State of Software Security 2023

Securing the Future:

Unveiling the State of Software Security in the Public Sector

VERACODE

Each year we publish a series of cuts of the data specific to verticals or geographic regions as companion research to the State of Software Security. These cuts allow us to narrow the lens slightly and explore where we are, how we got there, and how we could do things better. It also provides an excellent chance to tease out relevant trends that get somewhat buried in the aggregate data view of the main report. This search for the signal also lets us baseline performance against peers which is particularly interesting this year for our Public Sector cut. Within this report we reference the Public Sector which includes US Federal, US Local, Education, and other Government organizations globally. Over the last year, just under 82% of applications in the public sector had at least one security flaw found in their last scan over the last 12 months, compared to 74% of the private sector organizations.

Having a look at Figure 1, the Public Sector appears to be lagging behind all other segments/verticals when considering the introduction of Any Flaws, OWASP Top 10, and CWE Top 25. Looking at the position on the chart, where higher is better, it is troubling to note the gap from the "middle of the pack" to the results we see for Public Sector. In some cases there is a 7-12% higher probability (when considering the average for all non-public sector) for these applications that a flaw will be introduced over the last 12 months. To finish our examination of Figure 1 on a more positive note, the introduction of High Severity flaws is a bright point in the overall performance. We are not sure why this category is so much better but could speculate that there is a shift in focus in these organizations towards fixing "high severity flaws."

Let's pause here though. From Figure 1, one might jump to conclusions that the Public Sector is performing poorly; however, as we tell the story you'll find out that something else is at play here and the picture is not as cut and dry as it seems. Over time, the application lifecycle is distinct and not as dire as Figure 1 might have you believe. There is room for improvement and we will suggest some possible ways to achieve that.

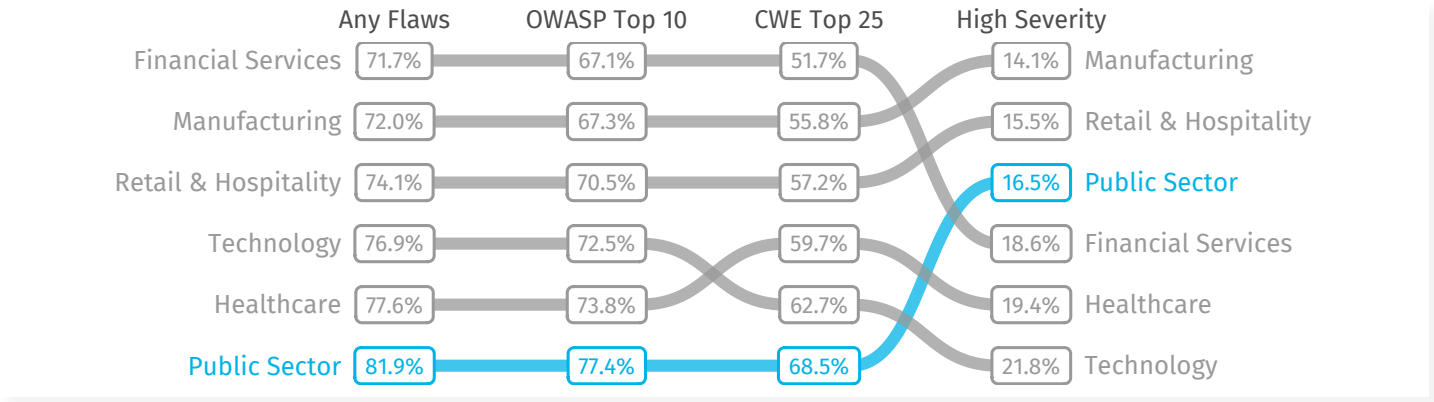

Figure 1: Percent of applications that had a flaw found in their last scan over the last 12 months, by category. Lower numbers are better.

In Figure 2 we see that 37% of applications are written in .NET, making it a clear choice for application development by Public Sector organizations. Java is the most popular language overall across all organizations except in the Public Sector. Here, Java comes in close second at almost 34%. That is where the closeness ends though, as .NET and Java are very clearly preferred over other development languages. JavaScript is only in use 11% of the time, and all other languages combined only account for 17%. We think this clear preference toward .NET and Java has some influence on the results we will examine later on.

