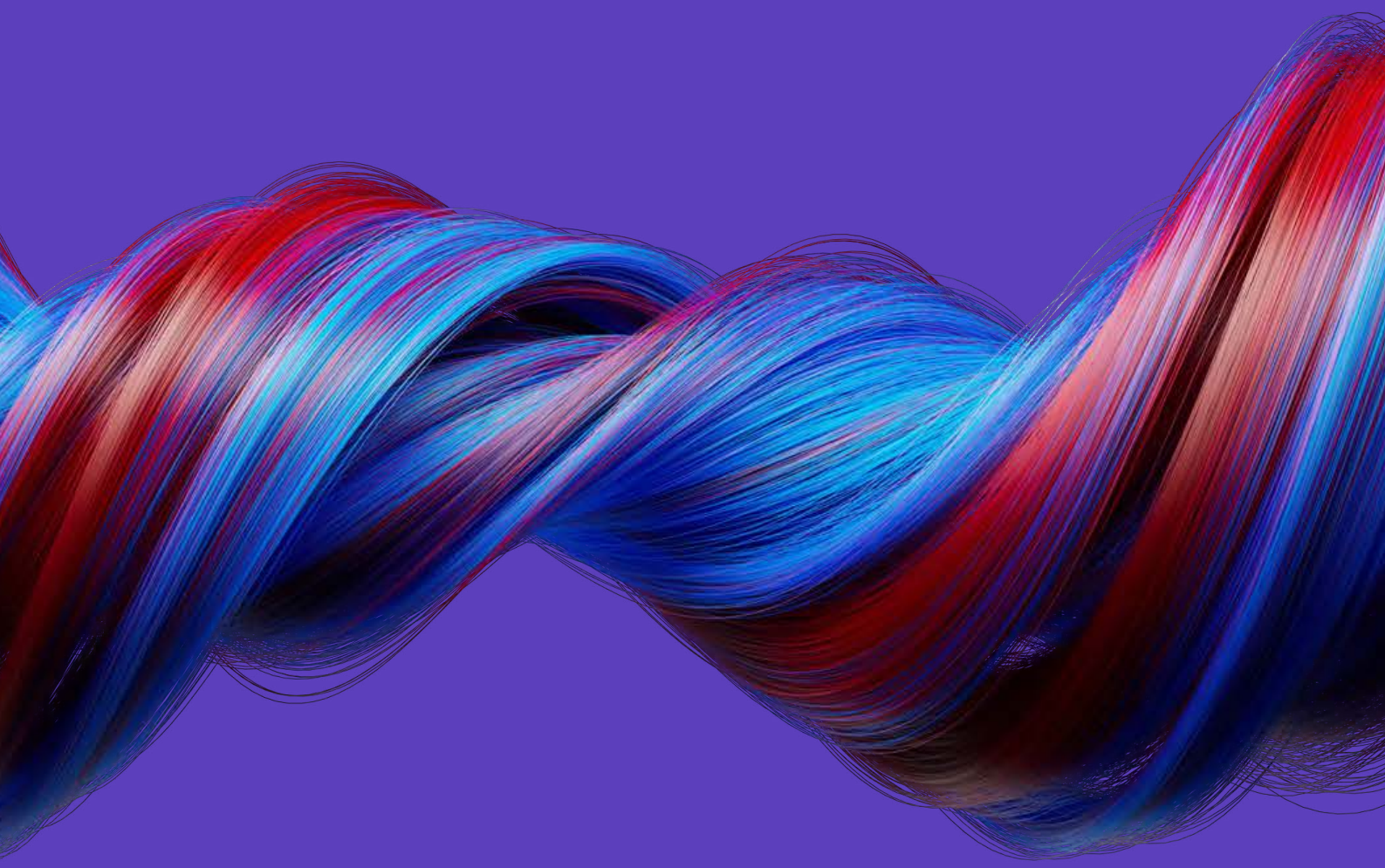
invicti
AppSec with Zero Noise

API Vulnerability Testing in the Real World

Best practices for building API security testing into your SDLC



WHITE PAPER

Table of contents

Under the radar: APIs everywhere

APIs (application programming interfaces) have progressed from being add-ons to a core application to become the fundamental building blocks of modern software architecture. Web applications made up of hundreds of microservices rely on APIs calls to exchange data and execute business functionality. On the one hand, this makes it possible for independent teams to rapidly develop components in parallel, stepping up the pace of software innovation. On the other, it exposes application internals to the entire world, making thorough security testing more important than ever.

