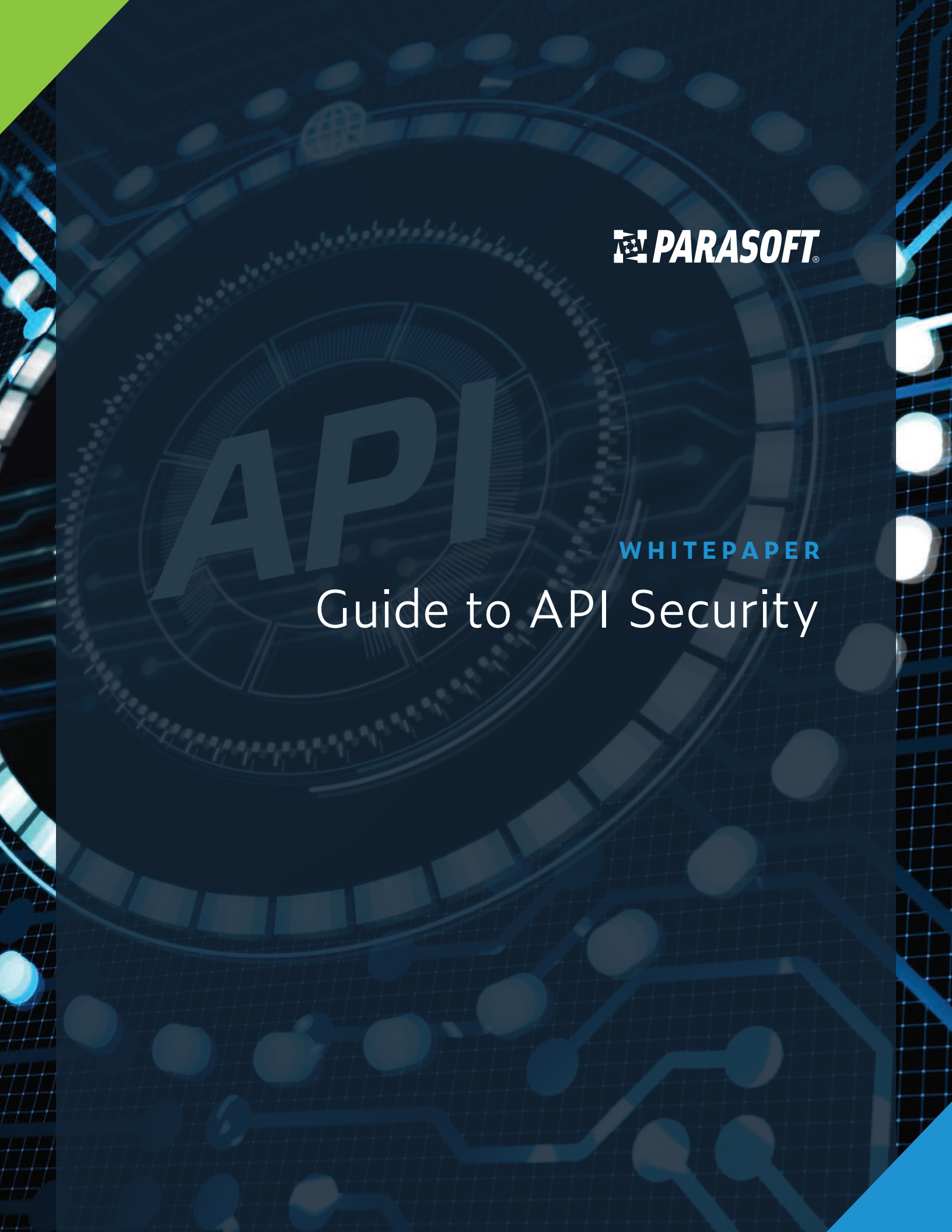


WHITEPAPER

Guide to API Security

WHY APIS ARE AN ATTRACTIVE ATTACK VECTOR

Today's adversaries are goal oriented, whether that goal is stealing trade secrets, personal information on consumers, or harming a business through a denial-of-service attack. Finding zero-day vulnerabilities and attacking custom applications is hard. A simpler attack vector is always preferred, whether that's attacking a supplier in a supply chain with poor software security practices, tricking a user into revealing insider credentials, or exploiting a known vulnerability in a common open source software component.

APIs represent over 80% of all web traffic and offer attackers similar attack techniques as vulnerable open source software.

