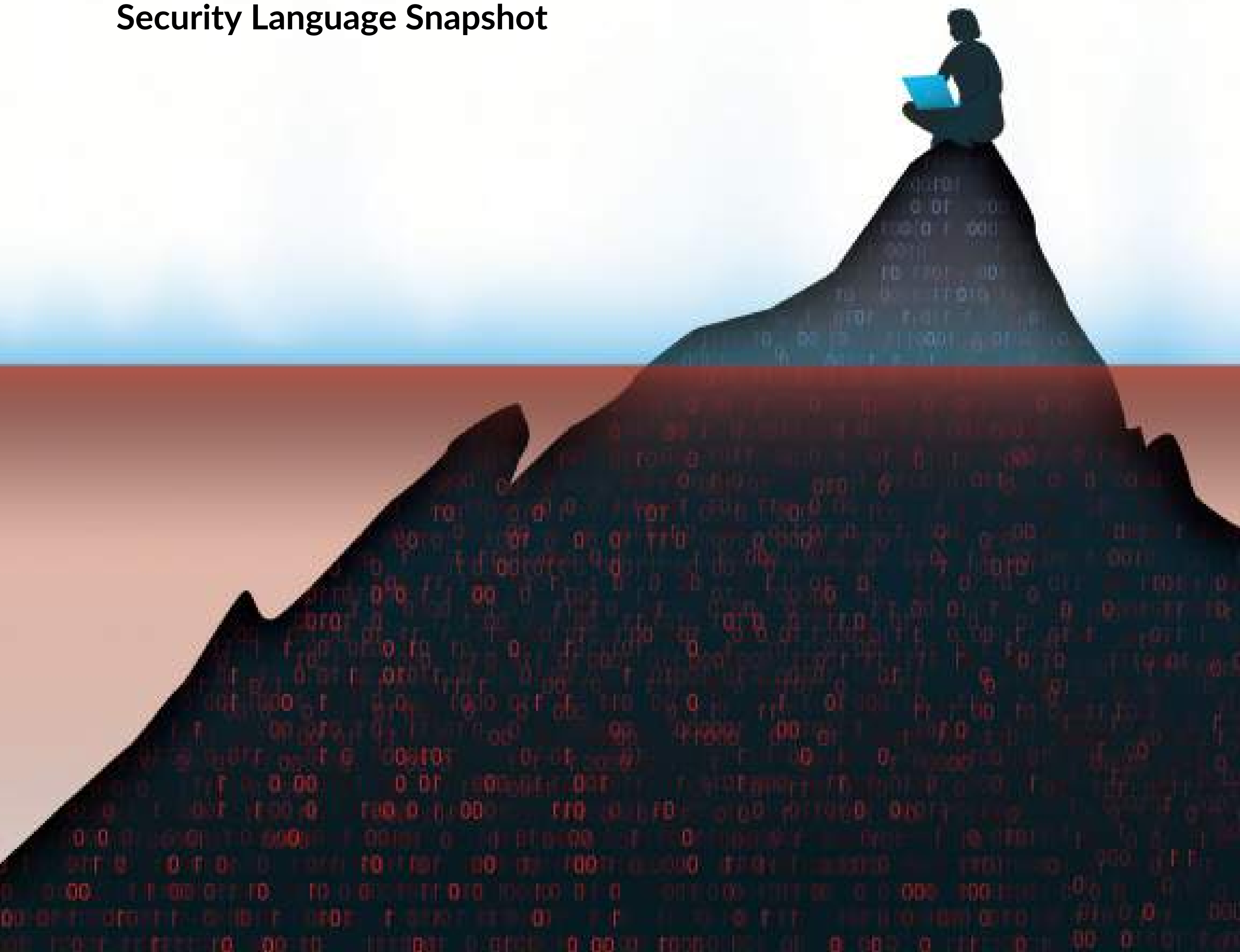
VERACODE

2024

Security Debt by Language

2024 State of Software Security Language Snapshot



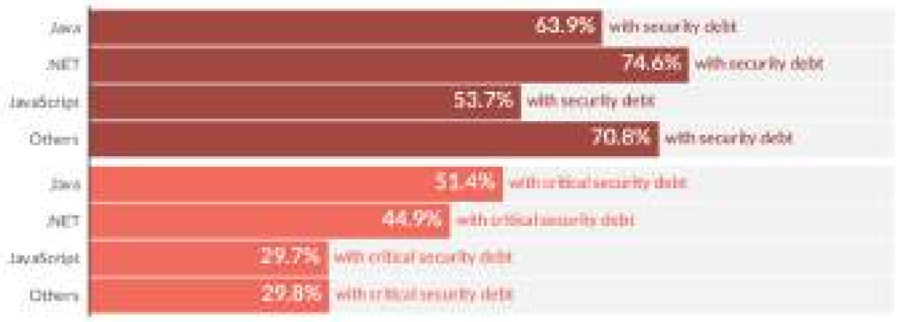
Veracode's State of Software Security 2024 spotlights the pressing issue of security debt in today's applications and provides a wake-up call to organizations worldwide. In particular, it seeks to answer questions such as "How risky is security debt really? How should it be tackled? And what's the best way to do it?" Whereas the main report analyzes security debt collectively across all organizations, this snapshot focuses exclusively on findings and differences across Java, JavaScript, and .NET applications.

The demand for speed and innovation in software development has resulted in the accumulation of risk known as security debt. For the sake of consistency in this analysis, security debt is defined as all flaws that persist unremediated for longer than a year. Overall, we found flaws exceeding that threshold in 71% of organizations, making security debt far from a rare phenomenon.

