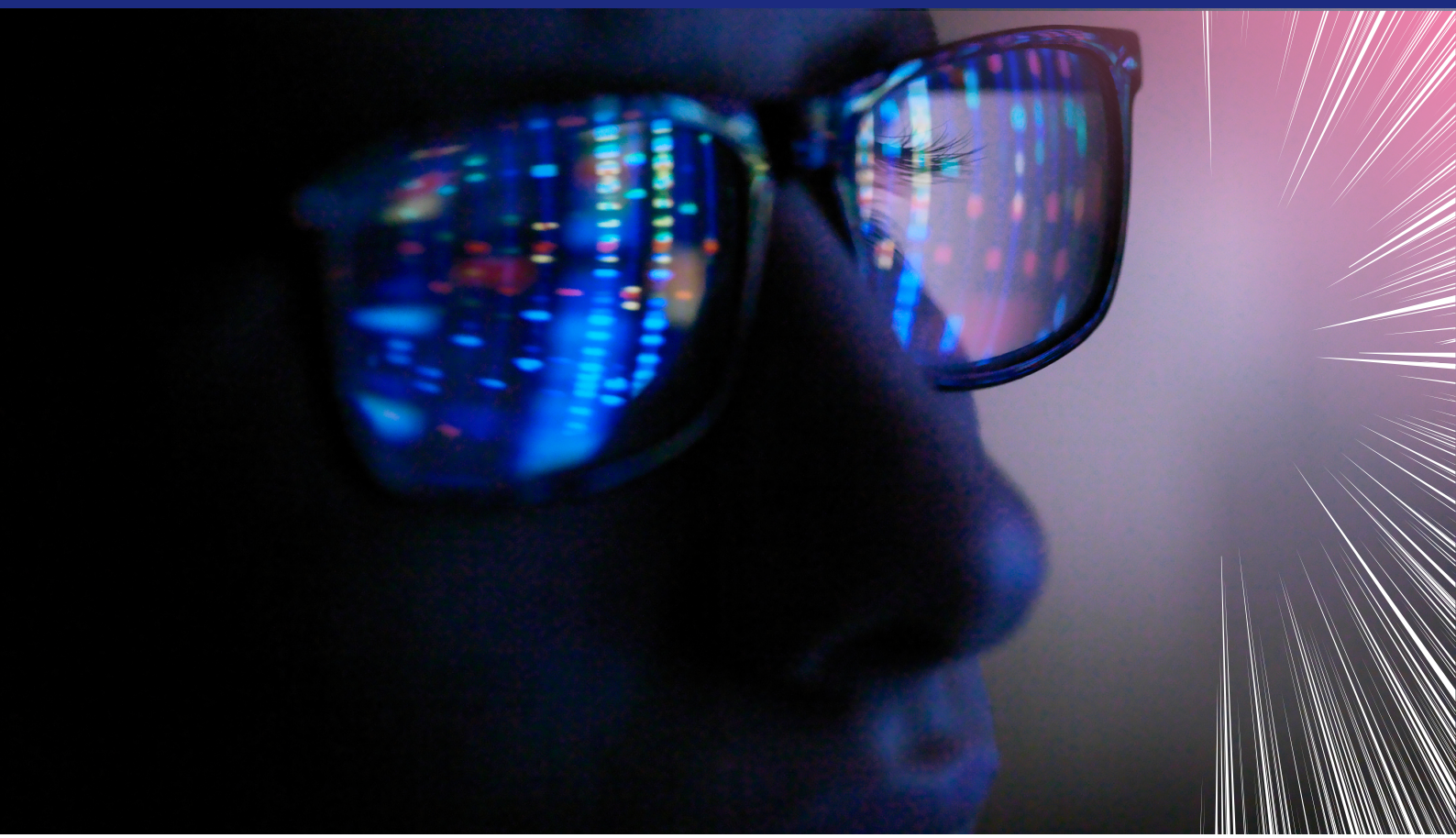
Checkmarx
The world runs on code. We secure it.

eBook

A Guide to Modern API Security

Checkmarx EXPERT PARTNER

CERT2CONNECT
CYBER & CLOUD SECURITY



Introduction

APIs are like highways: They're everywhere, and although they provide the foundation for interaction between all manner of resources, they also pose challenges.