A single carefully crafted API request can directly yield valuable data and is less prone to detection than, say, manual login attempts, so cybercriminals are now routinely including APIs in their scope of reconnaissance and attack.

Unless centrally managed and properly tested, undocumented API endpoints can make it into production, quietly increasing the overall attack surface.

Documented or not, each API endpoint also needs to be tested for vulnerabilities just like any other part of the application, so it doesn't become a security blind spot.

In the real world, finding a way to discover assets and perform accurate web application security testing across the entire attack surface to cover both UI and API is definitely a non-trivial task. This Invicti white paper shows the practical challenges of API vulnerability testing, technical solutions to overcome them, and best practices to make it all work in a modern web development pipeline.

Highlights from this white paper include:

- ✓ Why API vulnerability testing in modern development pipelines runs into so many challenges
- ✓ How advances in AppSec technologies have made it possible to discover and automatically test APIs along with the rest of your expanding web attack surface
- ✓ What best practices you can follow to make API security testing a routine and efficient part of your secure software development lifecycle (SDLC)

The API conundrum: Challenges in securing APIs

In a classic case of only searching under the streetlight, surprisingly many organizations have been overlooking or downplaying the API part of the web application attack surface when planning and executing their AppSec programs. While most companies are aware of the need to discover and scan APIs, there are some very real practical challenges to be overcome on the way to making API security a routine part of web application security. Under pressure to add API testing to an ever-growing security backlog, AppSec professionals often struggle to find the right tools and workflows for the job. This chapter outlines some of these obstacles and peels back the layers of a complex API conundrum.

Terminology: Access to APIs vs. access via APIs

Any discussion of API security should start with a clear definition to avoid confusion. Application programming interfaces are exactly what the name implies: interfaces for programmatically interacting with an application. This means that there are two separate levels of security involved:

- ✓ **Securing access to the interface itself:** All requests incoming to a private API must be suitably authorized in order to be passed on to the application. Attackers focused on this layer of security will attempt to break or bypass authorization to obtain access to the API and send requests or attacks to the application.
- ✓ **Securing access to the underlying application:** At its core, an API is just another channel for interacting with the application that needs to be included in application security testing. At this level, API security means securing the application against web attacks that arrive via the API (alongside similar attacks that originate from the user interface or other channels).

This paper focuses primarily on the latter aspect of API security, showing how to incorporate API discovery and vulnerability testing into a broader web application security program.

Before you begin: Getting a handle on API vulnerability testing

Depending on the current toolset and workflow, being told to also test APIs from now on can lead to some serious head-scratching. An interface is, by definition, an abstract and predefined way to access some underlying application, service, or system. That makes it hard to check for vulnerabilities at code level because behind the uniform API layer, your application might use any number of programming languages and technologies to get data to and from the interface. Some endpoints might make other API calls, or you could even have an API on top of another API as a compatibility layer.

All this makes dynamic security testing a practical necessity so you can test APIs like the black boxes they are. And while you can (and should) perform periodic penetration testing on your APIs just like any other part of your web application environment,

> **the sheer scale, complexity, and speed of modern web development calls for an automated vulnerability scanner for routine testing.**

But bolting an additional tool onto your AppSec setup can mean extra work on integration and then even more work to manage yet another source of security reports. And this is assuming you can find a quality scanner that will flag real vulnerabilities without flooding you with false positives.

APIs are just another way to interact with a web application, so they should be tested for vulnerabilities along with the rest of the application, ideally using the same tools and processes. That way, you can make your testing and remediation workflows simpler instead of more complicated—but to do this in practice, you need to negotiate quite a few hurdles.

