invicti
AppSec with Zero Noise

Cover Your APIs

Securing Your Hidden Web Attack Surface

White Paper

Executive summary

Web application programming interfaces (APIs) present a huge – yet still underprotected – attack surface for cybercriminals. While application security testing has traditionally been focused on the user interface, modern applications rely heavily on APIs for data exchange and to build the application architecture. Today's enterprise web apps are no longer monolithic applications but rather constellations of loosely coupled services communicating through APIs, all subject to rapid development in response to shifting business requirements – and each a target in its own right. **While these innovation pathways are crucial for growth, they also dramatically increase the web attack surface, leaving organizations exposed to attacks that target APIs in order to directly access sensitive data and functionality.**

As Gartner® estimates, "By 2023, 90% of web-enabled applications will have more surface area for attack in the form of exposed APIs rather than the user interface, up from 50% in 2020." It is therefore quite clear that application security testing must also cover APIs. However, legacy AppSec approaches haven't kept pace with technical developments, leaving organizations struggling to reconcile multiple testing tools and processes – or altogether overlooking web application APIs in their security programs. With API abuses expected to become the top threat vector in 2022, continued neglect can only mean yet more successful cyberattacks.

This white paper shows the importance of including APIs in web application security testing and outlines a modern approach to vulnerability testing that covers the entire attack surface of modern web applications, from development to production.

Highlights from this white paper include:

- The importance of web APIs and web services in software development
- The challenges of including APIs in application security testing, and how to overcome them
- The security and efficiency benefits of holistic AppSec

What you don’t see can hurt you most

At the risk of revisiting an industry-standard analogy, APIs are a major part of the hidden underwater part of the iceberg that is your web application environment. Just as 80% of an iceberg lurks under the surface, the vast majority of modern applications work with APIs more than their user interface. In fact, research shows that APIs account for a massive **83% of all web traffic**, in no small part because they are what powers content delivery networks and data back-ends.

We are used to thinking that the user interface and the application are the same thing, but the visible controls are only the start of the att ack surface available to threat actors. In the physical world, criminals are unlikely to march in through the front office if they can sneak in through the service entrance. Cybercriminals are no diff erent – why waste time trying to hack a login form when they can quietly extract the same data through an API? More importantly, APIs are far more likely to slip under the radar during testing and asset inventory, which makes them harder to secure and easier to exploit.

As we will see later, a graphical user interface is increasingly just a way to retrieve data from APIs and present it to users. In modern service-oriented architectures, most – if not all – application functionality is implemented as web services and exposed through APIs. In effect, APIs are the gatekeepers of the world’s business logic and data – and with information now the most valuable resource, it is no wonder that API-based attacks are on the rise.

If you want to avoid becoming the next breach headline, there is no question that you need to test and secure your web applications in their entirety, including both UI and API – because cybercriminals will find the weakest spot and strike there. But while web application security testing is already a fairly mature segment, API security is still playing catch-up, and there are many misconceptions and misunderstandings to clear up around testing the API part of the web attack surface.

> “By 2023, **90%** of web-enabled applications will have more surface area for attack in the form of exposed APIs rather than the user interface, up from **50%** in 2020. By 2022, API abuses will move from an infrequent to the most frequent attack vector, resulting in data breaches for enterprise web applications.”

Source: Gartner, Magic Quadrant™ for Application Security Testing 2021, Dale Gardner, Mark Horvath, Dionisio Zumerle, 6 January 2021. GARTNER and MAGIC QUADRANT are a registered trademark and service mark of Gartner, Inc. and/or its affiliates in the U.S. and internationally and are used herein with permission.

Why APIs make attractive targets

API creators may only be expecting well-formed requests from known systems (or even the same application), so they are less likely to check incoming calls with the same care as for user-facing pages.

Web APIs are designed with automated access in mind, which is convenient both for valid users and for attackers.

Many start life as private APIs intended for testing or internal use but later slip into production, oft en without inventory, documentation, logging, or access control.

API endpoints are easy to define but hard to find, test, and secure.

Demystifying web APIs

Part of the reason for the rising tide of API-based attacks is that many organizations are still not sure what web APIs they have, what they should do about them, and what questions to ask when investigating API security. This section explains web APIs and related concepts, showing how they work, what hides behind the acronyms, and why API security is still widely misunderstood – and underestimated as a risk vector.

Attacks on APIs vs. attacks via APIs

Anyone researching web API security will soon find that the term is used in two very different senses. One usage relates to securing access to the API itself, while the other is about securing the underlying applications and services against attacks that come through that API. Understanding the difference is crucial for selecting the right tools and methods, so here is a brief overview:

Attacks on APIs

For the vast majority of web APIs, getting access that allows you to make API calls is only possible with the proper authorization. This usually requires an access key combined with some form of automated authentication to verify that the requesting system is permitted to access the API in the first place. Attacks on APIs focus on bypassing access controls to then allow threat actors to make API calls that target the underlying applications or services.