With APIs, those challenges include, first and foremost, security risks. The more APIs you use, and the more complex your API architectures, the harder you'll need to work to ensure that security issues with APIs don't undercut the value that APIs bring to your applications and infrastructure. Issues such as weak API authentication, injection attacks against API endpoints, API sprawl and more can turn APIs into the weakest link in your security strategy, if you don't address these risks effectively.

That's why we've prepared this e-Book: To help organizations understand the new types of security risks that arise from APIs, as well as how to handle them. Without going too far into the weeds, the following sections provide a technical walkthrough of how APIs work, which security challenges they create, and which best practices developers can follow to contain those risks.

If you use APIs today – which you very likely do if you are leveraging cloud-native technology in any way – mastering the concepts discussed in the following pages is crucial for ensuring that you can balance the benefits of APIs with security challenges. Indeed, avoiding APIs is simply not realistic for most development teams today, given how central APIs have become to application architectures and cloud-native deployment environments. This e-Book helps developers stop living in fear of API security issues so that they can take full advantage of APIs, while simultaneously creating applications that are as secure as possible.

Table of Contents

The Basics of API Sprawl and API Security Risks 4

- Why Are APIs So Popular? 4
- The Risk of API Sprawl 5
- Sprawl for Internal vs. External APIs 5
- Managing Security in the Age of API Sprawl 5
- Section Summary 6

Managing Your API Attack Surface 7

- How Do Documentation Tools Like Swagger Fall Short? 7
- Why It's Hard to Keep Track Of APIs 8
- How Microservices Architectures Lead to API Communication Flows 8
- Not Knowing All Your APIs Leads to Catastrophe 8
- Section Summary 9

Understanding the Top API Security Risks 10

- Why API Security Matters 10
- Major API Security Risks According to OWASP 10
- Beyond OWASP: Additional API Security Best Practices 11
  - Require Access Control 11
  - Avoid API Sprawl 11
- Section Summary 11

Addressing API Security Challenges 12

- Reasons Why APIs Are Hard to Test and Secure 12
- Tools and Approaches That Work Well Together – and Some That Don't 12
- How to Effectively Scan APIs 13
- Ways to Address the Continuous Development of APIs and Ever-Changing Contracts 14
- Section Summary 14

Conclusion 15

The Basics of API Sprawl and API Security Risks

To understand why API security is so critical for modern developers, you must first understand how central APIs have become to modern development; and why widespread use of APIs has led to a phenomenon called API sprawl.

APIs have been ***around for decades,*** and so have API security challenges. What has changed over the past seven or eight years, however, is that usage of APIs has surged.

If you had to find an analogy for this explosive growth of APIs, you might choose to compare them to cell phones. Like cell phones, APIs were once considered a niche technology for use in specific, limited circumstances. And now, they're everywhere. Just as the total number of cell phones in the world ***increased*** from a couple of million circa 1990 to around 2.5 billion today, the total APIs in existence has grown from a few dozen in the early 2000s, ***approaching 1.7 billion active APIs by 2030.***

In most respects, the proliferation of APIs is a great thing. But, like the proliferation of smartphones, it also presents a major challenge in one key aspect: Security. More APIs means more opportunities for attackers to exploit API security vulnerabilities – especially if organizations fail to keep track of which APIs they are using.

Let's explore this phenomenon, which is sometimes known as API sprawl, and what it means for modern security strategies.

Why Are APIs So Popular?

The fact that there are so many APIs in existence today is not an inherently bad thing. On the contrary, APIs provide a number of key benefits to developers and users, such as:

- **Integrations:** APIs make it easy to share data between applications and services.
- **Distributed environments:** Because APIs serve as the glue that binds together discrete microservices, cloud services, and so on, they play a vital role in allowing developers to take full advantage of modern, distributed infrastructure. By extension, they help enable resilient and scalable environments.
- **Lower development effort:** In some cases, APIs allow developers to incorporate functionality into software by borrowing it from third-party services instead of having to write it themselves.
- **Simplified user experience:** APIs can drive better user exp eriences by allowing users to share data with multiple applications seamlessly, for example, or sign in once to access multiple applications.

The list could go on, but the point is clear: There are numerous factors today that encourage developers to use APIs extensively. That's why it has become common today to talk in terms of ***"API-first"*** development and design, which means APIs lay the foundation for the way developers design and build software.