Challenge #1: Taming the noise

Application security teams are all too familiar with overwhelming numbers. A large organization may have hundreds if not thousands of developers but only a handful of security engineers. In smaller organizations, it is not uncommon to have a security team of one. With the cybersecurity skills gap still looming large, the odds seem unlikely to improve, and throwing in API discovery and scanning adds yet another work multiplier. Worse still, this catch-all term hides additional practical complexity.

There are a few main types of APIs used on the web. While REST is by far the most popular API architecture, more complex or older business systems may use SOAP, while cutting-edge applications dealing with big data are likely to rely on GraphQL. So before choosing a type-specific tool, you would need to poll all the development teams about the API types they currently build, maintain, or plan to add in the future. Assuming you get this information and it is accurate, you may need to support more than one API type—and that could mean several different tools to acquire, integrate, and manage.

The next multiple-choice question is about API definition formats. Across the main API types, there are at least a dozen popular definition formats (starting with Postman, OpenAPI a.k.a. Swagger, WADL, and WSDL) and several more proxy export formats that you might need when you don't have full definitions. Which ones do you need? Which don't you need? So you have more polling, planning, and—again—potentially multiple tools to run and maintain for different formats.

And this is all before looking at the number of API endpoints to add to your vulnerability testing workflows and the number of additional scan results to process. Even adding only a moderate-sized external REST API can mean several dozen extra URLs to scan—but what about service-oriented architectures? If you have an application made up of a hundred microservices, you have a hundred more service APIs to scan. And what if you have a dozen such applications? And each API yields several scan results?

Without a careful and centralized approach to API discovery and API vulnerability scanning, the multipliers will keep piling up, threatening to overwhelm already overstretched security resources. To take control, you need to simplify and reduce, not multiply and complicate.

Challenge #2: Knowing what to test

When doing API security testing, the fundamental difference compared to web application testing is that you can't simply fire up a crawler and directly crawl the whole API like you would a web page. The only way to be sure of covering an entire API is to have its definition—and that usually means getting the current definitions from your developers (and hoping they are accurate). Because APIs can change rapidly, you would ideally need to do this before every security test. In any sizable development organization, doing this manually on a regular basis would be a major headache for the security team and an extra chore for the developers.

Even assuming you can spare the resources and time to collect definitions manually, the complexity issue rears its head again as soon as you start looking at specific definition files.

Unless you have standardized API policies in your organization, you could end up with a dozen different formats for various architectures and API platforms, and that typically means multiple tools for importing and testing. And keep in mind that it's unlikely for every API in your environment to be known and documented—not when it's so easy to create a "temporary" API endpoint that later slips into production without official testing or documentation.

If you want to avoid security weak spots, having up-to-date definitions for all your APIs is a must. However, very few organizations can claim they fully document all their APIs, so to fill the inevitable gaps between what you know and what you're actually running, API discovery is another crucial piece of the puzzle. Whether it's through network traffic analysis, integration with API management systems, inferring API endpoints based on crawling data, or a combination of these and other approaches, security testing tools need to bridge the gap between specification and reality. Considering the rapid pace of development and frequent release cycles, plus the potential disconnect between changes to back-end APIs and front-end web and mobile applications, all this needs to be done automatically in a continuous process to maximize coverage.

Challenge #3: Getting broad coverage and accurate results

Authentication is now a vital requirement for web security testing, especially with business applications that serve very little data or functionality to unauthenticated users. This goes doubly for APIs, where unauthenticated vulnerability scanning would be more or less pointless. But many vulnerability scanners have, historically speaking, struggled with automated authentication even on user-accessible sites, leaving some users doubtful whether thoroughly scanning an API for vulnerabilities is even possible.