Attacks via APIs

Once attackers have obtained API access by bypassing access controls or simply finding an unsecured endpoint, they can start probing the application in the hope of finding exploitable vulnerabilities. At this point, they can use the whole array of available web application exploits – with the added bonus that they are now using an access path designed specifically for remote and automatic use.

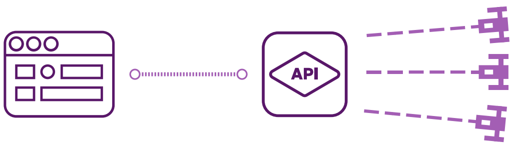

Figure 1. Attacks on APIs are about getting access to the interface itself.

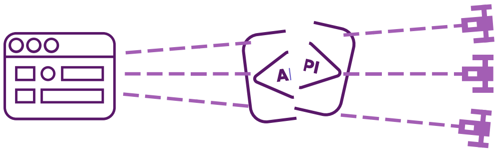

Figure 2. Attac PIs are about using API access to attack the underlying application.

Web API glossary

APPLICATION PROGRAMMING INTERFACE (API)

A connection for exposing software functionality to other systems and applications. This is a general programming concept, not limited to web development. An API defines a request format and a return format.

WEB API

An interface for accessing web-based software. There are many models and standards for defining and calling web APIs, the most widely used model being REST. Any sizable web application is likely to have an API somewhere to allow automated access to selected functionality. For example, an e-commerce platform may provide sellers with a manual form for adding single items while also exposing an API endpoint for adding items in bulk from third-party inventory management applications.

WEB SERVICE

Web-based code that is not a standalone application but only performs a specific operation. Web services are only accessible through their interfaces, so whenever you have a web service, you also have an API. While the opposite is not true (because systems and applications can also have APIs), you will often see the terms web API and web service used interchangeably.

API ENDPOINT

The URL for calling a specific web API. The endpoint is the address to which API requests and parameters are sent. What endpoints are exposed, how they are accessed, and what format the requests should be depends entirely on the API type and design. Ideally, all endpoints should be listed and documented in the API specification.

Common web API types

REST

Short for REpresentational State Transfer, REST is by far the most popular type of web API. Rather than a strictly-defined format or protocol, REST is a general style of web application architecture that sets out guidelines for designing and implementing HTTP communication between web-based systems and components. Each operation available through a RESTful API corresponds to a different endpoint (URL). A list of all the endpoints along with their parameters and data formats makes up the API specification. JSON is the most widely used data format, but XML and others are also supported.

SOAP

Originally called XML-RPC, SOAP was the first widely-used standard for web service communication. SOAP uses XML messages to exchange requests and responses. SOAP was created to be universal, extensible, and technology-agnostic, which makes it relatively difficult to use and optimize compared to HTTP-based REST communication, especially for simple scenarios. However, being a full-featured protocol with its own service description language (WSDL), it is well suited for more complex communication.

GraphQL

The newcomer in this list, GraphQL introduced the ability to query APIs in a similar way to databases. A single GraphQL endpoint receives queries, passes them to a resolver, and returns only the requested data. This allows clients to get exactly the information they need in a single request instead of assembling it across multiple calls. When the introspection feature is enabled, developers can interactively query a GraphQL endpoint to discover what data it can provide. While still relatively new, GraphQL is rapidly gaining popularity because of its usefulness for applications that work with very large data sets, and its adoption can only accelerate in the coming years.

Meet your ever-growing web attack surface

One thing is certain: if you have web applications, you already have APIs somewhere in your web attack surface today. The original use case for adding them was to extend access to application functionality beyond the graphical user interface, most notably for integrations and automation. With the move to service-oriented architectures and agile development, it became clear that APIs and the services behind them were taking center stage as one of the fundamental building blocks of modern web apps. In terms of security, the web application has gone from one big target to hundreds of little targets – and that means hundreds of potential entry points to defend.

In by the service door: APIs for extending access

An external API is essential for any web app that needs to interact or integrate with other software. In that respect, APIs are the glue that holds together the entire modern web. They make it possible to add communication between all kinds of systems, fueling innovation and unlocking new business opportunities. Mashups, from content aggregators to enterprise dashboards, are a prime example of apps that rely on content and functionality from external sources, retrieved and manipulated via APIs.

In the enterprise software world, APIs are crucial for integration and customization. For example, Invicti products provide dozens of REST API endpoints that allow organizations to automatically run vulnerability scans, fetch results, manage users, and perform many other operations. In this case, the API makes it possible to integrate vulnerability testing into existing workflows and customize out-of-the-box integrations where necessary. Crucially for security and data confidentiality, only authorized and licensed users can make such API calls.