The Risk of API Sprawl

Yet, there is one major downside to the surge of API adoption in recent years: API sprawl.

API sprawl is the use of APIs to such a large extent that businesses struggle to keep track of which APIs they are using, and which security vulnerabilities may linger within those APIs.

To go back to the cell phone analogy, you could compare API sprawl to what happens when businesses adopt overly liberal Bring Your Own Device (BYOD) policies for their employees by allowing workers to use personal mobile devices at work. If businesses don't enforce strong governance policies regarding exactly how mobile devices can be connected to their networks, they may end up with a situation where they struggle to keep track of which third-party devices are in use within their environments, let alone whether those devices are secure.

API sprawl is similar in the respect that, if a business uses too many APIs without systematically tracking where and how they are used, it becomes very difficult to ensure that those APIs are used securely.

Analyst firms like ***Gartner point to API sprawl*** – and the security issues it introduces – as a major issue that businesses will need to address as they continue to make use of APIs.

Sprawl for Internal vs. External APIs

It's worth noting that API sprawl challenges affect both internal APIs (meaning those that a company develops in-house to connect its own microservices or applications) and external APIs (which are APIs created by third parties to support integrations with outside resources).

In some senses, external APIs pose a greater threat with regard to API sprawl because it's easier for attackers to discover and abuse external APIs. But internal APIs, too, can be exploited by attackers who identify flaws within them. For instance, an internal API could be abused in order to escalate a breach from one application into other applications that integrate with the breached application using an internal API.

The point here is that, even if you don't use external APIs (or you use them sparsely), it's critical to make sure that you know which APIs you are using and how they are being used, so that you can react quickly to security issues that arise with any type of API on which you rely.

Managing Security in the Age of API Sprawl

Faced with the security challenges that arise from API sprawl, what's a business to do?

The answer is clearly not to stop using APIs. While that would mitigate API-related security issues, it would deprive businesses of the many benefits that APIs offer.

A better solution is to use APIs as often as you like while making sure to manage the security risks that they introduce. Doing so hinges on a few key practices:

- **API governance:** You should include rules for APIs within your organization's governance policies. The rules should explain when both internal and external APIs may be used by your developers, and which security practices (such as the ***OWASP API security recommendations***) need to be followed when using those APIs. Your governance policies should also ensure that developers systematically document which APIs they are using and where, so that it's easy to know which systems are affected by an API security vulnerability.
- **Track API security vulnerabilities:** Keep track of disclosures about API security issues for any external APIs you use. (For internal APIs, you'll need to identify vulnerabilities yourself, because there are no disclosures by third parties about APIs you develop and use yourself.)
- **Monitor your APIs:** In addition to following disclosures of API security issues, continuously monitor APIs to detect usage anomalies that could signal abuse.

Practices like these help to establish a happy medium with regard to APIs. They let you take full advantage of APIs while mitigating the risk of API sprawl.

Section Summary

Just as there's no avoiding cell phones today, not using APIs is simply not an option for most businesses. That's why it's critical to use governance strategies and security tools to mitigate the security risks that can arise from API sprawl. With a little effort, you can benefit fully from APIs without letting APIs undercut your business's security.

Managing Your API Attack Surface

You might think that taming API sprawl and staying on top of API vulnerabilities shouldn't be that hard. Can't you just document which APIs you use, then refer to the documentation to assess your API risks?

In theory, you can. But in practice, managing API sprawl – and minimizing your API attack surface, which means minimizing the potential for your APIs to be exploited – is much easier said than done. This section explains why, and offers tips on what developers can do about it.

How Do Documentation Tools Like Swagger Fall Short?

Documenting APIs comes with inherent difficulties. The main problem areas include ambiguity of exposed functionality, incomplete or undocumented content, and incorrect responses. Development teams rely on tools like Swagger to help them document their APIs and use automation to mitigate some of those issues.

Swagger is not a panacea, however, and using Swagger/OpenAPI for API development does not protect you against all vulnerabilities. Recent ***research conducted by Soufian El Yadmani of Cybersprint*** found many flaws in this technology, including many vulnerabilities listed in the OWASP API Top 10.