Compounding these doubts are lingering misconceptions about the accuracy of automatic vulnerability scanning in general, rooted in a deep mistrust of legacy scanners designed in the days of static websites. Faced with modern JavaScript-driven web applications such as SPAs, where a dynamically-generated front-end communicates with a service-oriented back-end via APIs, legacy tools can miss vast swathes of the application attack surface. When you add authentication and authorization along with modern enterprise must-haves such as single sign-on or multi-factor authentication, a scanner that can't authenticate and then run a thorough test is worse than no scanner at all because it gives you a false sense of security while doing very little useful work.

In the same way that keeping up with the latest web technologies and application architectures requires non-stop research and development, keeping up with web vulnerabilities and adding security checks as soon as new attacks are discovered is also a full-time job.

Creating, maintaining, and running automated security tests that balance performance and accuracy is already hard enough for interactive websites and applications. When you add the complexity and opacity of APIs, you are raising the bar to a level where very few tools make the grade—and that's even before considering the practicalities of making it all work in your existing development routine.

Challenge #4: Keeping up with the pipeline

Let's assume for a moment that you've successfully negotiated all these hurdles and found a way of running vulnerability scans on your APIs on a regular basis. Depending on the size of your application environment and the maturity of your AppSec program, this could mean hundreds of vulnerability reports arriving with every scan. Now what? Simply running a scan does nothing to improve security. To make your applications more secure, you need to identify and remediate real vulnerabilities before they make it into production—and there's a long way to go from scan results to fixes.

Automation is the name of the game in web development, especially with agile methodologies and, more recently, the growing use of AI assistants to pump out ever more code, yet security testing still tends to get manual treatment. This is mostly due to a lingering lack of confidence in the accuracy and integration capabilities of automated scanners, as well as organizations still putting a premium on functionality and time-to-market rather than security. To avoid flooding developers with false positives, overworked security teams often get burdened with double-checking scan results and then manually assigning only actionable issues to developers.

When dealing with APIs, you are upping the ante by throwing in even bigger numbers. Handling all those additional security checks is not a problem for any serious vulnerability scanner, provided, of course, that it can access all the test targets. But if you extend application security testing to include your APIs and, overnight, find yourself facing double or triple the number of vulnerability reports you had before, you'd better be sure they don't contain false positives—or you won't be able to feed them into the development pipeline for remediation.

Speaking of the development pipeline, integrating security testing with existing toolchains is another common headache. Without purpose-built integrations with industry-standard issue trackers, CI/CD platforms, and web application firewalls (WAFs), organizations are forced to roll their own solutions by gluing together disparate tools and formats. Each new tool means another integration project and, depending on the specific products and technologies, the end result can be far from seamless, pulling developers out of their optimized workflows and introducing friction (and delays) into the highly automated pipeline.

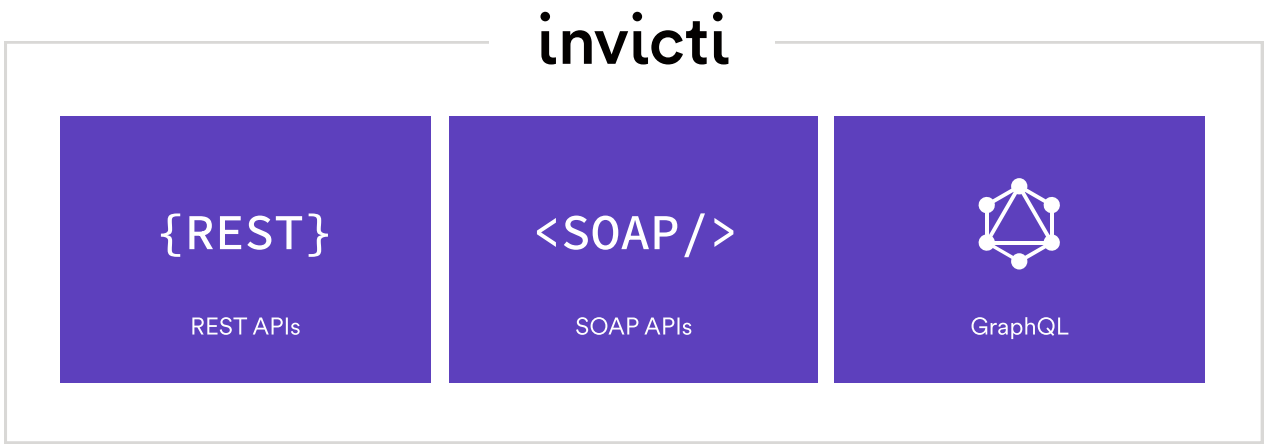
Making API vulnerability scanning a technical reality