- Like open source software, most organizations lack visibility to all the APIs they use and context about how the APIs are used. API use has increased greatly over the past 10 years, driven by the adoption of microservices to simplify and accelerate software development. According to a study by Ping Identity, 25% of the companies surveyed have over 1,000 APIs, while 35% report having between 400–1,000 APIs. More worrisome, 51% are unsure if their security team have visibility to all the APIs used in their organization.
- Like source code in open source software, API documentation may be available to an attacker and can expose an application's logic and data. Even private APIs can be easily understood if the APIs use insecure methods like HTTP, text-based REST, and SOAP protocols.
- Known vulnerabilities in open source software and in commonly used public APIs provide a ready-made attack vector for attackers. In 2019, a vulnerability in the Twitter Kit framework API failed to properly validate the api.twitter.com SSL certificate left millions of iOS users vulnerable to man in the middle attacks.

LEAKY AND POORLY DESIGNED APIS

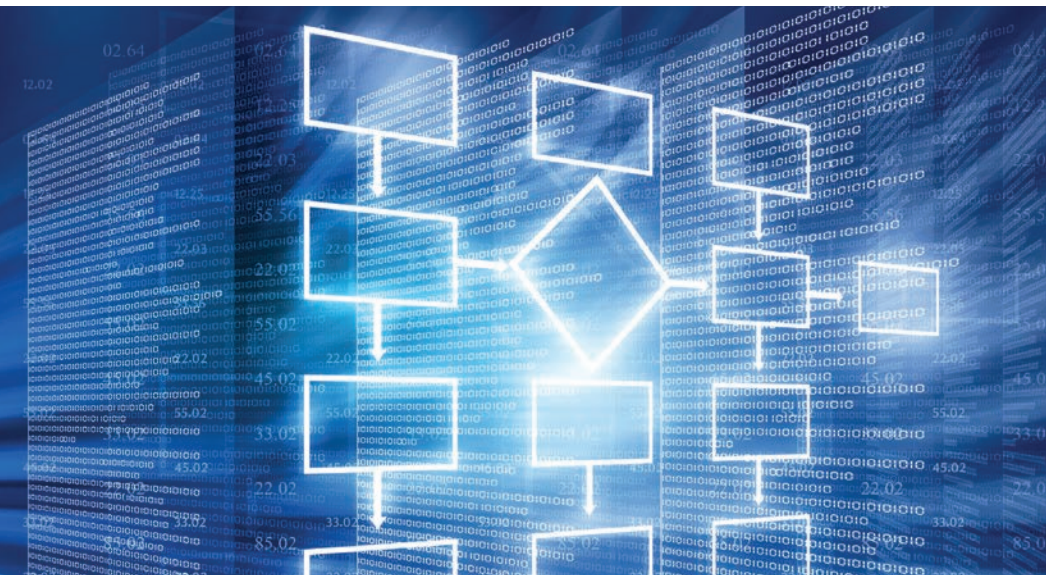
Often, the problems leading to API-related breaches are not because the attacker (and in some instances security researcher) is particularly clever or diligent. Instead, many of the issues are related to poor design and implementation of the API.

Central tenants of software security are strong authentication/authorization, the principal of least privileges, and data protection, as well as to test for both abuse and misuse cases. Recent security exposures at Clubhouse, John Deere, and Experian are examples that show how basic software security practices were not followed, leading to leaked information on users, customers, and consumers.

> "Poorly designed APIs can disrupt businesses and erode confidence in consumers especially when they expose privacy issues."
>
> —Security solutions expert at Parasoft

"Poorly designed APIs can disrupt businesses and erode confidence in consumers especially when they expose privacy issues. The Clubhouse API vulnerabilities that allow ghosting, trolling, and eavesdropping are examples of why security must be designed in and follow sound software engineering practices where least privilege is a core building block to eliminate a class of security issues," explains the security solutions expert at Parasoft.

In each case, the security researchers who identified the vulnerabilities simply used the API—as it was designed—to access information on millions of users and/or customers, as well as misuse and abuse the intended functionality in the API. Some of the API designs were open to any user without authentication and provided the ability to look up any (or all) users and download names, addresses, purchases, and other information. These were avoidable errors in the design, implementation, and testing of the APIs.



The security researchers who identified the vulnerabilities simply used the API—as it was designed—to access information on millions of users and/or customers.

TECHNICAL CHALLENGES ARE SIGNIFICANT

A well-designed API requires input from product owners, architects, and security to properly understand abuse and misuse cases to design security in. Using the examples above, authentication controls are critical since the authentication mechanism is exposed to everyone, including an organization's adversaries.