	Java	.NET	JavaScript	Other
Public Sector	33.9%	37.4%	11.3%	17.4%
Financial Services	51.3%	24.1%	13.5%	11.2%
Technology	37.6%	28.6%	15.5%	18.2%
Manufacturing	38.3%	28.9%	15.7%	17.2%
Retail & Hospitality	42.9%	24.6%	15.9%	16.5%
Healthcare	38.2%	29.4%	13.9%	18.5%

Figure 2: Development Language Usage by Vertical

When examining the types of flaws found by scan type in Figure 3, we see that at least with static application security testing (SAST) and software composition analysis (SCA), Public Sector, shown in blue, is performing well and, in the case of SCA, considerably better than the average application, with all sectors shown in gray. When it comes to dynamic application security testing (DAST), however, the picture is not as good. We know that each scan type is better at finding different classes of flaws, but to have two out of three scan types at or better than par and then one underperforming is curious.

At least for SCA one potential answer is that .NET releases are controlled by Microsoft so there is some due diligence and control around the releases in terms of open source. Public Sector relies heavily on .NET, and .NET applications have a lower percentage of open source in them. With .NET the ratio is about 60% open source as opposed to approximately 97% open source with Java. Given the near 50/50 split between .NET and Java for the top two languages, one could speculate that differences in the ecosystem could be enough to drive down the number of flaws reported with SCA. Additional awareness, maturity, or sophistication in terms of usage of SCA could also be making the difference here specific to Public Sector and the software supply chain. It could also be the result of more restrictions or prescriptiveness in the use of open source in general, such as the introduction of a pre-defined catalog or other countermeasures to control which libraries are available for inclusion.

If we consider the United States separately, another potential contributing factor to the SCA performance versus other organizations could be the actions taken as a result of Executive Order 14028. EO 14028 is intended to direct US Federal agencies to take steps to protect the software supply chain and put a program in place to be able to deliver software bill of materials (SBOM) for information sharing, transparency, and visibility purposes.

That doesn't necessarily address a few things though. First off, despite the standout performance in SCA, a look back to Figure 1 shows that the Public Sector performed the lowest in terms of flaws over the last 12 months. A relatively lower flaw introduction rate could still yield a less than optimal application flaw percentage if the flaws that are introduced are not remediated. If the remediation rate is not as steep as it should be or the scan cadence is too choppy then this could result in the contradictory behavior we are seeing. This is only speculation, and is intended for the readers of this research to take this information back to their own teams for examination of their own programs.

Questions to ask:

How many days does it take to remediate 50% of flaws that are introduced? — target <120 days

What percentage of flaws survive three months after introduction? — target < 55%

What percentage of flaws survive two years? — target < 14%

If teams can claim to hit these targets then they will be able to state they are performing as well as the leading development teams by SoSS 2023 benchmarks. This was approximately the remediation timeline for teams that developed applications in JavaScript in the main SoSS 2023 report; however, it should be considered a performance profile and not language related. Teams that cannot achieve those numbers likely will eventually carry an increasing balance of flaws that will accumulate over time.

