noname

# Introduction to Shift Left API Security Testing

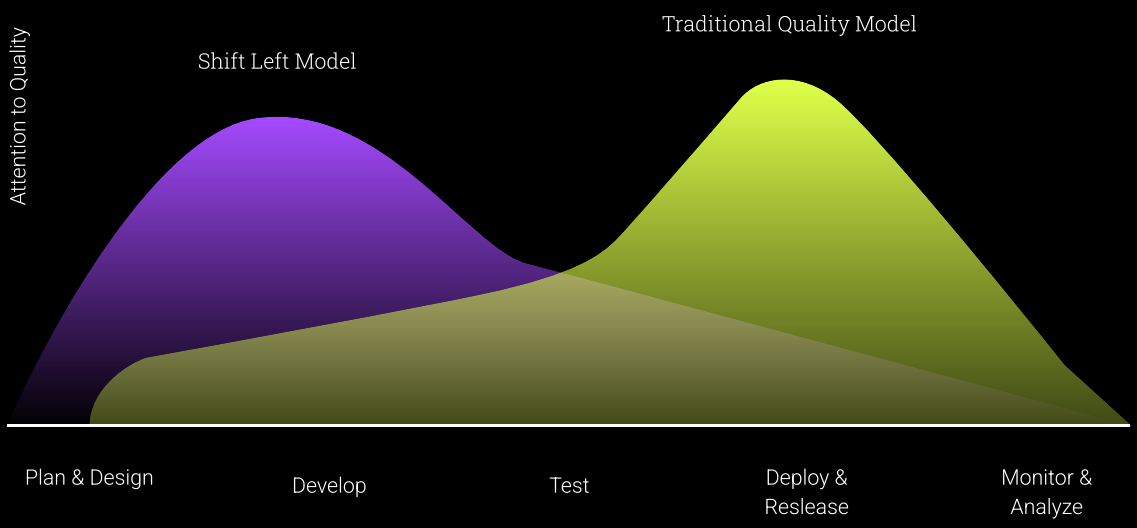
# What is Shift Left Testing?

With the number of APIs skyrocketing, companies are facing increasing challenges when it comes to security. Oftentimes, either there aren't enough security personnel who know how to test APIs, the number of APIs are growing faster than the security team can keep up with, or the existing security tools lack adequate coverage. Any one of these three scenarios can spell disaster for your environment. However, there is one overlooked aspect that could also weaken your API security posture if not addressed – and that's testing APIs early in the development process.

When it comes to API development, it's not just a matter of testing but also when you test your APIs. The traditional model places testing closer to the deployment phase. And though this is definitely a vital step, testing only during this time is insufficient and can lead to serious vulnerabilities. How exactly? By consolidating testing into one phase of the software development lifecycle (SDLC), you create a bottleneck in the process as there is a never ending supply of code to test.

To alleviate this bottleneck and expedite the process, certain steps in this evaluation phase may be incomplete or mistakenly bypassed altogether. As a result, code quality suffers and your API attack surface widens. And remember, this is the one phase in the process for testing - so anything missed here won't be caught until it's too late.

Shift Left is an approach of moving a variety of tasks earlier in the development process. This means that tasks that are traditionally done at a later stage of the operations should instead be performed at earlier stages–particularly those related to API security and software testing.

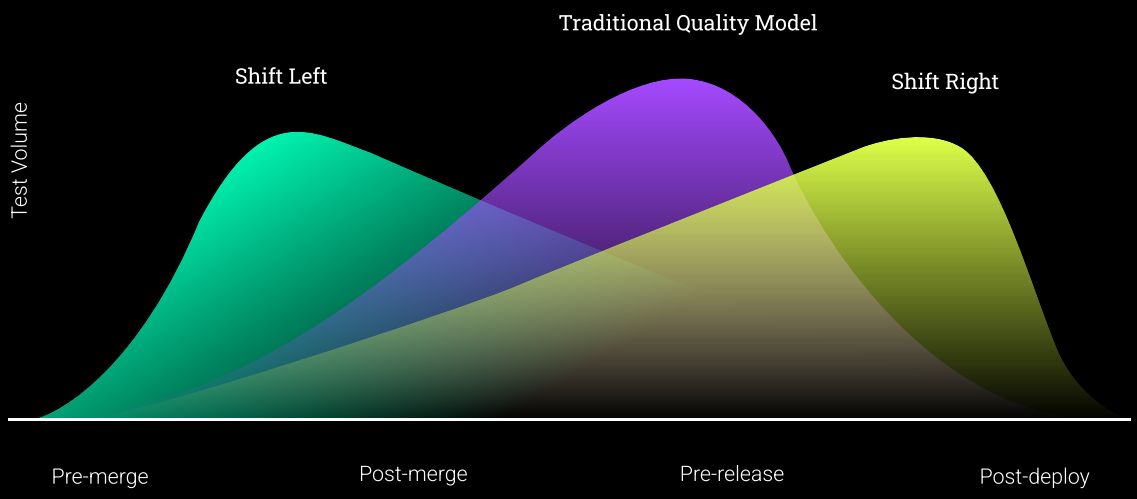



With security and testing baked into each step of the API development or DevOps process, a shift left approach ensures developers will be monitoring for vulnerabilities throughout the lifecycle. Shift left principles enable security teams to increase developer autonomy by providing support, expertise, and tooling while still delivering the required level of oversight. Developers can release more secure code at scale, build API security into the design, and make fixes early in the development process instead of scrambling to fix them later. Code testers are able to evaluate features as they are created and help ensure higher quality.