Authentication controls, therefore, go beyond simply requiring users and systems to authenticate. They include controls to prevent credential stuffing and brute force attacks, weak passwords, revealing sensitive information like auth tokens and weak encryption keys.

Strong authentication controls are only a first step. As noted by the OWASP API Security Top 10, organizations need to also ensure proper authorization at both the object level and function level.

- Input validation is required to prevent injection flaws.
- Rate limiting is required to restrict the number of lookups a single user could conduct each minute, hour, or day.
- Adequate logging and monitoring are required to expose attacks more quickly.

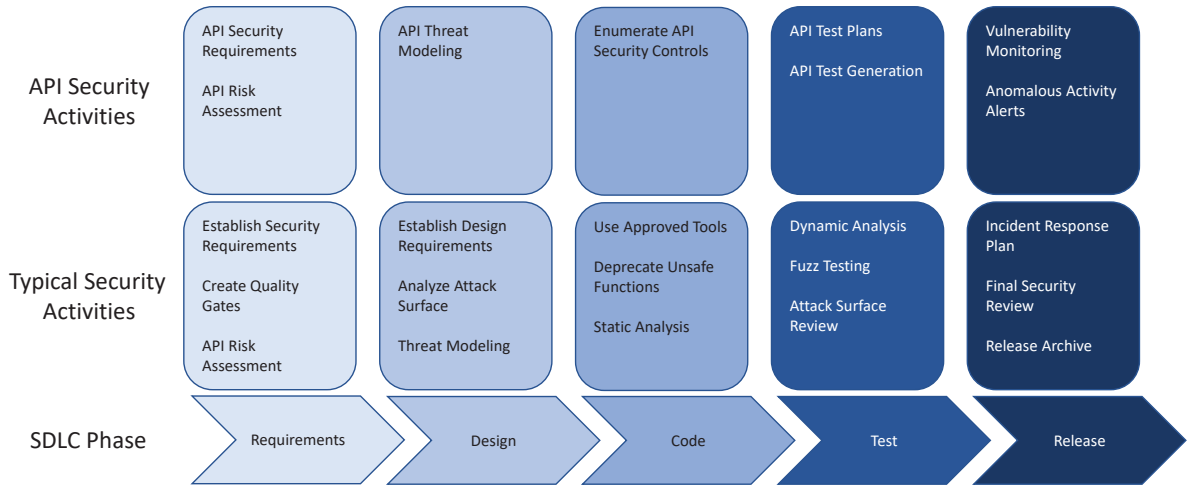


Figure 1: Mapping API security to the SDLC.

Development and security cannot rely just on traditional testing tools to catch these oversights. To prevent breaches, organizations must:

- Recognize threats in the design phase of the API software development life cycle.
- Clearly specify controls for proper implementation in the build phase.
- Communicate functionality and logic to ensure proper and adequate testing.

Traditional testing tools are focused on analyzing code using static analysis to identify coding errors or running scripted "attacks" using dynamic analysis tools. Manual penetration testing attempts to identify misuse cases, often by manipulating input to the user interface. Comprehensive API security testing isn't addressed directly by any of these methods.

DevOps engineering, compliance, and security teams face several impediments when attempting to secure APIs, including:

» **API discovery.** Visibility to all APIs is required. One cannot secure something without first knowing it exists. Understanding the attack surface and which APIs are exposed is essential for security testing.

API test coverage and discovery are important data points to minimize risk associated with API attacks. As mentioned, the shift from monolithic applications to microservices has greatly increased the number of API in the average organization. This fundamental shift will require organizations to be more proactive in their API discovery and attack surface analysis.