In general, APIs require a unique access key to confirm that requests are authorized. All requests to non-public APIs also need to be authenticated, with single sign-on now being the norm for enterprise applications. If either authorization or authentication is weak or missing, threat actors may be able to send API calls to the underlying application or service to extract or modify data. Worse still, if undocumented API endpoints makeit into production, they are likely to be less secure than official endpoints, either because they were never intended to be publicly accessible or because they slipped under the testing radar.

APIs in service-based applications: Every app is a hundred apps

Modern web apps routinely use API calls to exchange information between components, especially with the move to microservices and serverless architecture. Rather than storing and executing all application code on a central server, developers implement operations as separate web services that communicate via APIs. Depending on the design choices, a single web application can be made up of dozens of services or hundreds of microservices. This API-driven approach enables agile development and rapid innovation, as teams can work in parallel to build independent components without waiting on a bigger release.

With so much functionality now deployed as granular services, web APIs can also serve as a standalone multi-purpose back-end. The same API can then deliver data and perform operations for multiple websites, business systems, mobile apps, IoT devices, and more. With back-end services doing the heavy lift ing, it becomes simpler and faster to create new applications. Again, this helps to drive innovation by focusing time-sensitive efforts on building the right front-end and business logic to get maximum benefits from existing back-end services.

For all its benefits, spreading access to data and functionality across dozens of standalone services has major security implications. Instead of having just a single central application to test and secure, organizations now also have a nebula of web services, each a target in its own right. The convenience of automated access by design is a double-edged sword, giving malicious actors much more freedom to craft and conduct attacks on a massive scale with less chance of arousing suspicion. Combined with the risk of undocumented and inadequately secured endpoints, this makes APIs a fast-growing vector for cyberatt acks with their many possible consequences, from data breaches and denial of service to ransomware deployments.

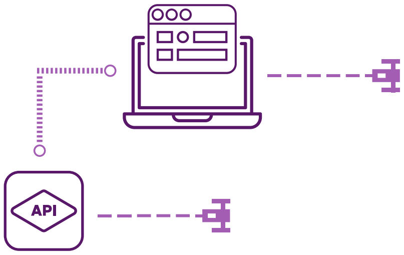

Figure 3. Even monolithic legacy applications often have an eternal API for integration and data exchange that comprises part of their attack surface.

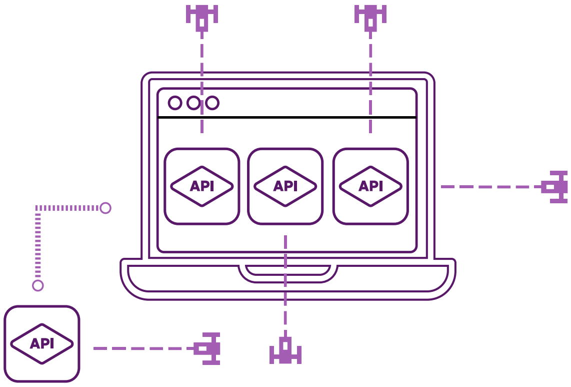

Figure 4. Modern web applications not only have external-facing APIs but are themselves made up of dozens of microservices, each with its own API – all contributing to the overall attack surface that you need to test.

Building API scanning into your AppSec program

Automated vulnerability scanning is a crucial part of any mature AppSec program, especially with sprawling web application environments that change frequently, often on a daily basis. **To know your true security posture, you need to cover your entire web attack surface, which means scanning your web assets both through UIs and via APIs – and scanning both in development and in production.** To complicate matters further, APIs can change and expand much faster than user-facing interfaces, making purely manual testing impractical and strengthening the case for efficient and accurate scanning.

Research shows that security often takes a back seat when release deadlines loom, and this goes doubly for APIs. Whether they are perceived as less exposed or are simply harder and more time-consuming to find and test, API security can slip out of mind when time is short. This is especially true when a separate toolchain is needed for the API part of security testing, further increasing the risk that time pressures will cause vulnerabilities to slip into production.

Incorporating APIs into the overall AppSec testing strategy presents a major technical and organizational challenge. To maximize test coverage without stopping development pipelines to wait for results, you need sufficiently advanced tooling and a systematic approach to application security to make it all work without having to chain together multiple tools and processes. Here is how Invicti makes this possible.

Get the API definitions so you know what to test

When scanning websites and applications, vulnerability scanners crawl pages and follow links to build up a list of URLs for testing. But you can't crawl an API – you simply have to know the URLs of the API endpoints you want to test. For this, you need API definition files that are created during development and maintained across the entire lifetime of a production application. You also need to support all the popular definition formats, know the type of API, and have accurate security checks ready to probe the application via that interface.

