Protecting APIs From Modern Security Risks

Table of Contents

Overview

Today's applications are different from those built just a few years back. A significant difference is that APIs are increasingly being used to power new customer-facing applications, connect with partners, and drive microservices environments. APIs are everywhere in application environments, and they increasingly expose and exchange sensitive data, making them a growing target for attackers. These changes in APIs and applications explain why we've seen a significant increase in attacks targeting APIs in recent years.

Sophisticated attackers have moved beyond common cross-site scripting (XSS) and SQL injection (SQLi) attacks and have shifted their focus on finding unique vulnerabilities in APIs. Traditional tools like WAFs and API gateways depend on signatures to identify attack patterns and can't detect or prevent new attack methods that target the unique nature of APIs. These traditional tools are based on proxy architectures and lack the granular understanding of each unique API needed to protect from modern attacks.

As with anything in security, there is no silver bullet to protect you from API security risks. API security requires a combination of capabilities as part of your security strategy to provide the right level of protection.

Why APIs Are Difficult to Secure

There are many factors to consider that make APIs challenging to secure. Their uniqueness, constant change, and focus of developers all make APIs a unique challenge to protect in any environment. Consider the following as you build your strategy to protect your APIs, applications, and data:

Each API Is Unique

Each API is unique in its structure, parameters, endpoints, and behavior. Given this, each API has unique vulnerabilities to consider when building a security strategy. Since every company creates its own unique APIs, there is no standard to protect them to avoid exposing them to vulnerabilities. This is precisely why a security solution that understands your API's unique logic is needed to provide proper protection.

APIs Constantly Change and Evolve

In today's Continuous Integration/Continuous Delivery (CI/CD) environments, companies are developing new versions of applications faster than ever, and API's are continuously changing. Consider a new version of an API is released. Now one parameter in that API has changed out of thousands of parameters. Changes like this happen all of the time, and more often than not, these changes are completely transparent to security staff tasked with protecting APIs.

An average API has more than 6,000 different parameters that lead to many actions, so the potential for vulnerabilities is massive. Combining the constant rate of change with the number of API parameters and each API's unique nature makes ensuring the right level of security exponentially challenging, if not impossible.

Developers Don't Think Like Attackers

Developers excel at writing code, developing new features and are increasingly mindful of security, but they don't think in the devious ways that attackers do. Developers strive to make applications work in efficient and scalable ways. They're motivated to develop efficient code and new functionality. On the other hand, attackers look for ways to manipulate applications, APIs, and their logic in unintended ways to reach their goal.

Also, consider that there is little incentive for developers to achieve 100% security for their product. That's an intangible goal starting with an unknown number of potential gaps and vulnerabilities. A developer's goal is to quickly deliver the best code and bug-free to help deliver the next release on time. Developers are not able to guarantee that every gap in the API is secured. As the saying goes, security has to be right 100% of the time, attackers only once.

Security and Developers Need Better Synchronization

Security and development teams often seem as if they're working against each other with their charters and goals at odds. Development must move fast to deliver applications and updates quickly. On the other hand, security can be seen as slowing progress, requiring any new application or feature to pass stringent requirements before being released.

This lack of perceived alignment between teams means one side has to compromise. For example, when security identifies a vulnerability, it can be challenging to agree with development on the details and priority for remediation. Seeing eye to eye can take time, leaving applications vulnerable.

Current Approaches to API Security

As with all things security, there is no silver bullet to protect you from API security risks. It's a combination of efforts at the end of the day that need to be part of your security stack. Here are some things to consider as you think about your security strategy to address modern threats.

Authentication

While authentication is foundational for any security strategy, keep in mind that it's a low bar for an attacker to step over when it comes to API security. Relying on authentication can give you a false sense of being truly secure. For many applications, especially those that are public-facing, signing up to get legitimate credentials is a trivial process. In a matter of a few minutes or less, an attacker can have a legitimate username and password and begin poking around your API. For public APIs, relying on authentication is like putting a lock on your door and then giving the keys to anyone who asks. The barrier to entry for developer-facing APIs isn't much higher. We can't discount how trivial it can be for a determined attacker to get a set of credentials through phishing or other means.

Authorization

After authentication of a user, what they are allowed to do is controlled through authorization. For example, authorization defines if a user or entity is authorized to make a specific API call to request data from a database. While authorization paired with authentication may seem like the end all be all for API security, implementing and maintaining authorization is easier said than done. APIs create a complex web of logic that is unique to each application. The logic within a single application alone is enough to make a security admin's head spin but think about how that complexity increases as you connect multiple applications via API. Add on top of that the various roles and authorization requirements of users, admins, developers, etc., and that web gets even more complicated. Ensuring you have comprehensive authorization policies in place that don't break your application, especially in a constantly changing environment, can seem like a never-ending impossible task.

Rate Limiting and Throttling

Another approach to API security is to limit the amount of activity that can be performed against the API by a user or entity. Rate limiting can help stop volume-based distributed denial of service (DDoS) and credential stuffing attacks or stop bots from scraping and manipulating site data. While these types of attacks are of legitimate concern, they do not apply to every application.

Web Application Firewalls

The heyday for Web Application Firewalls (WAFs) has arguably come and gone. Many of these tools came to fruition when attacks like cross-site scripting (XXS) and SQL injection (SQLi) were prevalent, and signatures could identify these predictable attacks. A combination of development frameworks and developer best practices have significantly reduced the prevalence of these vulnerabilities causing attackers to look elsewhere for ways to attack an application.

Another challenge with WAFs is that they depend on a combination of signatures and proper configuration to be effective. Using signatures means WAFs only look for known attacks based on predictable attack patterns. Also, keep in mind that signatures need to be kept up to date to block the latest types of attacks.