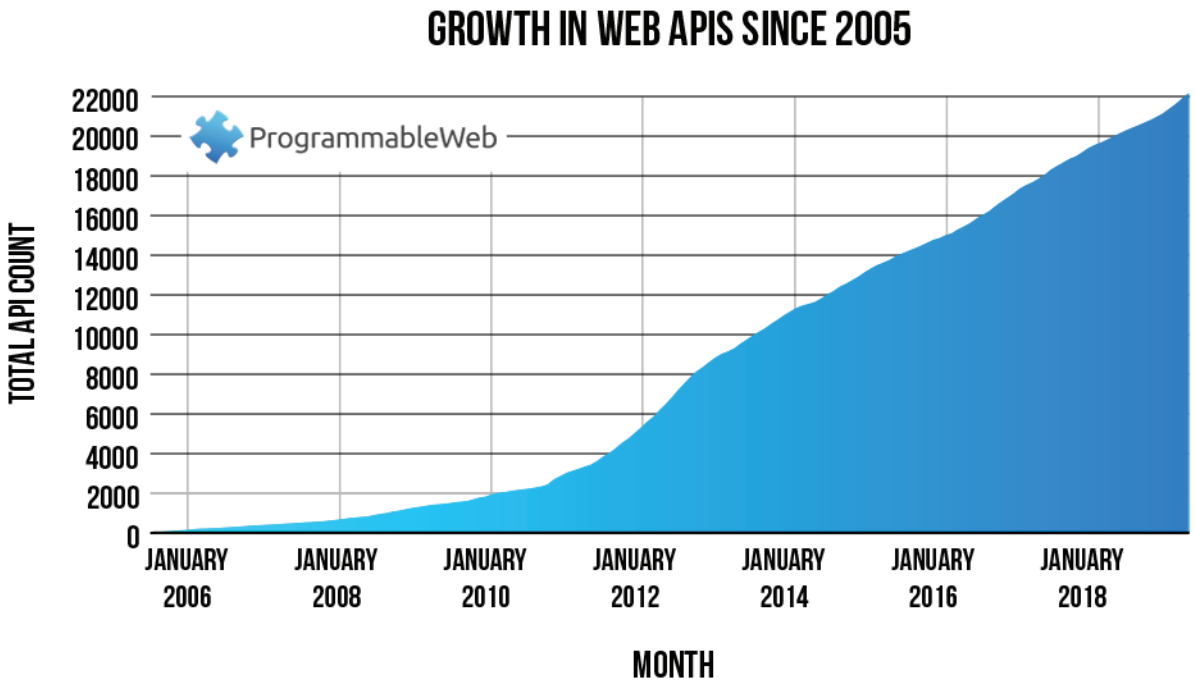


Figure 2:
Use of APIs is on the rise.

- **Poorly documented and undocumented APIs.** In a best practice, software projects include detailed documentation for internal parties to simplify maintenance and to help developers new to a project understand how the application works. Likewise, API require documentation for developers using the API, whether internal developers and testers or third parties.

  In reality, pressure to add features and accelerate time to market often means documentation is limited. This is especially true with APIs, which are usually simpler and/or intended for internal use. Without proper internal documentation, however, DevOps engineering has imperfect knowledge on how an API works. Without API documentation, including contracts, definitions, and specifications, QA and security are left to model the intended behavior of APIs and guess use and misuse cases, including usage constraints.

- **Testers are not trained in security.** Security resources are scarce in most organizations. It's a growing national issue. QA and DevOps engineering are usually focused on functional testing, ensuring the features listed in the requirements document perform to specifications. While this includes use cases like throughput and capacity testing, it can miss security tests that often focus on abuse and misuse cases (such as, calling an API with intentionally wrong arguments or not properly handling the values returned by the API).

- **API interactions are complex**. Security and testing are often familiar with use cases for a user interface. API use cases are more complex, in particular between microservices. While the security issues and exposures discussed above are straightforward, the rapid increase in microservices and resulting APIs complicates security, potentially allowing sophisticated attackers to link multiple APIs in a multistep attack.

What are contracts, definitions, and specifications?

APIs require documentation for developers, and users—both internal and external. Note that not all users are entitled to internal documentation.

- **Contracts** define a "service level agreement" between the API provider and consumer describing how the API will behave, including endpoint URLs, the actions of each endpoint, and valid arguments. The contract allows predictability around functionality for applications calling the API.

- **Specifications** are descriptions of how an API behaves and functions, including communication protocol, syntax, rules for response messages, and expected results from a call. Common specifications include REST, SOAP, and RPC.

- **Definitions** are similar to specifications in detailing the functionality of the API. Definitions are machine readable and help organizations visualize how the API functions, outline linkages to other APIs, create test cases, generate implementation code, and use tooling to monitor API performance.

- Access control policies with different hierarchies, groups, and roles can be confusing to defenders and lead to inadequate security controls. Further, while testers may understand intended use cases, attackers are focused on misuse cases and leveraging the API in unintended ways to bypass security controls.
- To properly assess API interactions and detect exposed attack surfaces, organizations need to understand the intended paths defined in API documentation, as well as the unintended paths that could be poorly defined in API documentation. Both the intended and unintended paths need to be explored in depth to determine the extent of possible abuse and misuse.