A crucial prerequisite for addressing all these challenges and smoothly folding API security into your wider application security program is having the technical ability to make it all work together. With API endpoints adding yet more moving pieces to the already complex patchwork of web development, starting from the outside and working your way in is the surest way to maximize security testing coverage regardless of underlying complexity—and that means dynamic application security testing, or DAST.

Built on mature and proven vulnerability scanning technologies, Invicti offers a pragmatic DAST-first approach to web application and API security, uniquely offering both discovery and security testing within a single platform that spans applications and APIs. This chapter outlines the industry-leading technical capabilities that underpin Invicti's solution to the API security conundrum.

Cover all major API types and definition formats in testing

Adopting a unified approach to API vulnerability testing starts with knowing what API types are used in your web application environments, listing the API endpoints to be tested, and having the technical means to test them. REST APIs are by far the most common type of interface in modern web applications, especially for lightweight microservice communications. Then you have SOAP APIs that are still used in many financial systems and other enterprise applications that require precise interface and data format definitions. And finally, we have GraphQL—a relatively young API type that is rapidly gaining popularity, especially in big data applications. Invicti has all three major API types covered out-of-the-box, with built-in dedicated security checks and support for various ways of importing and discovering API definitions.



Import definitions and schemas

The sheer number of different API definition formats used to be a major obstacle for centralizing API security testing, often requiring multiple tools and complicating the process. Invicti comes with built-in support for 16 different formats, including Postman, OpenAPI (Swagger), WADL, WSDL, and more. These include both actual API specification formats and other popular API definition sources, such as project files and technology-agnostic CSV exports. For GraphQL, where you are typically working with a single endpoint, you can import your data schema file or provide an introspection URL (if the introspection feature is enabled) to allow Invicti to learn the query structure automatically.

For several of the definition formats, you also have the option of supplying the definition as a URL rather than a file. This is extremely useful for centralizing and automating security testing, as you can then always load the current API definition from a predefined location and be sure that every scan you launch uses the latest version. In fact, with additional scripting, it is even possible to automatically launch a scan whenever the API definition is updated to maintain continuous security.

Supported API definition formats

ASP.NET Project File	Burp Saved Items	Comma Separated Values (CSV)
Fiddler	HTTP Archive	I/O Docs
Invicti Session File	OWASP ZAP (formerly Paros)	Postman
RAML	Open API (formerly Swagger)	Web Application Description Language (WADL)
Web Service Definition Language (WSDL)	WordPress REST API	GraphQL Schema/ Introspection
gRPC		

Discover additional API endpoints

Having the definitions is crucial because you can't directly crawl an API as you would a web application—you need to know the endpoints and request formats. In the real world, though, there will always be some undocumented APIs that you still need to find and test. To help with this, Invicti provides multi-level API discovery spanning zero-config specification file detection, API management platform integration, and network traffic analysis in container environments. Discovered endpoints are added to your inventory and can be tested as if you had the specs from the very beginning.

The Invicti scanner also automatically imports any supported API definition files that it finds when crawling an app, as well as examining the structure of the URLs it encounters. For any crawled URL that looks like an API call, it will attempt to infer the API endpoint and test it. This includes heuristic URL rewriting to discover underlying request parameters and probe them for weaknesses just like an attacker would.