Configuration allows for customization to help align these products with some of your application's uniqueness. Configuration, however, requires someone with in-depth knowledge of the application to configure the WAF properly to ensure that it's completely effective. As with signatures, the configuration needs to be kept up to date to keep pace with your application's latest threats and changes.

Shifting Left

Development organizations have taken a host of approaches to improve the security of anything that gets released to production. These approaches include code scanning, penetration testing, and making developers more security-minded with security training. These approaches help, but they are also complex, time-consuming, and expensive while still leaving gaps. For example, because of the expense of penetration testing, you might only focus tests on a specific application area, leaving the rest of the application untested.

Code scanning often results in a large list of vulnerabilities that may not be exploitable and, therefore, a lower priority. Sorting these from higher priority vulnerabilities to fix them with limited development resources can be a challenge.

Finally, when it comes to developers, anything that helps them improve security is essential. Still, a developer will never think like an attacker who looks for devious ways to misuse APIs and exploit vulnerabilities. That's not to say secure development practices efforts are wasted; rather, they're not enough, especially considering the types of attacks becoming increasingly common.

You're Still at Risk

While all of the previous section approaches provide essential layers of security, each can leave a gap for attackers who often use subtle methods to garner significant results. For example, a single attacker with a carefully crafted API call can get an application to give up a trove of sensitive, personally identifiable information (PII). This type of breach not only has customer confidence and PR implications but can also result in a financial hit with compliance-related fines.

Denial of Service (DoS) is another threat to your applications. Like with the data exfiltration example, an application can be overwhelmed to the point of an outage using a carefully crafted API call. This type of DoS does not require the time or expertise needed to orchestrate a distributed attack and can be done in such a way that it is missed entirely by other solutions in your security stack.

Account takeover is another potential goal of an attacker. As with the other examples, subtle changes to an API call can result in an attacker gaining access to any other user's account, giving them the ability to access sensitive data, change account information, or even make fraudulent charges.

These examples only scratch the surface of what attackers are after when they target your APIs. With subtle changes to API calls, these attack attempts are missed by the current solutions in your stack yet still have a significant impact.

Key Components of a Strong API Security Solution

The evolution of applications has led to increased API usage, which has created new and unique vulnerabilities for your applications and unique opportunities for attackers. While traditional security approaches may still have relevance and provide value in your security stack, the following are required capabilities to ensure protection for your modern applications and APIs.

Signature Free, No Configuration Required

A security solution that depends on signatures or configuration is not realistic with today's applications. As outlined earlier, applications are constantly changing, and the logic for every application is unique, meaning signatures are ineffective. Configuration requires time, expertise and is prone to error or, at the very least, will likely be incomplete. Security solutions need to learn the unique details of your environment autonomously and keep up with changes automatically to be effective.

Continuously Updated Inventory

Creating a comprehensive inventory of the APIs you know and including those you aren't aware of is critical to understanding your risk and properly aligning your security resources. Applying automation to maintain an up-to-date catalog helps keep pace with dynamic development practices and application teams releasing new, unknown APIs into the environment. Automation helps security teams see APIs as they're deployed, stay up to date as changes are made, and verify that all APIs meet security requirements. Combining this catalog with visibility into the types of content exposed through your APIs, such as PII and other company intellectual property (IP), can be extremely powerful when it comes to aligning your security strategy, resources and meeting compliance requirements.

Attack Detection and Prevention

Today's API attacks focus on targeting unique API logic with subtle methods that bypass traditional security solutions. To detect and prevent API attacks effectively requires a deep understanding of the logic of each API. Based on this understanding, a solution must learn and baseline the typical behavior for your APIs and look for subtle deviations from that typical baseline to identify and stop attacks. Deviations fall into two categories - those that are harmless and those that are malicious. The solution must differentiate between the two, highlighting only those that are truly malicious to help security teams focus on and quickly block real attack attempts.

Separating harmless deviations from truly malicious activity is key to helping security teams focus and identify actual threats. Providing security teams with comprehensive insights into each attacker's activity helps to reduce false positives and ensure a fast response to block high priority threats. Providing this insight early in the attack cycle is critical to stopping an attacker before they're successful in reaching their ultimate goal.

Eliminating Vulnerabilities

The best way to improve security is for developers to eliminate vulnerabilities. Identifying, prioritizing, and eliminating vulnerabilities can be easier said than done since developers' primary focus is on creating new features and innovating, often under tight schedules. Providing clear, concise insights into why a vulnerability exists, where it exists, and how best to resolve it helps to ensure an efficient remediation process. Integrating these insights into existing developer tools and workflows further improves efficiencies and bridges the gap between security and development teams.

Leveraging attacker activities to identify vulnerabilities helps in two ways. First, it lets you use an attacker's efforts against them, almost as if they were penetration testers working on your team. Next, it helps you focus remediation efforts on the real, high severity vulnerabilities, not theoretical, corner case vulnerabilities that scanning solutions may identify. With these insights, you can ensure you are making the most of limited developer resources.

Summary

The evolution of applications and increased adoption of APIs has changed the game for application security. Traditional methods that once provided protection are no longer enough to protect from modern threats. Security strategies must take into account the evolution of applications and the evolution of attackers who now target the unique logic of applications. Security solutions cannot depend on signatures or configuration to provide protection. Solutions must provide comprehensive and up to date visibility in dynamic, changing environments. Solutions must also identify and stop subtle attacks targeting the logic of applications and finally provide insights to help teams eliminate APIs vulnerabilities to continuously improve the security posture.