One of the key observations in the SoSS 2024 is that the prevalence of security debt varies quite a bit among the different languages. Per the top half of Figure 1, 75% of organizations that run .NET applications had security debt—the highest among the four popular languages shown here. That's followed by Java at 64% of organizations and JavaScript is considerably below the average at 54%.

Figure 1: Prevalence of language-specific security debt among organizations

1

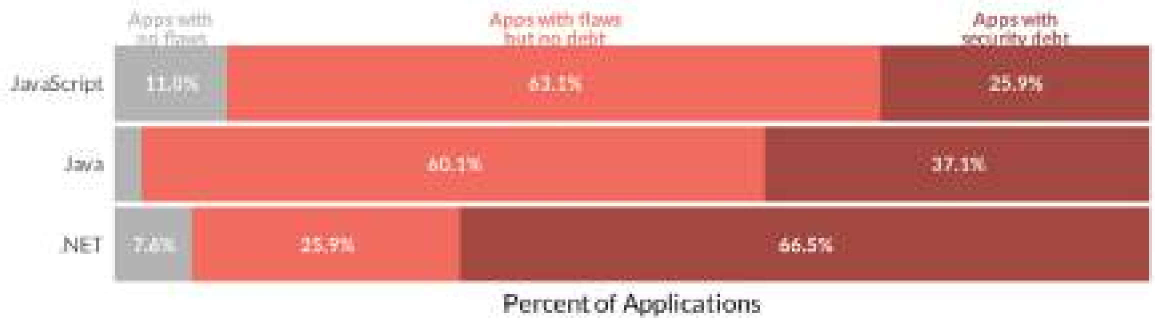


But not all debt carries the same risk, which is why we distinguish debt associated with critical security flaws in the bottom half of Figure 1. Here we see that JavaScript maintains its edge, with the lowest proportion of organizations using it exhibiting critical security debt (30%). Jumping up from there, .NET and Java swap places at 45% and 51% of organizations containing critical security debt respectively. We'll return to that notion of critical security debt at the end of this snapshot.

Another way to look at language-specific debt is at the application level in Figure 2. Notice that most of the applications (that were at least one year old) using these three languages had security issues in their last scan. Flaws constituting security debt were detected in two-thirds of .NET applications, 37% of Java apps, and 26% for Javascript.