Invicti provides vulnerability scanning support for the most popular API types, namely REST, SOAP, and GraphQL. Users can import API definition files in industry-standard formats, including Postman, OpenAPI/Swagger, WADL, and WSDL. **In practice, this means that as long as you have the definition files and keep them updated, your automated security testing process can routinely cover both the UI and APIs during vulnerability scanning.** And if you don't yet have an application security testing process that is **integrated into your software development life cycle** (SDLC), building one is crucial to ensure that security can keep up with the pace of development.

Integrate API testing into your development lifecycle

API definitions are prepared and maintained by development teams, so integrating vulnerability testing into the development pipeline makes it far easier to include APIs in security testing workflows. Invicti designs its AppSec solutions with SDLC integration in mind, which makes them a natural fit for API testing. **By adding API definitions as they are created and updated, developers can ensure that every vulnerability scan covers the entire existing attack surface at every stage where security testing is integrated, including once an application is in production.**

Invicti's out-of-the-box integrations with popular issue trackers, CI/CD systems, and collaboration tools allow companies to plug automated security testing – APIs included – into their existing workflows for maximum efficiency. Combined with extremely accurate security checks and actionable vulnerability reports complete with remediation guidance, this is a tried and true way of bringing measurable security improvements without burdening development teams with external workflows or separate toolchains.

Ensure consistent accuracy across the entire application

To make sure that you have the same level of protection all across your application, you need to use the same security checks for API-based testing and conventional testing. While this would be extremely hard to do with separate tools, Invicti makes it possible by consistently probing the entire attack surface using one integrated solution. **By running the same high-quality tests both on interactive pages and on API endpoints, you can eliminate weak spots in your overall security posture.**

As ever with automated testing, you need to strike the right balance between finding vulnerabilities and minimizing false positives. This is where Invicti's track record is second to none in the industry, with automatic vulnerability confirmation taking the guesswork out of dealing with application security reports for the majority of exploitable vulnerabilities. With well over a thousand security checks accumulated, expanded, and continuously improved since the earliest days of vulnerability scanning, combined with cutting-edge technologies such as IAST, Invicti can help to bring effective and practical application security testing into any modern web development workflow.

Enforce authenticated scanning

Authentication is another common stumbling block for automated security testing. In a world where very few sensitive resources are deliberately exposed to unauthenticated users, authenticated vulnerability scanning is already an important requirement when testing websites and applications. **Since all APIs require some kind of authentication, support for authenticated scanning is an absolute necessity to allow the scanner to access API endpoints for testing.** From basic authentication to single sign-on with OAuth2, Invicti ensures support for all popular authentication methods used by modern web applications and APIs. Authenticated vulnerability scanning provides maximum test coverage and the most realistic picture of your real-life security posture across all web-facing assets, including websites, applications, web services, and APIs.

Modern applications require modern AppSec

If you have web applications, you also have web APIs – and you need to secure both to avoid weak links in your security posture. Not that long ago, manual testing was the only way to check the API part of your web att ack surface, but manual tools and processes cannot hope to keep up with the pace of modern web development. **With enterprise applications being rapidly built from dozens of services or hundreds of microservices all communicating through APIs, extending automated vulnerability scanning to cover APIs was the logical next step for web application security – but that required advanced technical solutions and efficient workflow integration.**

Invicti provides a single platform for comprehensive web application security testing, covering both the visible and the hidden parts of your web attack surface. As you are using our industry-leading scanning technology in development and production, you can also ensure that your web APIs are subjected to the same level of vulnerability testing. All this is integrated into your existing workflows without clunky add-on toolchains or waiting for external security testing.

You now have all you need to scan and continuously secure your entire web attack surface – so get to it before the next attack comes.

About Invicti Security

Invicti Security is transforming the way web applications are secured. An AppSec leader for more than 15 years, Invicti enables organizations in every industry to continuously scan and secure all of their web applications and APIs at the speed of innovation. Through industry-leading Asset Discovery, Dynamic Application Security Testing (DAST), Interactive Application Security Testing (IAST), and Software Composition Analysis (SCA), Invicti provides a comprehensive view of an organization's entire web application portfolio and scales to cover thousands, or tens of thousands of applications. Invicti's proprietary Proof-Based Scanning technology is the first to deliver automatic verification of vulnerabilities and proof of exploit with 99.98% accuracy, returning time to development teams for critical projects and innovation. Invicti is headquartered in Austin, Texas, and serves more than 3,500 organizations all over the world.

invicti

FIND US

linkedin.com/company/invicti-security
twitter.com/invictisecurity
facebook.com/invicti-security
invicti.com
invicti.com/contact/