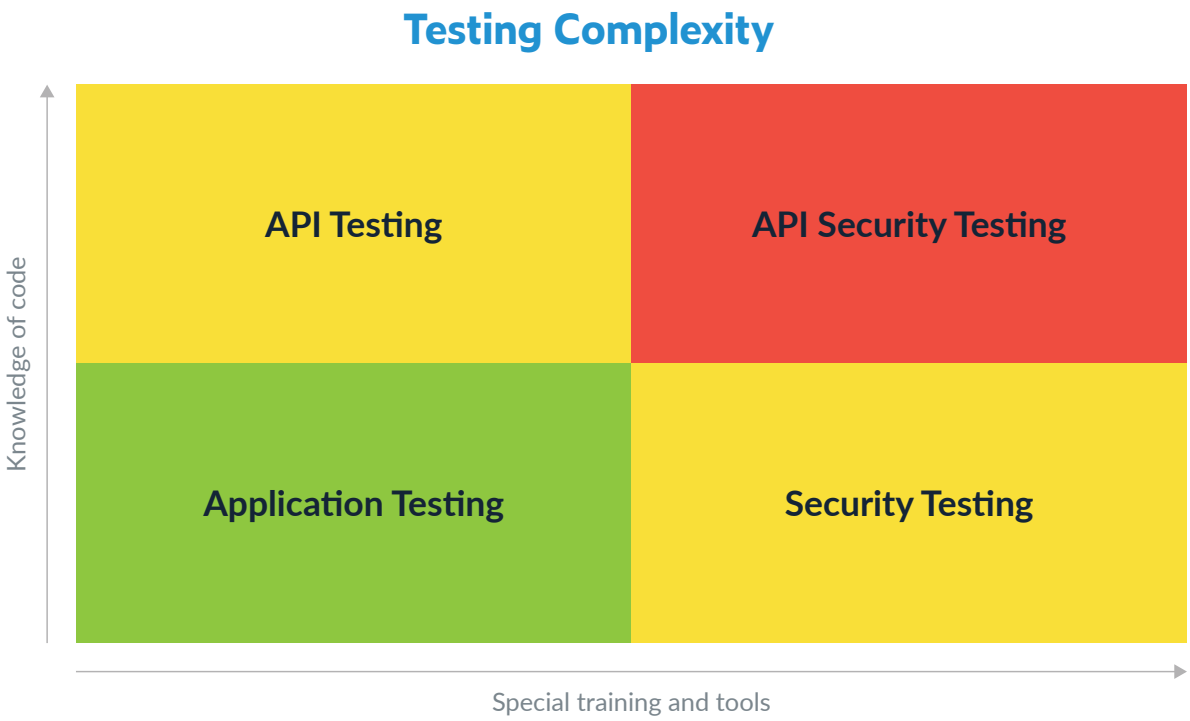


Figure 3: Complexity in API security testing.

- **Security doesn't understand API functionality.** Security testers understand the applications interface visible to a user, but rarely understand proper use of APIs. A user interface has an expected path of activity. For example, a user logs in, searches for an item, adds it to a shopping cart, provides payment details, and checks out.

  Testing and securing APIs require an understanding of an application's logic about which security often has less information. APIs are more like Lego blocks that can be assembled in a variety of ways. Adversaries may use APIs in unintended ways—even linking APIs to attack an application—like how many ways can one reach the shopping cart or store payment information? Complete testing requires programmatic visibility to the APIs and their interactions, edge cases, protocols, and knowledge of expected data types. Creating tests is possible with sufficient resources, of course, but this skill has a very steep learning curve.

"One interesting aspect of testing APIs, whether it's for functionality, performance, or security, is that APIs are surprisingly flexible. In a sense, they're like Lego pieces that can be assembled in multiple ways, both expected and unexpected. Being able to fully test possible combinations is critical to making sure that your APIs are secure," says Arthur Hicken, chief evangelist at Parasoft.

> "APIs are surprisingly flexible … Being able to fully test possible combinations is critical to making sure that your APIs are secure."
>
> —Arthur Hicken, chief evangelist at Parasoft

WHAT CAN API SECURITY TESTING DO FOR YOU?

TRADITIONAL SECURITY TESTING TOOLS STRUGGLE

Traditional tools used in traditional ways are only a partial solution. Static analysis tools typically examine source code. While they can identify some issues with API, their lack of knowledge on intended functionality limits their completeness.

