VERACODE

2024

Security Debt in the Public Sector

State of Software Security
2024 Public Sector 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 for the public sector.

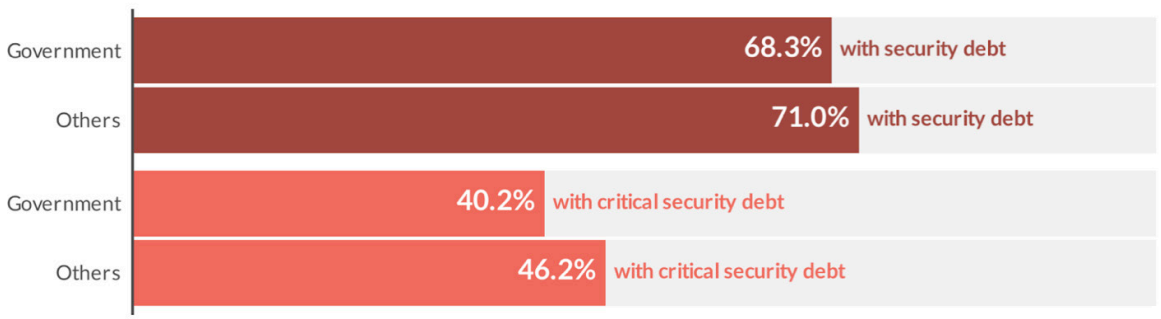
> **How risky is security debt really? How should it be tackled? And what's the best way to do it?**

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 one year. Overall, we found flaws exceeding that threshold in 71% of organizations, making security debt far from a rare phenomenon.

The public sector is roughly on par with the overall ratio, with 68% of organizations exhibiting some level of security debt across their applications. Even more concerning, 40% of public sector entities have high-severity persistent flaws that we'll classify as critical security debt. These flaws represent the highest risk to applications and thus warrant priority remediation. We'll return to that notion at the end of this snapshot.

Figure 1: Prevalence of security debt in the public sector

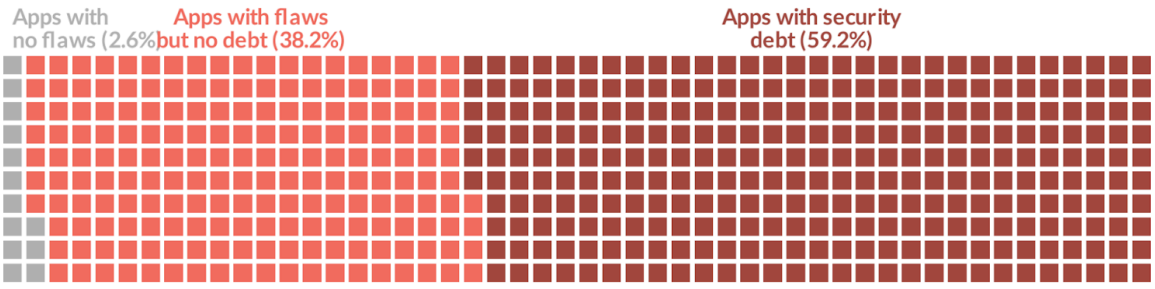
1



It's also enlightening to break down security debt at the application level in Figure 2. Here we see that well over half (59%) of all applications across the public sector have debt, which is higher (worse) than the overall rate of 42%. Furthermore, just 2.6% of applications are flaw-free, compared to 5.9% across other industries. So, while (slightly) fewer public sector organizations have security debt, they tend to accumulate more of it.

Figure 2: Prevalence of security debt across all applications greater than one year old

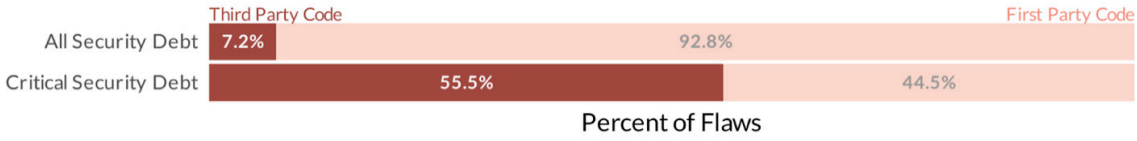
2



Does security debt exist in the code written by developers or in the third-party code they import into the codebase? The answer is a resounding “both.” Specific to the public sector, in Figure 3, we observe that most (93%) of all security debt affects first-party code. But the majority of critical security debt comes from third-party dependencies. Thus, focusing on both first and third-party code is necessary to drive down security debt within organizations.

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

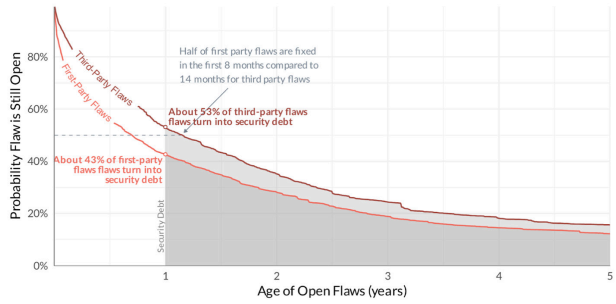
3



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 fix them. Analysis of remediation timelines in the public sector reveals that the half-life of flaws in third-party code is 14 months, compared to 8 months in first-party code. Ideally, we’d like to see the flaw remediation curves in Figure 4 dive down sharply, with a very low percentage that become security debt at the one-year mark. Unfortunately, that’s not the reality shown here.