For example, issues can happen when a web framework is integrated with Swagger. Developers using the ***FastAPI framework*** can receive automatically generated Swagger UI fields from the code without declaring the API specification. This works fine on paper, but it might not work with your particular business requirements. The CRUD model might return a different result based on certain circumstances, for instance, or use a different way of calculating the end result that deviates from a typical case. The framework might be unable to introspect the correct result and rely instead on manual intervention. In this case, developers will have to completely own the dependencies and generated documentation to ensure that they match the expected outcome.

In addition, Swagger generates complex code that has little opportunity for customization, which makes it inconvenient to use. For example, if hypermedia links are missing from the response, developers might try to intervene by writing custom queries that deviate from the spec, thereby increasing the risk of producing undocumented responses.

Overall, the hardest part is establishing proper workflows for working with Swagger in the optimal way – writing specs in YAML, implementing the specs, and writing unit tests that conform with those specs. The API maintainer's job is to make sure that they don't deviate from the supported features and that they keep the spec as an authoritative source of truth.

Why It's Hard to Keep Track Of APIs

There are many reasons why APIs are hard to keep track of, and therefore hard to support at scale, but they all relate to technical debt. Working on many products as part of a broader ecosystem is a very typical scenario. Some parts of the system might be unknown due to a change in business priorities and focus, when some API services were neglected and forgotten – until they failed to work. This is an example of code ownership debt.

Another type of technical debt is people debt. This happens when you allow developers to work on critical API systems for a long time and then they decide to resign from the company. The domain experience that these people acquired when building those systems may be lost to future maintainers unless there is a good handover process. Having many API services and no one who understands how they work significantly contributes to the initial problem.

The issue becomes even bigger when you introduce architectural debt by implementing APIs using microservices. Let's explain.

How Microservices Architectures Lead to API Communication Flows

APIs and microservices are directly related. As you develop applications using microservices, you create highly decoupled services that enclose their own domain and communicate with each other using APIs. For example, a user microservice needs to communicate with the auth microservice to authenticate the user before responding to a request to view that user's profile. This communication dictates the use of an API contract between each microservice. In addition, API gateways can be used to aggregate multiple APIs under a single namespace, which helps maintain observability and centralized monitoring at a fundamental level.

More real-world domains may need many microservices. In practice, that means having a separate API layer behind each microservice – and each one has its own attack surface. It's not unusual to have hundreds of microservices that each expose OpenAPI interfaces. This leads to what is known as API sprawl. In this case, the sheer number of services makes things harder to maintain. It's very common for businesses to use APIs without tracking when or where they are used, which makes it harder to keep track of their security risks.

Not Knowing All Your APIs Leads to Catastrophe

It is a well-known fact that you cannot secure something when you don't know it exists. But achieving continuous runtime visibility into all APIs is not trivial. It requires that you understand the exposed parts of the API in depth and that you document the obscure sections, run static code analyzers, and subject the application to security testing. This is mandatory, since the failure to record and secure these parts is very dangerous.

APIs are already extremely susceptible to many kinds of attacks. Undoubtedly, there is no limit to what attackers can use when it comes to stealing sensitive data for malicious purposes. Here are some examples of attacks that you may encounter:

- **DDoS:** Attackers can target unknown parts of your APIs to do the maximum damage. They can overload the system with terabytes of bandwidth if they're successful, and you won't know what hit you. It's therefore important to safeguard against DDoS attacks on each exposed part of your APIs.
- **Innocuous access:** Hidden or undocumented parts of your API can be easier to exploit. By using unknown endpoints, for example, attackers can dig in quickly and gain unauthorized access without someone detecting anything suspicious. Unless the traffic from those endpoints is more consistent, it will be harder for the security teams to recognize the danger.
- **Injection:** API endpoints that are vulnerable to injection attacks (SQL, XSS, and so on) can help attackers expose sensitive data, leak credentials, and gain insights into how to attack the infrastructure. If the API under attack is unknown or undocumented, you will lose valuable time trying to figure out how to safeguard it.

Take extra care when you develop and expose APIs – especially the ones that are used for public access. These can be used as target practice for all sorts of attacks. Conducting a solid security assessment of your APIs will position you one step ahead of any future attacks.

Section Summary

Just because you use Swagger, follow the best practices for documentation, and lead by example does not mean that your APIs are secure by default. It literally takes a village to achieve a superior level of security – mainly because of the inherent difficulty of covering the attack surface of your APIs.