Dynamic analysis scanners also help, but often only look for entry points in the user interface and may have no awareness of APIs and their functionality. While it is possible for a skilled pen tester to identify weaknesses in APIs, it scales poorly and isn't practical from a time and cost standpoint. It's also incompatible with rapid development methodologies like CI/CD and DevSecOps.

These tools don't understand API authentication mechanisms and need to be aware of protocols such as OAuth2 and JSON Web Tokens. They also struggle with content types and HTTP responses that impact how vulnerabilities in APIs are detected. In addition, there are no links these tools can crawl to discover APIs, which fundamentally changes security testing for APIs. In short, existing security testing tools weren't designed to test the interactions between APIs common in today's applications.

ADOPTING BEST PRACTICES FOR API SECURITY TESTING

Adding API security testing to the software development life cycle is possible, even in rapid development environments, when organizations provide DevOps engineers with the correct tools and support. Importantly, security testing must consider common design flaws as well as implementation and configuration errors.

Building more secure software is not a secret. There are several guidelines and standards organizations can use to identify and mitigate risk. One, the OWASP Application Security Verification Standards (ASVS), is particularly helpful when considering designing and implementing APIs.

The Application Security Verification Standard describes itself as "a list of application security requirements or tests that can be used by architects, developers, testers, security professionals, tool vendors, and consumers to define, build, test and verify secure applications."

While achieving ASVS certification is an admirable goal, organizations seeking to incrementally improve their software security programs can use ASVS guidelines in a less formal way.

IT TAKES MORE THAN SHIFTING LEFT

Integrating API security activities early in the software development life cycle is possible, even in rapid development environments, when organizations provide their development teams with the correct tools and support.

START AT THE BEGINNING

Security testing early is essential for preventing and detecting security issues with APIs, but everything starts with a good design. Designing a secure API is not a secret. Among other things, APIs require clear definitions for the parameters allowed by security controls, rate metering to prevent brute force attacks, and enforcement of strong authentication mechanisms to prevent API misuse. There are resources to help, including the OWASP Enterprise Security API (ESAPI) design patterns.

SHIFT LEFT AND BUILD BETTER

"Shift left" refers to moving security testing activities earlier in the software development life cycle into developers' daily activities. Postponing security testing until late in the software development life cycle results in delays and higher development costs because fixing bugs becomes more expensive in the maintenance phase, as shown in IBM's study (figure 4).

The rising costs are due, in large part, to the difficulty of refactoring software to fix bugs (or change open source libraries), while maintaining the features and functionality of the original design.

In the ASVS standards, shifting left includes considering API functionality, authentication, and authorization in the design phase of the life cycle. This entails verifying:

- Communications between application components, including APIs, are authenticated.
- All components use least privilege principles.
- Calls to APIs are encrypted.
- API URLs don't expose sensitive information such as session tokens.