Figure 2: Prevalence of language-specific security debt among applications older than 1 year

2



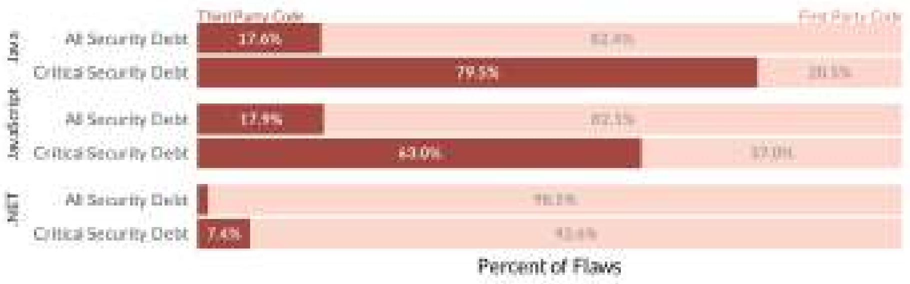
Now we have a sense of how much security debt is out there, but where does it come from? Is debt more prevalent in first-party code written by your developers or in the third-party libraries and open-source software they import into the codebase? The answer is a resounding "both," but it's not evenly distributed.

In Figure 3, we observe that the majority of security debt comes from first-party code across all of these languages. This suggests that solving your debt problem needs to incorporate training and tools to help developers write more secure code and find/fix flaws quickly.

But when we focus on *critical* security debt, things shift dramatically. Java and JavaScript have significant challenges with critical security debt in third-party code. 80% of the critical security in Java comes from third-party code and 63% of it comes from third-party code in JavaScript. .NET appears to prefer to keep things "in house" with only 7.4% of critical security debt originating in third-party code.

Figure 3: Proportion of security debt and critical debt in first-party vs. third-party code

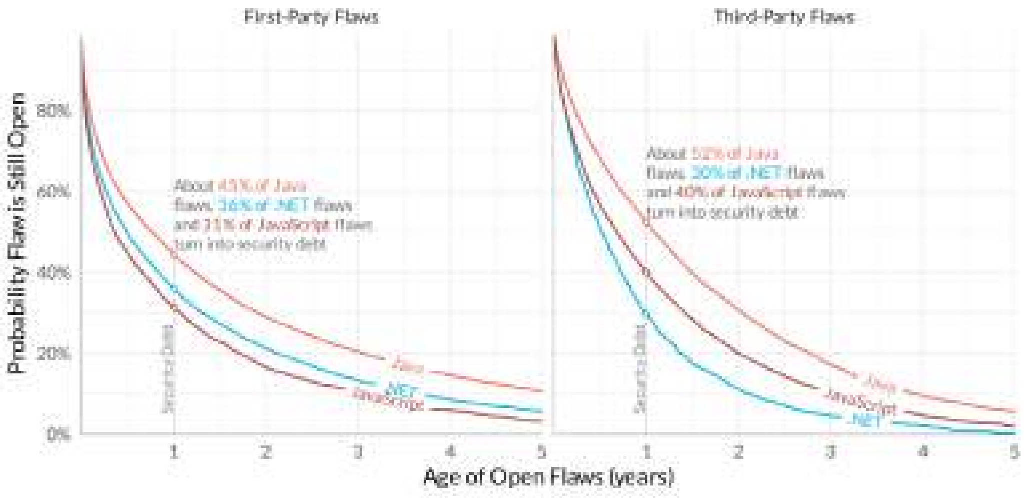
3



Now that we've looked at the prevalence of security debt in first and third-party code, let's see how long it takes to get rid of it. Analysis of flaw survival timelines across the different languages shows that Java has some catching up to do in the remediation race. About 45% of the first-party security flaws found in Java hang around for a year or more (turning into security debt).

Figure 4: Comparison of remediation timelines for first versus third-party flaws

4



That's not to say the other languages are breaking speed records. Just 36% of .NET and 31% of JavaScript first-party security flaws. And extending what we saw back in Figure 3, we expect around 52% of the third-party security flaws in Java to earn the security debt label, which is vastly higher than both JavaScript (40%) and .NET (30%). Ideally, we'd like to see the flaw survival curves in Figure 4 dive down sharply, with a very low percentage of both first and third-party flaws becoming security debt at the one-year mark. Unfortunately, that's not the reality shown here for any of these languages.

Download the full State of Software Security 2024 here:

www.veracode.com/state-software-security-2024-report