When dealing with undocumented APIs, the standard approach to testing is to set up a proxy to monitor traffic to and from the API and thus discover the endpoints and request specifications. This is especially useful for testing back-end APIs, including mobile application back-ends, or if API documentation is known to be incomplete. Invicti supports several popular proxy export formats, including Fiddler and Burp, so you can use proxy sessions as a data source for your definitions. Invicti Standard also comes with its own internal proxy, so you can run it in a local environment to record API traffic for later use in testing.

Authenticate automatically

Automated authentication is a practical prerequisite for API vulnerability scanning, as the scanner has to obtain API access before it can test the underlying application. Invicti provides mature support for popular API authentication methods, including basic HTTP authentication, JSON Web Tokens (JWTs), and OAuth2. With OAuth support, you get the ability to scan in single sign-on environments, which can be problematic for less advanced scanners.

As APIs increasingly come under attack, additional safeguards have been developed to hinder unauthorized access, including anti-CSRF tokens and message authentication codes (HMACs). These provide another stumbling block for less mature solutions, but with Invicti's pre-request scripting feature, you can easily customize scanner requests to include such additional codes and tokens. This can be done manually or, for Postman API exports that include pre-request scripts, Invicti can automatically import the scripts along with the API definitions. That way, you can test all your APIs for vulnerabilities with full authentication without resorting to clumsy and risky workarounds such as disabling safeguards just to get a scan to run.

Test for vulnerabilities with consistent accuracy

With authentication set up and test targets discovered, Invicti can use its full battery of security checks to safely probe the entire application attack surface for vulnerabilities, covering both UIs and APIs. Proof-based scanning technology is used to automatically confirm the exploitability of the vast majority of direct-impact vulnerabilities with no risk of false positives. Armed with trustworthy and actionable results, you have reliable data to automate application security testing at all stages of the development lifecycle and routinely address security issues without causing workflow bottlenecks.

Being able to trust vulnerability scan results and act on them directly without fear of false alarms and tedious exchanges with other teams is still a rare experience for organizations. Modern agile and DevOps workflows rely on automating everything you possibly can during development and testing, but developers are wary of security testing tools flooding their issue trackers with spurious warnings. To get vulnerability scan results that you can process directly in an automated and integrated process, you need a solution built on years of security research, development, and refinement, with security checks that are optimized to balance accuracy, performance, and scan safety.

Streamline AppSec with one central platform

By treating APIs as an integral part of the overall web attack surface and discovering and scanning them for vulnerabilities using the same centralized platform as your applications, the Invicti approach simplifies the AppSec program instead of adding yet more moving pieces. Even as web application security becomes a top priority, organizations are struggling to cover their entire environment and get quick and measurable security improvements from disjointed tools and initiatives. Plugging a separate process for API security into an already complex toolchain can result in even more delays between testing and actual security improvements.

Invicti brings a holistic view of application security with a DAST-based platform that enables you to discover, test, and secure APIs as an integral part of your overall attack surface. This can reduce the number of dedicated scanning tools in your workflows and gets your security engineers and developers working on actionable security issues that make a difference. Having DAST as the foundation means you can maintain overall visibility but also extend it to the back-end by enabling interactive application security testing (IAST) and software composition analysis (SCA) features within the same unified platform. That way, you get a centralized view of your web security posture across APIs, websites, and web apps, across multiple testing methodologies, and across all stages of your software development lifecycle.

[1]For API vulnerability scanning, it may not always be possible to exfiltrate complete proof via the API and technically mark the issue as automatically confirmed, but the results are just as reliable.

Best practices for building API security into your AppSec program

To be effective without burdening your already overworked security and development teams, API security needs to be a routine yet unobtrusive part of your overall web application security program—and that means tight integration with the SDLC. Because APIs are defined and modified in development, you need to build automated API security testing into the development pipeline to be sure you're catching all changes, no matter how frequent they are.

Based on hundreds of customer stories and over a decade of supporting organizations in deploying application security testing, we've put together a collection of best practices for integrating API discovery and vulnerability scanning into your wider AppSec program with Invicti.