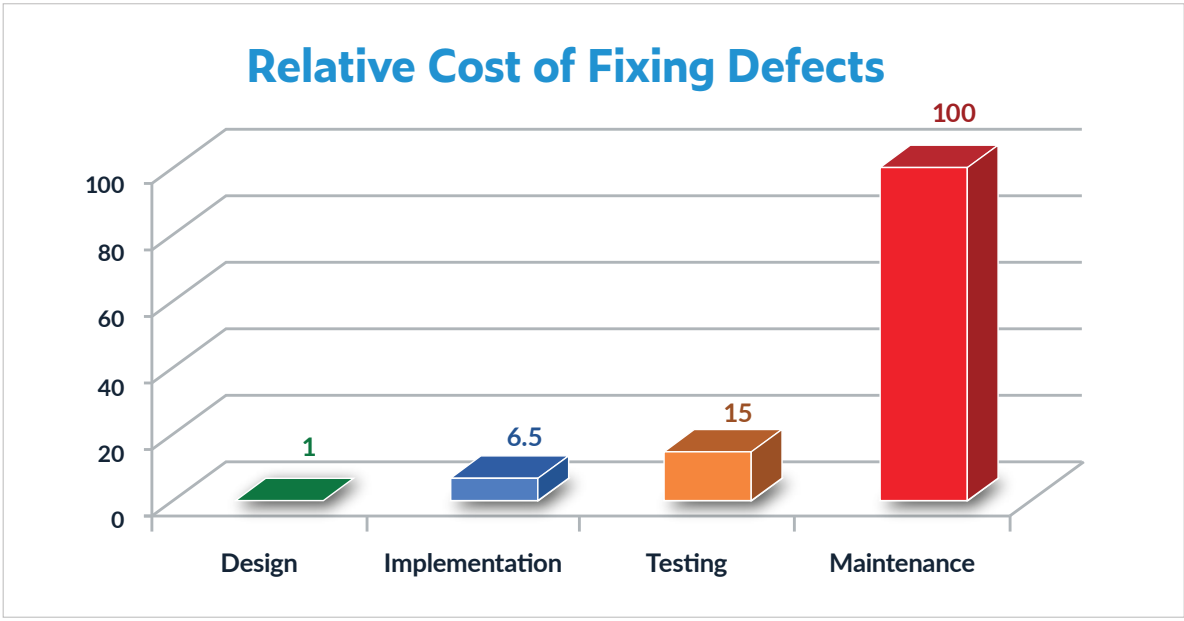


Figure 4: Remediation cost by each stage of the SDLC.

KNOW WHAT TO DEFEND AGAINST

OWASP also provides guidance through the OWASP API Top 10, which describes the consensus opinion on the most common and damaging vulnerabilities found in APIs. As you can see, many of these are concerned with identity and access management issues. Some, like Injection and Logging issues are already used and integrated in most application security testing suites.

"It's not a matter of shifting left but pushing left with informed context about likely threats that can be codified into misuse and abuse cases to identify attack surface areas and exposed APIs," says the security solutions expert at Parasoft. "This can be used to prevent common things we see in the OWASP API Top 10, as well as guide a more complete and thorough approach for API security testing."

2019 OWASP API Top 10

API1: Broken Object Level Authorization

API2: Broken User Authentication

API3: Excessive Data Exposure

API4: Lack of Resources & Rate Limiting

API5: Broken Function Level Authorization

API6: Mass Assignment

API7: Security Misconfiguration

API8: Injection

API9: Improper Assets Management

API10: Insufficient Logging & Monitoring

TRAINING DEVELOPMENT AND TESTING

Recent research indicates worldwide there are between 3.5 million and 4 million unfilled cybersecurity jobs. It's not likely to get better soon, as help from universities is not on the way.

Training developers in secure coding has long been a compliance requirement—and treated as such. Annual, 3 to 4 hour computer-based training courses were assigned with little or no follow up. A better approach is to treat training as a process. Deliver more frequent, more consumable, on-demand content as developers confront coding issues. When delivered in the developers' workspace, these "micro courses" reinforce secure coding practices on a daily basis and greatly increase knowledge retention.

> "It's not a matter of shifting left but pushing left with informed context about likely threats that can be codified."
>
> —Security solutions expert at Parasoft

A NOVEL APPROACH TO COMPREHENSIVE API SECURITY TESTING

A better method for testing APIs is to leverage functional testing to provide visibility to the APIs underlying the application and their functionality, and augment test generation using artificial intelligence (AI) to build appropriate test cases.

"API testing can be surprisingly complex. It requires knowledge of testing, how the APIs are used, and how they could be used in different ways than the user stories. When you add security to the mix, you need even more knowledge because again the API layer is deeper and more technical than the UI layer," explains Arthur Hicken, chief evangelist at Parasoft.

> "API testing can be surprisingly complex. When you add security to the mix ... the API layer is deeper and more technical than the UI layer."
>
> —Arthur Hicken, chief evangelist at Parasoft

Parasoft SOAtest's Smart API Test Generator leverages a browser plugin that uses AI to automatically convert manual and automated UI tests into automated API tests. Instead of simply collecting traffic, recording it, and playing it back, Smart API Test Generator applies AI to discover meaningful patterns, apply threat models, and understand relationships between those API calls. It can then generate automated API test scenarios that perform the same actions as your UI tests but are fully automated and easily extendable.

