Security

Practical AppSec, Part II: The Limitations of "Shift Left" (and Why Runtime Is the Right Time)

Jeanette Sherman
June 24, 2024

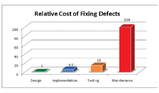
In the first part of our Practical AppSec series, we talked about the reasons developers don't trust AppSec – and how that distrust is rooted in tools that focus on theoretical vulnerabilities rather than real, provably exploitable ones.

Even when security alerts frustrate engineering and security teams, they're still plugging away at those dashboards full of CVEs, trying to make them all turn green. Why? Well…because of shift left, of course.

We all know it's the right way to do application security, because it saves money. If you read security blogs, you'll have seen a specific number over and over. Fixing vulnerabilities earlier in the development life cycle saves you up to 100x versus fixing them later, according to pretty much every security vendor out there. Except…does it?

"Shift Left" Rests on Shaky Foundations

OWASP founder Mark Curphey earlier this year produced compelling evidence that the much-vaunted 100x statistic was just an "urban myth," sourced from an internal IBM training manual based on data gathered, *at absolute latest*, in 1981. A lot of things have changed in software development in that time (I can certainly think of a few), but the statistic lingers on stubbornly – probably because it made sense for a lot of vendors selling shift-left products to use it in their ROI calculators.



The IBM chart that launched a thousand "shift left" initiatives – but it came from 1981 and wasn't based on real data

According to recent research, shifting left may not really save any money or effort at all. In a paper from 2006, three researchers tried to find evidence for the "delayed issue effect," in which an issue requires more effort to resolve when it is found later in the development cycle.

The team "found no evidence for the delayed issue effect," and concluded that it was a "historical relic that occurs intermittently only in certain kinds of projects."

What if we rethought application security without the truisms and urban myths? What if we thought about what would actually work best for the problems and situations today's organizations find themselves in?

Let's take a moment for a thought experiment that takes us outside the realm of development – and into the kitchen.

The Peanut Butter Jar: A Thought Experiment

You're cooking for a friend, and you know he has a peanut allergy. You're a considerate friend (and certainly don't want to end the night with anaphylaxis), so you don't use any peanuts in the food. You even check the labels of everything you use, just to be sure.

When your friend comes over, he says he's got something new – a detector that can determine if it's safe for him to eat the meal you've cooked. He turns on the detector, and it flashes red. "It's ok," he says. "I'll grab something on my way back."

"Hold on a minute," you tell him. "I was so careful to make sure there were no peanuts anywhere in our meals. You can search my whole kitchen."

So the two of you do just that, and after an hour of searching, you find it: a small, unopened jar of peanut butter on the back of a shelf. "I didn't use this! It's still sealed!" you say.

Even if no one *intentionally* lied about the benefits of pushing security earlier into the SDLC: the frustrations, wasted effort, and wasted time of today's AppSec programs are all a debt being paid to the foundational myth of "shift left."

But a new era is coming – as organizations search for lean, practical AppSec solutions, they're learning that the only way to really see what's exploitable is to check on an application while it's running.

If scanning a running application is more focused and efficient than scanning the content of a manifest file, why isn't everyone already doing it – and what else could a solution with that kind of visibility do for an AppSec program?

Stay tuned: we'll talk about the answers to these questions in the final part of this blog series – Practical AppSec, Part III: How Oligo Keeps Focused on Real Risks.

Related Posts

Security

Recent CrowdStrike Outage Emphasizes the Need for eBPF-Based Sensors

Guy Kaplan
July 19, 2024

Security

App-Level eBPF Applications - User vs. Kernel Probes

Avi Lumelsky
July 1, 2024

Security

Critical RCE Vulnerabilities in OpenSSH (CVE-2024-6387, CVE-2024-6409) - How to Detect and Mitigate

Guy Kaplan
July 1, 2024