security solutions after a proof of concept that is limited in scale, and then the vendor struggles with performance and availability under full production load. This results in time lost and spent troubleshooting and re-deploying for both the enterprise and vendor. The result is a sub-optimal outcome from the decision, deployment delays, vulnerable APIs, performance impact on production APIs, and cost over-runs. Establishing clear performance and scale goals then testing against them before you select your API security solution can save a tremendous amount of time and money in the long run.

Pitfalls to Avoid

- Avoid testing API security solutions in small DevOps environments and not at a production scale or based on the future needs of your organization.
- Steer clear of vendors with no H/A support.

Ten

Ensure *Your* Application Architecture and Data Residency/Privacy Requirements Are Supported

Guideline

Make sure you can deploy a vendor across all your API environments with a unified management portal across them. If not, these environments can be discovered by an API attack surface discovery tool mentioned in the first criteria. Additionally, ensure your API security solution integrates with your current and future choices in CDNs, API gateways, load balancers, service mesh architectures, and cloud providers. Regardless of your deployment choice, your API security solution must support data residency and data privacy requirements as well as regulations applicable in all the regions where your APIs are hosted. That may require support for SaaS, on-premises, and hybrid deployment options. Validate data residency and privacy compliance as part of your vendor evaluation. When applicable, ensure your API security vendor has appropriate industry certifications like PCI, SOC2, ISO, etc.

Importance

SaaS-based API Security solutions may send customer data to hosting environments located elsewhere for analytics. That hosting environment's geolocation and certifications may compromise your data residency and data privacy compliance. Data residency and data privacy law violations can cost organizations millions in fines and prevent them from doing business in certain parts of the world.

Pitfalls to Avoid

- Avoid evaluating in a single environment type if your organization's infrastructure is heterogeneous.
- Don't take the vendors' word for it regarding data residency and data privacy— validate claims with tests and documentation to ensure it meets all your needs.
- Avoid SaaS-only solutions where data residency and data privacy laws are strict.
- Avoid accidental exposure to GDPR fines by shipping PII data to a third-party cloud.

Conclusion

Organizations are rapidly adopting an API-first development methodology because of APIs' power, flexibility, and efficiency. The shopping, finance, manufacturing, and marketing apps we use every day are all based on APIs, connecting back to compute resources located elsewhere — the cloud, the data center, or both. Unfortunately, threat actors love APIs for the same reasons that developers do. APIs are susceptible to a range of automated attacks and vulnerability exploits that can lead to data loss and system compromise. To protect existing and future APIs, organizations need to implement a forward looking API security solution that unifies runtime API visibility, security risk monitoring, and patented behavioral fingerprinting technology to consistently detect and protect against ever-evolving online attacks.

Protect Your APIs While Empowering Your Developers with Cequence Security

Schedule Your Cequence API Security Platform Demo at cequence.ai/demo

CEQUENCE® SECURITY

100 S. Murphy Avenue, Suite 300, Sunnyvale, CA 94086 | 1-650-437-6338 | info@cequence.ai | www.cequence.ai