Keep tabs on your API definitions

Rather than chasing after API specifications and docs every time you need to run a scan, plan to spend a little time upfront with your development leaders to **define a centralized and automated process to keep track of all API definitions in your organization**. In practice, this could be as simple as adding a script to the build chain to ensure that all API definition changes are reflected at predefined locations. You can then use Invicti's import from URL or import from file features to get the latest endpoints and feed them to the scanner (potentially even using Invicti's own internal API calls to automate the process). That way, you are always testing the full scope of your current APIs completely automatically to maintain continuous security.

Build API discovery into your application security process

However good your API inventory practices, you should always assume there are still endpoints outside your visibility. Whether manual or automated, **API discovery should be a routine part of your broader application security process** to ensure that all your sources of information about APIs are helping you ensure nothing goes untested. Invicti provides multi-layered discovery for REST APIs, combining zero-config specification discovery with API management platform integrations and network traffic analysis in Kubernetes environments. Whether loaded from known definitions, entered manually, or identified through discovery, all the API endpoints can be selected as vulnerability scan targets in Invicti and scanned for vulnerabilities, with their vulnerability and remediation status tracked using the built-in vulnerability management functionality.

Integrate API security testing into existing collaboration tools

Developers work best when they have clear and actionable tickets in their issue tracker. By integrating API vulnerability scanning into existing development toolchains, you can automatically create issues based on predefined severities, allowing your developers to **handle and remediate security defects like any other type of bug**. With Invicti's proof-based scanning and issue tracker integrations, you can safely define rules to automate only confirmed vulnerability reports and send them directly to the developers, complete with remediation guidance. Repeat scans can also be incremental for fast results. And once an issue is marked as fixed, Invicti can automatically retest the specific web asset to verify that the vulnerability is truly resolved.

Plan for solid results and value from day one

Rather than treating API security as yet another thing to bolt onto your development and security toolchains, think about it as one part of a wider AppSec picture. Look for ways to get real security value without lengthy deployments and without burdening your teams with clunky external tooling or avoidable manual work. For Invicti, this means taking a holistic view of web application security and **treating APIs as one more attack surface to continuously cover with discovery and accurate vulnerability scanning**.

Test all API endpoints all across the SDLC

Instead of relying purely on manual API security testing in later stages of the application lifecycle, you can take an integrated DAST-based approach to trigger automatic API vulnerability scans at multiple stages of the pipeline. This lets you **probe the entire exploitable attack surface of the application as executed at each phase of development**, up to and including production. By continuously covering both UIs and APIs with accurate and fully automated vulnerability scanning, you can set a security baseline and maintain it regardless of frequent software updates or changes to the threat landscape. Under this consistent DAST umbrella augmented with API discovery, you can then add other application security processes as required for your business environment and AppSec program, with the flexibility to mix and match manual and automated methods for API security testing.