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

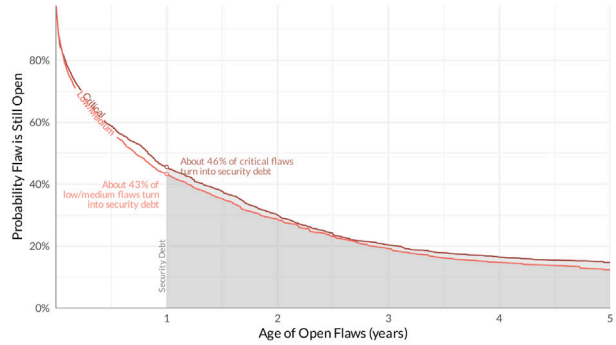
4



Speaking of how long it takes to fix things, let’s focus on the first-party flaws and the criticality of the findings. Figure 5 looks at the rate of remediation for those findings and the proportion of them expected to reach our definition of security debt within the public sector.

Figure 5: Comparison of remediation timelines for lower-severity vs. critical flaws

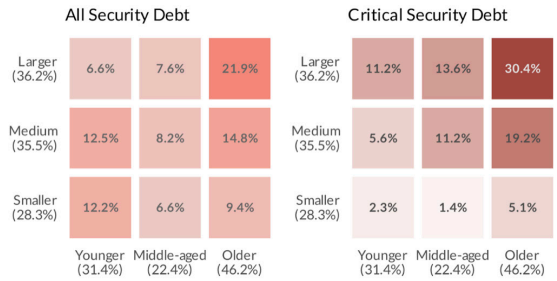
5



The codebase of most applications grows over time, so it's logical that a correlation might exist between age and the accumulation of security debt. To test this, we grouped applications into three roughly equal bins according to their relative age and size. Per Figure 6, security debt in the public sector tends to concentrate in older, larger applications. This is especially true for critical security debt. That's not terribly surprising, but it does confirm the correlation and point remediation efforts in the right direction.

Figure 6: Distribution of security debt across application age and size

6

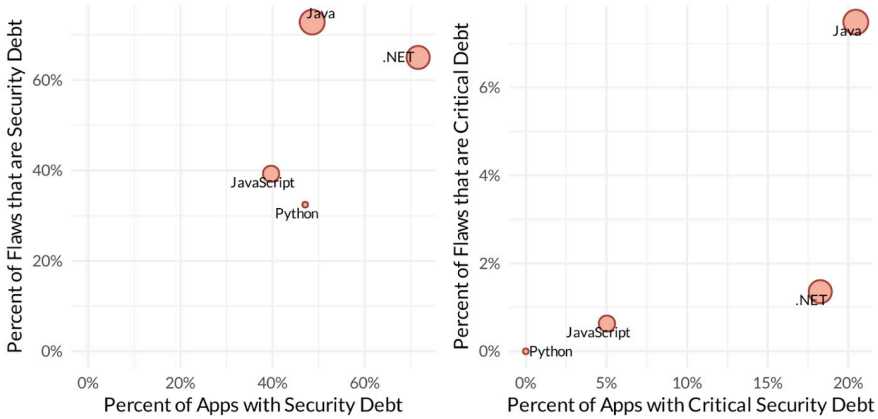


In prior SOSS versions, we've shown that the language used to develop applications has a strong bearing on security outcomes across the board (prevalence, flaw types, remediation times, etc.). It seems like a foregone conclusion, then, that languages would have a lot to do with an application's debt destiny, too.

Figure 7 compares the security debt profile for different development languages based on the most recent scans in public sector organizations. It does this by plotting the proportion of applications (x-axis) and flaws (y-axis) that exhibit any security debt (left) and critical debt (right). The size of the dot corresponds to the number of applications for each language. From this, it's clear that Java and .NET applications stand out as a huge source of debt in the public sector.

Figure 7: Prevalence of security debt by application development language

7

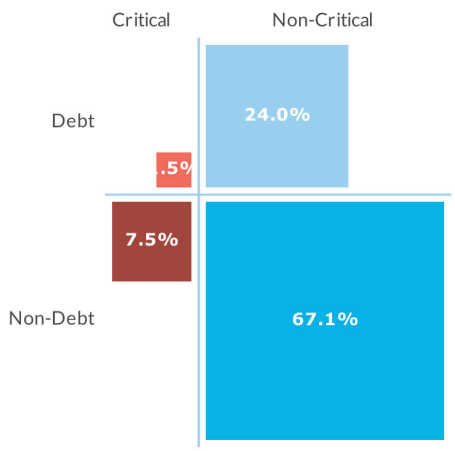


Finally, let's establish a baseline target for how many flaws organizations would need to remediate to keep critical security debt from piling up across applications. Figure 8 carves up all flaws identified across the entire codebase of the typical public sector organization into four quadrants. The horizontal separation is between critical and non-critical flaws. The vertical axis divides flaws that are currently debt vs. those that aren't (yet).

Two-thirds (67%) of all flaws in public sector organizations are neither debt nor critical in severity. We're not saying ignore them altogether (or they'll eventually become debt), but remediation of those flaws can be deferred in favor of those that represent greater risk. Instead, focus development teams on fixing the <1% of flaws that constitute critical debt. Once that's done, organizations can tackle critical flaws or non-critical debt based on their risk tolerance and capabilities.

Figure 8: Comparison of remediation timelines for lower-severity vs. critical flaws

8



The good news is that the debt reduction strategy we've described here is realistically achievable. Most organizations have the capacity to remediate all critical debt and start driving down risk in the other quadrants in Figure 8. We hope this snapshot has provided useful information to help your organization accomplish that goal.

Download the full State of Software Security 2024 here:

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

*This is based on applications that have been scanned for at least one year, since others haven't had the opportunity to accrue debt.