The shift left testing process is continuous, running parallel with development, and involves continuous communication between the clients, developers, and testers. The shift left testing process typically involves several steps:

1. Studying client requirements, application behavior, and end-user expectations
2. Developing tests for unit, integration, and functionality
3. Executing tests via end-to-end automation
4. Running non-UI tests as they are implemented

The practice also helps minimize defects along the way by encouraging both Test-Driven Development (TDD) and Behavior-Driven Development (BDD).




## Shift Left vs Shift Right: Why Shift Left Testing?

As we just established, a shift left security approach moves testing to the left on the timeline, so the team performs tests earlier and more often in the life cycle. In contrast, a shift right approach considers testing in production with real users to be more useful. The goals of shift right testing include:

- Verifying backend stability
- Establishing software usability via actual user preference such as Blue Green deployments or A/B testing
- Identifying real production issues when end users experience them

Shift right testing ensures real world software stability and performance by testing in production environments, with scenarios that aren't possible in the development environment, and improving user experience by collecting feedback and reviews from application users. Though this approach can improve some aspects of performance, it can leave you vulnerable since you won't be able to catch misconfigurations and design flaws sooner.

Shift left testing saves time by identifying bugs earlier in the life cycle. More rigorous detection of errors and bottlenecks in advance enable testers to improve initial designs and develop alternatives. The shift left approach ensures quality by allowing the development team to bake automation tools like API testing or unit testing right into the building process. Ultimately, apps, microservices, and APIs are more sufficiently protected. Waiting to test during production means the team is always playing catch up and fighting inherently greater risk.

## Types of Shift Left Testing

There are four general approaches to shifting testing earlier in the life cycle. Each approach offers a more accelerated pace than the one before it. These types of testing include: traditional shift left testing, which is the most moderate approach in terms of timing, followed by incremental shift left testing, agile shift left, and the almost immediate model-based shift left testing.

### Traditional Shift Left Testing

Traditional shift left emphasizes integration testing and unit testing, for example using modern test tools and API testing, over acceptance and system level testing.

### Shift left security in DevOps

The shift left approach in DevOps, much like other agile and DevOps projects, is characterized by numerous sprints of short duration in place of a single larger shift left testing project. Shift left in agile is often part of test-driven development (TDD).

### Incremental Shift Left Testing

A common technique, incremental shift left testing breaks the development cycle down into much smaller pieces, starting software testing earlier on the timeline so the pieces can build on each other.

### Model-Based Shift Left Testing

This type of shift left testing is earliest in the development cycle. Model testing essentially introduces models of executable requirements and tests them immediately. This is in contrast to the brief sprints of Shift left DevSecOps testing, the slightly longer wait of incremental shift left testing, or the longest wait with traditional shift left testing.

# How to Implement Shift Left Testing

There are some basic things to keep in mind when implementing shift left security and testing.

## Define goals.

Since shift left demands organizational and cultural change, management should first define goals for the process to ensure any new tool or process introduced into the development cycle will work for the team's existing development and testing methodologies.

## Automate security processes with security automation tools.

Use continuous integration (CI) tools, issue tracking tools, and test automation tools to help teams establish and automate security practices during all stages of the life cycle.

## Understand the supply chain.

Know how and where your organization develops apps and software before architecting a comprehensive shift left security program. The security risk posture of the supply chain is largely dependent on the security proficiency of others in the chain. This also helps your developers identify small steps where testing might be placed earlier in the life cycle.

## Train development teams in coding securely and visible culture.

Do not neglect the human aspects of risk during the move to shift left security. Constant visibility into application security should be part of the culture.

# Value of Shift Left API Testing

Shifting testing earlier in the development process offers a myriad of benefits for developers. These benefits can be summarized in two distinct outcomes: fixing vulnerabilities before production and innovating faster.

## Address Vulnerabilities Before Production

As discussed earlier in this document, testing APIs early and often shrinks the API attack surface and reduces the risk of successful attacks in production. Now you can minimize the potential for data leaks and manipulations to e-commerce APIs without any modifications to production infrastructure. By finding and fixing issues earlier, not only can you remediate faster, but you also can lower remediation costs by up to 100x. This includes improving compliance, and avoiding regulatory fines and reputational damage from incidents.

## Innovate Faster

By testing APIs early, you're also able to improve security without sacrificing velocity. A shift left approach empowers you to eliminate the bottlenecks identified earlier. You can increase your organization's confidence in APIs with continuous testing and reduce redundant pentesting and other third-party security testing costs. Ultimately, you are able to deliver secure code without having to become a security expert. As you can see, a practitioner can benefit immensely from using a shift left approach. But it's also very important to note how API security testing provides some very real benefits to your bottom line – specifically in terms of reducing risk, reducing costs, and increasing revenue.

## Reduce Risk

API vulnerabilities in production represent immediate risks, yet fully deploying runtime protection across all production environments can take time. Therefore, organizations should pursue both in parallel. Test in development to identify and remediate vulnerabilities, and implement posture management and runtime protection solutions in production. This provides the fastest path to eliminating vulnerabilities.

## Reduce Costs

Testing for API security early and often significantly reduces remediation costs, as vulnerabilities can be eliminated before ever presenting a risk to the organization in production.

## Increase Revenue

Automated testing allows developers to move quickly to meet customers' needs by ensuring that new releases are secure and unlikely to require refactoring or costly remediation in the future. Because they can deliver on customers expectations, and therefore deliver great customer experiences, they can maintain their competitive edge or expand a competitive lead in the market.

# About Noname Security

Noname Security is the only company taking a complete, proactive approach to API Security. Noname works with 20% of the Fortune 500 and covers the entire API security scope across four pillars — Discovery, Posture Management, Runtime Security, and API Security Testing. Noname Security is privately held, remote-first with headquarters in Silicon Valley, California, and offices in London.

nonamesecurity.com | info@nonamesecurity.com | +1 (415) 993-7371