This is why you need to establish an end-to-end API security strategy, complete with security testing, extensive discovery of available APIs, and the use of modern tooling and monitoring services. With some effort, you will improve the security posture of your APIs without hurting their overall performance.

Understanding the Top API Security Risks

As the last section explained, no matter how well you document your APIs, there's no guarantee against API security issues. That's why it's essential to understand the specific security risks that can impact APIs and know how to respond to them.

Toward that end, this section walks through different types of API security risks, as identified by organizations like OWASP.

Why API Security Matters

If you look at some of the most significant cybersecurity breaches in the past, you'll notice that many of them were facilitated by API security issues. Examples include:

- The ***T-Mobile breach,*** in which attackers exploited an API to scrape customer data.
- ***LinkedIn's exposure of 700 million customers' data,*** also due to an insecure API.
- A vulnerability ***in a Coinbase API*** that allowed attackers to essentially sell an inexpensive coin for BTC due to insufficient API data validation.

The whole point here is that even some of the largest organizations with a significant focus on software security can fall prey, since API-related security breaches are becoming widespread today.

In a way, that's not surprising. API adoption has surged in recent years, with developers now using ***between 10 and 15 APIs*** in each application they build, on average. Given the proliferation of APIs, it's pretty understandable why attackers are increasingly focusing on APIs as a vector for breaching applications and data.

Major API Security Risks According to OWASP

API security is made more complex by the fact that there are many ways to exploit APIs. The ***OWASP API Security Top 10 list*** summarizes the main risks that organizations face related to APIs. Among the most significant are:

- **Broken Object Level Authorization (BOLA):** BOLA leads to improper validation of requests to APIs, allowing users to do things like download data that they should not be able to access.
- **Broken User Authentication:** Even if proper object level authorization is in place, attackers can potentially still gain unauthorized access to resources via APIs by defeating the authentication mechanisms (like tokens) that are supposed to restrict access.
- **Excessive Data Exposure:** When you expose more data than is necessary via an API, you invite attack. Ideally, APIs should only expose the data they need to expose in order to serve their intended purpose. Your API shouldn't be an open door to all of the data inside your environment.

These, at least, are the top three API security risks according to OWASP. Check out the ***complete list*** for the full scope and depth of OWASP's recommendations on API security.

Beyond OWASP: Additional API Security Best Practices

The OWASP list is a great foundation for securing APIs. Arguably, however, API security strategies should include additional measures, including requiring access control and avoiding API sprawl.

Require Access Control
One major type of risk that the OWASP list doesn't address (at least not directly) is total failure to require any kind of authorization or authentication for APIs. If you create a public API – meaning an API that can be accessed by anyone that knows where it is hosted – and don't require authentication, you get into situations where attackers can scrape vast amounts of data in ways you didn't intend.

Avoid API Sprawl
We've said it before, and we'll say it again: A major API security risk is using so many APIs that you simply can't keep track of where and how they're integrated into your applications. When you do this, you end up with API sprawl.

API sprawl is bad because it makes it hard to determine the scope of a known API security issue. It also makes it difficult to ensure that all APIs are being used securely.

To avoid API sprawl, establish governance policies that dictate when and under which terms developers may use both internal and external APIs within your organization.

Section Summary

We could go on in discussing major API security risks. This is a somewhat subjective topic, and although security recommendations like those from OWASP provide a great starting point for API security, it's critical to think as holistically as possible about where API security risks lie within your organization and how to manage them. Sometimes, the most serious security risks arise from very simple mistakes – like failing to keep track of which APIs you are using, or choosing not to require authentication of any type for API requests.

Addressing API Security Challenges

Now that we know all about why API security risks exist and which forms they take, let's talk about solutions. This section discusses actionable strategies for staying ahead of API security challenges, as well as minimizing your risk of experiencing API security vulnerabilities in the first place.

Reasons Why APIs Are Hard to Test and Secure

APIs come in all shapes and sizes. The most critical parts of an API are often the most susceptible to attacks on vulnerabilities like those listed in the OWASP API Security Top 10. For example, attackers might attempt to abuse paths for logged in users, exfiltrate sensitive data by fuzzy testing the endpoints, or force the site down using DDoS attacks. As the API surface increases, of course, so does the risk of exposure. Imagine having to protect APIs with hundreds of endpoints, each with its own conventions.