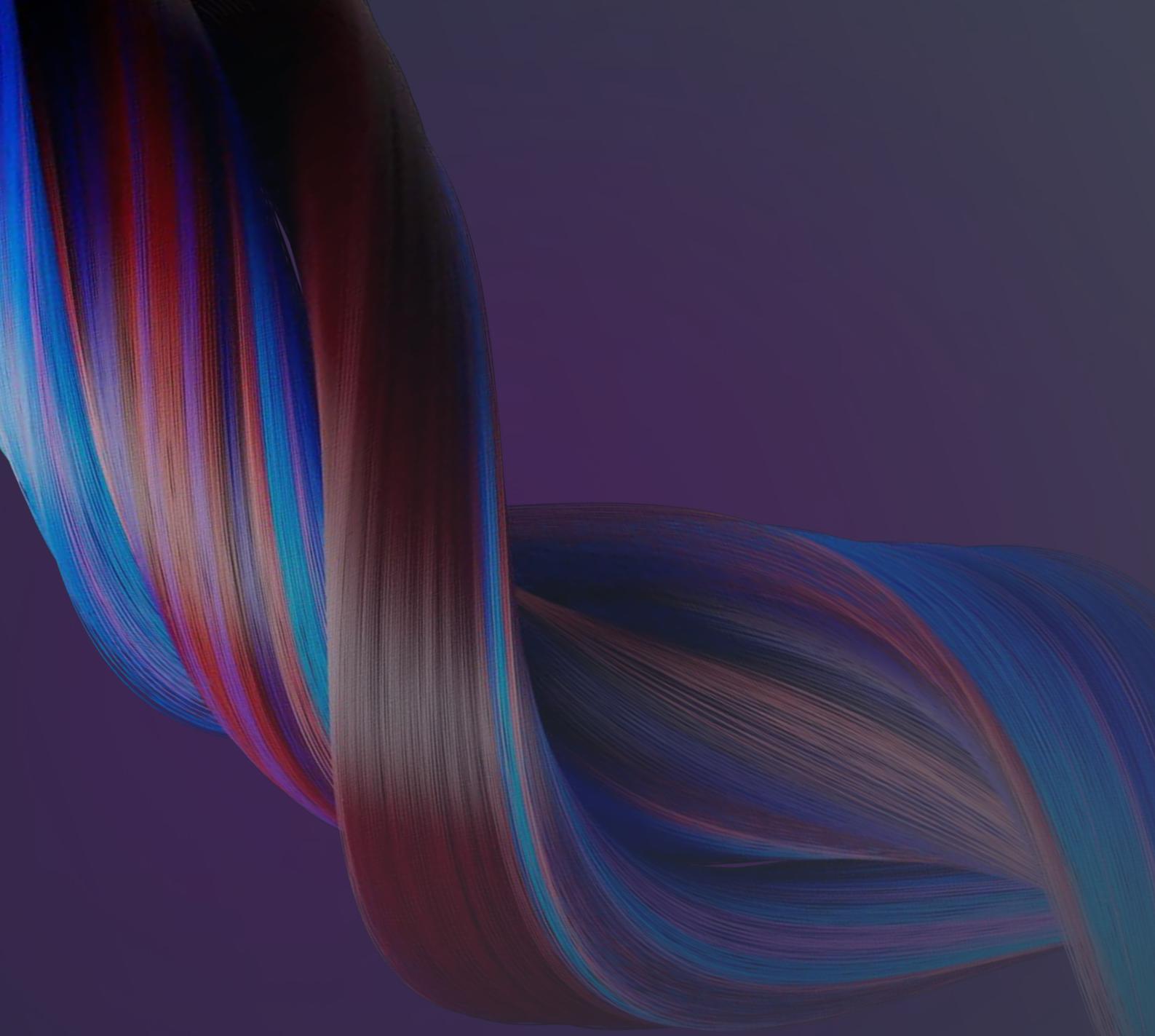
Getting the initial authentication and testing setup just right makes all the difference when testing web applications—and especially the APIs they rely on. To make sure you can quickly get measurable security improvements even in complex environments, Invicti offers step-by-step onboarding and guided success services.

Simplify and centralize AppSec to take away API security pains

Application programming interfaces are vital gateways to modern web applications and data back-ends, so application security testing cannot be complete without also covering APIs. At the same time, organizations are increasingly coming to the realization that effective and efficient web application security must be a routine part of the SDLC. Building API security testing into development workflows is the logical next step—and at Invicti, we've coupled it with API discovery for a centralized and automated AppSec platform that does what it claims to do.

The only practical way to ensure continuous API security despite the growing complexity and opacity of application environments is to simplify and centralize all application security testing—including APIs.

invicti
AppSec with Zero Noise



invicti

Invicti Security—which acquired and combined AppSec leaders Acunetix and Netsparker—is on a mission: application security with zero noise. An AppSec leader for more than 15 years, Invicti delivers continuous web application and API security, designed to be both reliable for security and practical for development while serving critical compliance requirements. Customers choose the Invicti platform to leverage DAST, SCA, and IAST solutions to better secure their environments and ultimately reduce risk across their web applications and APIs. Invicti operates globally with employees in over 11 countries and serves more than 4,000 customer organizations. For more information, visit www.invicti.com or follow us on LinkedIn.