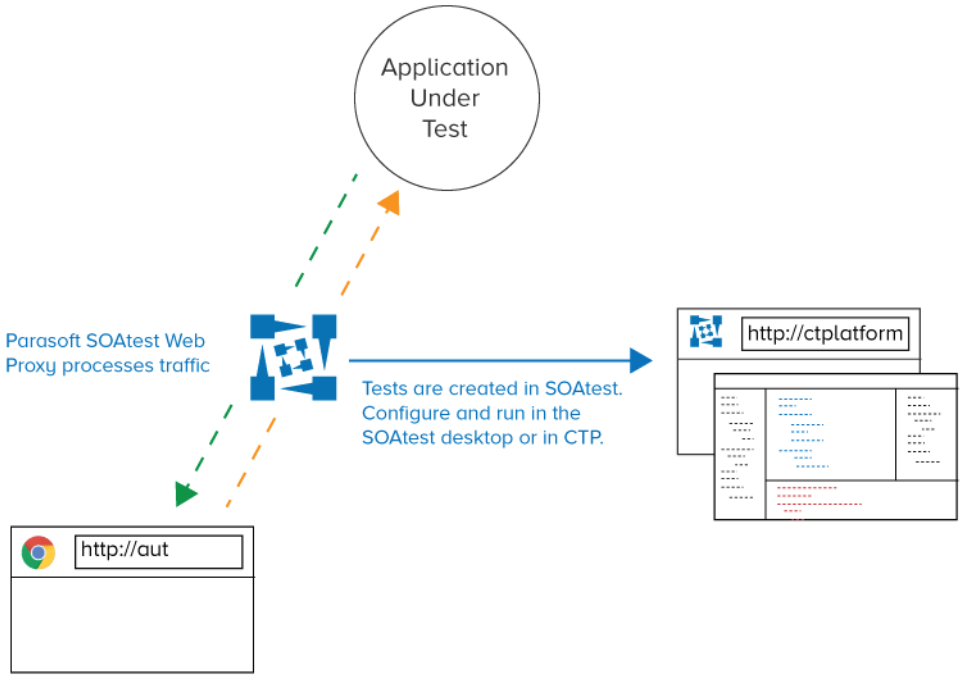


Figure 5: Automatically converting manual and automated UI tests into automated API tests.

Smart API Test Generator provides development, test, and security with several benefits.

Build API security tests without security expertise. Security resources are limited at even the largest organizations, in particular resources with experience in API security testing. SOAtest Smart API Test Generator eliminates barriers to API testing and flattens the learning curve, allowing novice users to build effective, scriptless test suites. No coding required.

Augment security team knowledge. SOAtest's Smart API Test Generator can also be used while security teams are testing applications, further extending the test cases generated.

Accelerate testing cycles. Automating the development of API tests means spending less time investigating test cases, looking for patterns, and manually building relationships to form each test scenario. When the UI changes, Smart API Test Generator's scriptless API tests are easily updated.

Scalable API testing. Automatically generated and scriptless API tests using visual tooling and test flow logic allow teams to cover more application logic with less effort, resulting in complete, end-to-end test scenarios.

Uncover and test undocumented APIs. By monitoring the interactions between the UI and the underlying APIs, the Smart API Test Generator not only tests the known and OpenAPI documented APIs, but also uncovers and builds tests for the undocumented APIs that were intended only for internal use and escaped your governance process.

Leverage dynamic analysis for deeper coverage. Through its integration with Dynamic Analysis tools, SOAtest can guide DAST testing to expose the attack surface of APIs and provide deeper testing coverage that standalone DAST tools can't reach.

Contextual training. Parasoft security solutions provide specific contextual training as well as high-level security training is available directly through the documentation. Training ranges from free to paid and includes video, hands-on, and more.

Teams can also take advantage of the full security training and certification program available from Parasoft partners.

TEST FASTER. TEST BETTER.

APIs will continue to grow as a preferred attack vector due to their increased use, organizations' limited visibility and awareness of their misuse cases, and incomplete testing. Leveraging UI testing and AI to automate test development allows organizations to scale comprehensive API testing across their entire application inventory without the need for additional security resources.

TAKE THE NEXT STEP

Learn how Parasoft's SOAtest Smart API Test Generator can help your team test faster and better. Request a demo with one of our experts today.

ABOUT PARASOFT

Parasoft helps organizations continuously deliver quality software with its market-proven, integrated suite of automated software testing tools. Supporting the embedded, enterprise, and IoT markets, Parasoft's technologies reduce the time, effort, and cost of delivering secure, reliable, and compliant software by integrating everything from deep code analysis and unit testing to web UI and API testing, plus service virtualization and complete code coverage, into the delivery pipeline. Bringing all this together, Parasoft's award winning reporting and analytics dashboard delivers a centralized view of quality enabling organizations to deliver with confidence and succeed in today's most strategic ecosystems and development initiatives — security, safety-critical, Agile, DevOps, and continuous testing.