This could happen quite easily when there are multiple teams working on the same project and contributing to the same API. The API might not have a sole owner; instead, it might be exchanged between multiple stakeholders (such as project owners, developers, testers, operations, and network and security teams), with each group submitting their own piece to deliver new features to their customers.

Relying too heavily on some parts of the delivery pipeline to include sufficient security controls is equally problematic. For example, you shouldn't rely on the development team to constantly provide the most secure software all the time. Developers do not habitually think like attackers. They will incorporate basic pragmatic reasoning and accept logical trade-offs when delivering sprint goals. However, they will miss important security considerations fairly often simply because they are generally unaware of use and abuse scenarios. Likewise, if you expect your WAFs to perpetually block all unwanted traffic and your static code analyzers to always pinpoint all security flaws in the code, then you will be in for a big surprise when attackers exploit unknown vulnerabilities or zero-day bugs.

Therefore, since APIs are hard to test and secure, you must be innovative in finding conventional tools, agile methodologies, and various approaches that work well together. Let's explore some of them.

Tools and Approaches That Work Well Together – and Some That Don't

Once you have established that you need to use flexible approaches to keep your APIs secure, you want to make sure that you are following the right guidance. There are lots of great guides and best practices for API security, but it's important to differentiate the ones that make the most sense for your APIs from the ones that don't.

Some of the approaches that work with API security are mainly focused on establishing in-depth defenses:

- **Authenticate and authorize by default:** When you are developing APIs, you should explicitly mark unauthenticated endpoints (and not the other way around). For example, you should protect all endpoints with strong authentication (2-factor) and assign a default role that cannot read or modify any data. That way, there is less risk of exposing new endpoints that do not have any sort of protections.
- **Use API security scanners:** Tools and services in this category include runtime protection, DAST (vulnerability scanning tools), static code analyzers, and security bots. These tools provide a nice layer of defense against baseline security issues.
- **Use penetration testing:** You can hire professionals who think like attackers and are able to perform sophisticated tampering. These professionals have access to unconventional tools and leverage techniques that help expose issues that security scanners won't pick up by default.

On the other hand, some approaches might not work in the long run because they possess inherent risks. For example:

- **Security through obscurity:** This means that we try to secure part of the API by making it harder to use or to discover its endpoints or behavior. This is like an Easter egg hunt where you expose certain parts of the API only to clients that know where to look for them. Or you might make it harder for attackers to guess some of the API schemes by using special query parameters or headers. Although this approach might work for certain cases, it is not considered 100% secure. Cunning attackers might figure out a way to uncover those hidden parts of the API or infer how the API response works, which would give them the ability to retrieve sensitive information.
- **Using JSON Web Tokens (JWTs) for storing sensitive data:** JWTs offer a good balance between security and convenience when working with APIs – as long as you abide by the rules. If you store sensitive data in the JWT payload, for example, you are already compromising security since JWTs can be decoded easily. Always use industry-standard JWT libraries, strong JWT secrets, and short expiration tokens. In addition, always use HTTPS.

We mentioned before that APIs can be secured by automating the scanning of application code. Let's explore that in more detail.

How to Effectively Scan APIs

Start with the source of truth, the source code. By scanning the source code, you get a complete view of what APIs are inside of a project. This also allows developers to easily and quickly fix any security issue that has been identified by this static analysis. Since APIs need to be human-readable, scanning the API Documentation as Swagger can also be key to identifying risk—checking key risks like access control, configurations, and best practices. This is an example of ***shift-left security,*** wherein DevOps teams ensure that security is built into application development rather than added on later.

Next, once those APIs are moved to an environment, it is easy to perform real E2E tests based on predefined rules against your real API. If the scanner finds any defects or suspicious red flags, it will report them to a dashboard for triaging. These scanners include many checks (like the OWASP API Security Top 10 and open CVEs) and can automatically create policies without intervention.

Even if you have a perfectly valid specification file, though, it doesn't mean that the scanners will find all of the issues. It's equally important that you are able to feed the scanner raw HTTP recorded sessions using either Fiddler or Burp so that it can verify unknown parts of the API.

Ways to Address the Continuous Development of APIs and Ever-Changing Contracts

At the end of the day, how can you address the continuous development of APIs and ever-changing contracts when you have to contend with multiple attack vectors in real time?

There are several options to consider:

- **Scan early and often:** Software is developed at the speed of light nowadays. The only way to keep up with the ever-changing contracts is scanning the API Documents and Source Code when they are changed by the development team. This can alert them immediately to critical vulnerabilities and risk. Allow the developer to learn better practices and secure the APIs well before they are live.
- **Create clean environments for security testing:** Sometimes you can't slow down the release of new features in the name of security. It's important that security teams create special environments where they can introduce novel security testing tools and scanners for advanced security testing. The primary idea is to conduct intelligent analytics, pinpoint hidden attack vectors, and expose vulnerabilities without affecting the production environment. Once a security issue has been found and mitigated in this special environment, the security team can patch the production systems using the current change request system.
- **Adopt DevSecOps workflows:** ***DevSecOps*** is about integrating your IT security team into the full lifecycle of your app. Put simply, this means that security teams follow short and frequent development cycles, integrate security tools and vulnerability scanners with minimal intervention, ensure that all operational technologies run with the optimal security configurations, and promote a security-first mindset across isolated teams. This can be accomplished by including automated security checks throughout the CI/CD process and creating service templates that are secure by default for the development teams to adopt.

Section Summary

Let us conclude by emphasizing the fact that there is no way your APIs can be 100% secure at all times. Instead, organizations should be constantly vigilant to scan their APIs for risks and vulnerabilities. As a security professional, it's crucial that you stay informed about the status of the API security ecosystem.

Reading security advisories like the ***ones from Checkmarx*** is a great way to acquire this knowledge. Their advisories discuss real public exposures while providing deep root cause analysis and explaining mitigation tactics. Feel free to ***sign up*** for more tutorials about API security and DevSecOps.

Conclusion

For most developers today, there's no avoiding APIs. APIs have become so widespread that it's just not practical in most cases to minimize security risks by choosing not to use APIs.

Nor is it possible to manage your risks simply by documenting APIs thoroughly or sticking to certain types of APIs. No matter which APIs you use, or how you manage information related to them, problems with authentication, data leakage and more can lead to serious security incidents.

The good news is that it is possible to catch API-related security risks before they turn into breaches. Start by automatically scanning your applications to determine which APIs you are using, which helps prevent API sprawl. You should also scan your APIs themselves for known security issues or vulnerabilities. Adhering to DevSecOps best practices helps, too, as does designing APIs in ways that minimize the risk of sensitive data exposure.

When you do these things, you get to enjoy the many benefits that APIs offer, without the security headaches they can pose.

About Checkmarx

Checkmarx is constantly pushing the boundaries of Application Security Testing to make security seamless and simple for the world's developers while giving CISOs the confidence and control they need. As the AppSec testing leader, we provide the industry's most comprehensive solutions, giving development and security teams unparalleled accuracy, coverage, visibility, and guidance to reduce risk across all components of modern software – including proprietary code, open source, APIs, and Infrastructure as code. Over 1,675 customers, including 45% of the Fortune 50, trust our security technology, expert research, and global services to securely optimize development at speed and scale. For more information, visit our website, check out our blog, or follow us on LinkedIn.

Checkmarx

Checkmarx at a Glance

1,675+
Customers in 70 countries

750
Employees in 25 countries

45%
of the Fortune 50 are customers

30+
Languages & frameworks

500k+
KIKS downloads in 2021

The world runs on code. We secure it.

About Cert2Connect

Expert partner for Checkmarx Application Security in The Netherlands. Founded in 2012 by experienced professionals in the IT and cyber security industry. Our clients include both large companies and SME organizations in various market sectors, including banks, financial institutions, telecom, manufacturing, services, logistics, education, government, software development, providers and others.

Our vision: “Recognize and acknowledge risks. Mitigate them and stay in control with state-of-the-art cyber and cloud security solutions.”

Cert2Connect has among others a strong focus on application security. Not surprising when you consider that many, if not most, weaknesses arise in application development. This focus is reflected in much of our support for companies and organizations. We have a complete portfolio for the security of applications.

For more info visit our website www.cert2connect.com or contact us via info@cert2connect.com

CERT2CONNECT
CYBER & CLOUD SECURITY