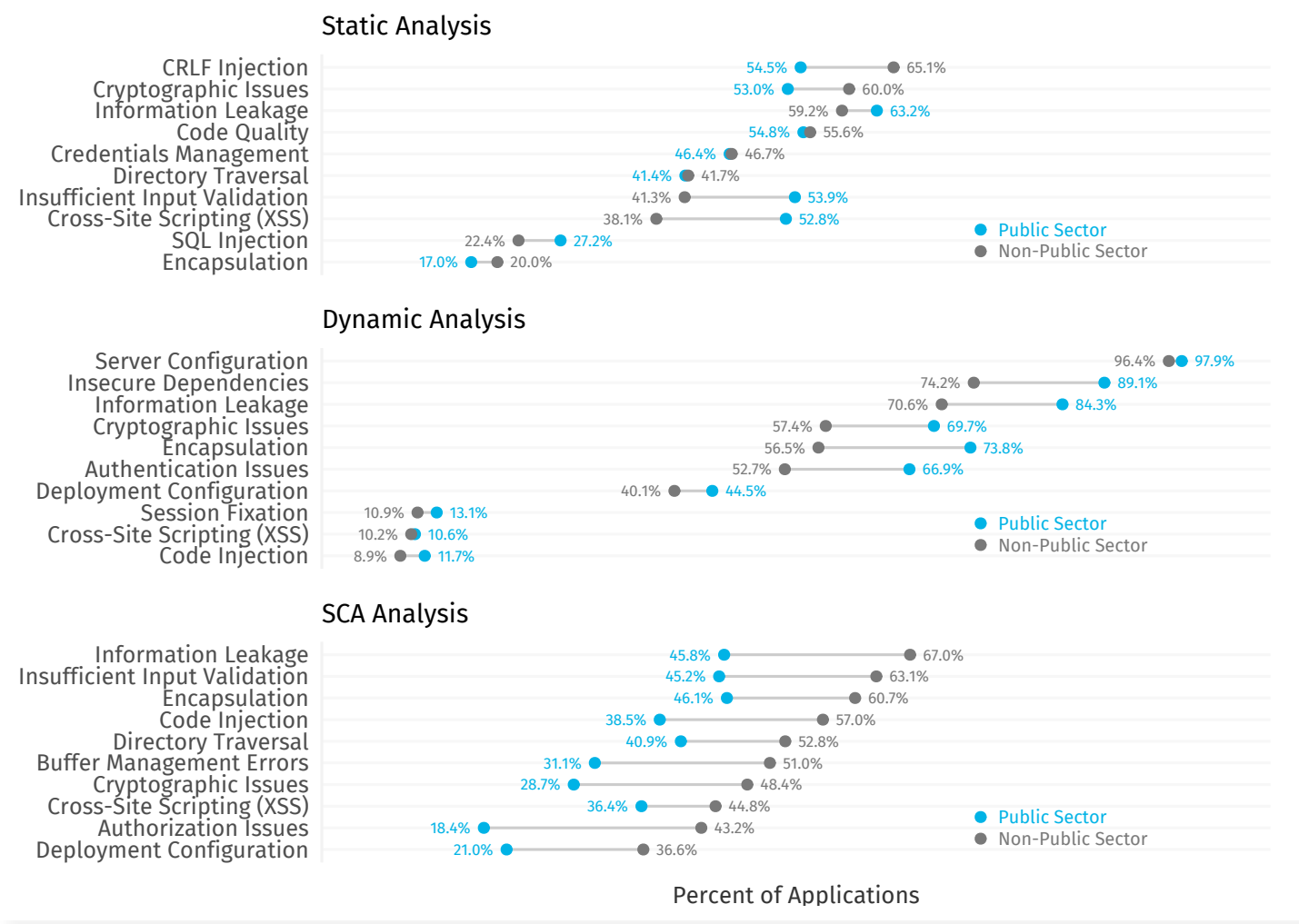

Figure 3: Top Flaw Types by Scan Type

VERACODE

Figure 4 shows that fewer applications delivered by Public Sector introduce flaws early in their lifecycle compared to other organizations. Then, interestingly enough, they don't seem to continue to improve to quite the same levels as other organizations. It is not dramatically higher but it is a distinct profile deviation from when applications should be performing their best: when they are new. Then at about the one and a half year mark, the lines cross and the applications delivered by Public Sector teams dip below the average for other sectors until about the two and a half year mark, when it crosses above for a short period. The challenge with the profile from an analysis standpoint is the individual plots jump up and down much more than the "everyone else" applications and you can see this by the shading. This is indicative of a slightly erratic performance profile and perhaps temporary bursty but unsustained focus. We smoothed out the line with a rolling five-month average to show the overall trajectory, but this jumping up and down needs to be kept in mind.

A couple more questions based on the lifecycle profile in Figure 4. Are applications tested extensively before they are delivered to get the better initial lifecycle flaw profile and then do they only get minor modifications after release, including less remediation? In other words, do these applications grow at a slower rate than everyone else? During early analysis we thought this might be the case, but we had a look at the growth rate for Public Sector applications and it looks very much in line with the general population. That means there is potential room for improvement with flaw introduction profiles possibly via more regular scanning and due diligence on secure coding practices after the initial flaws are discovered during application onboarding. However, we do have to note that applications delivered by Public Sector teams have an overall better lifecycle profile through the five year mark which tends to conflict with Figure 1. That is what led to the speculation that remediation might not be as aggressive or too bursty, which we will mention again.

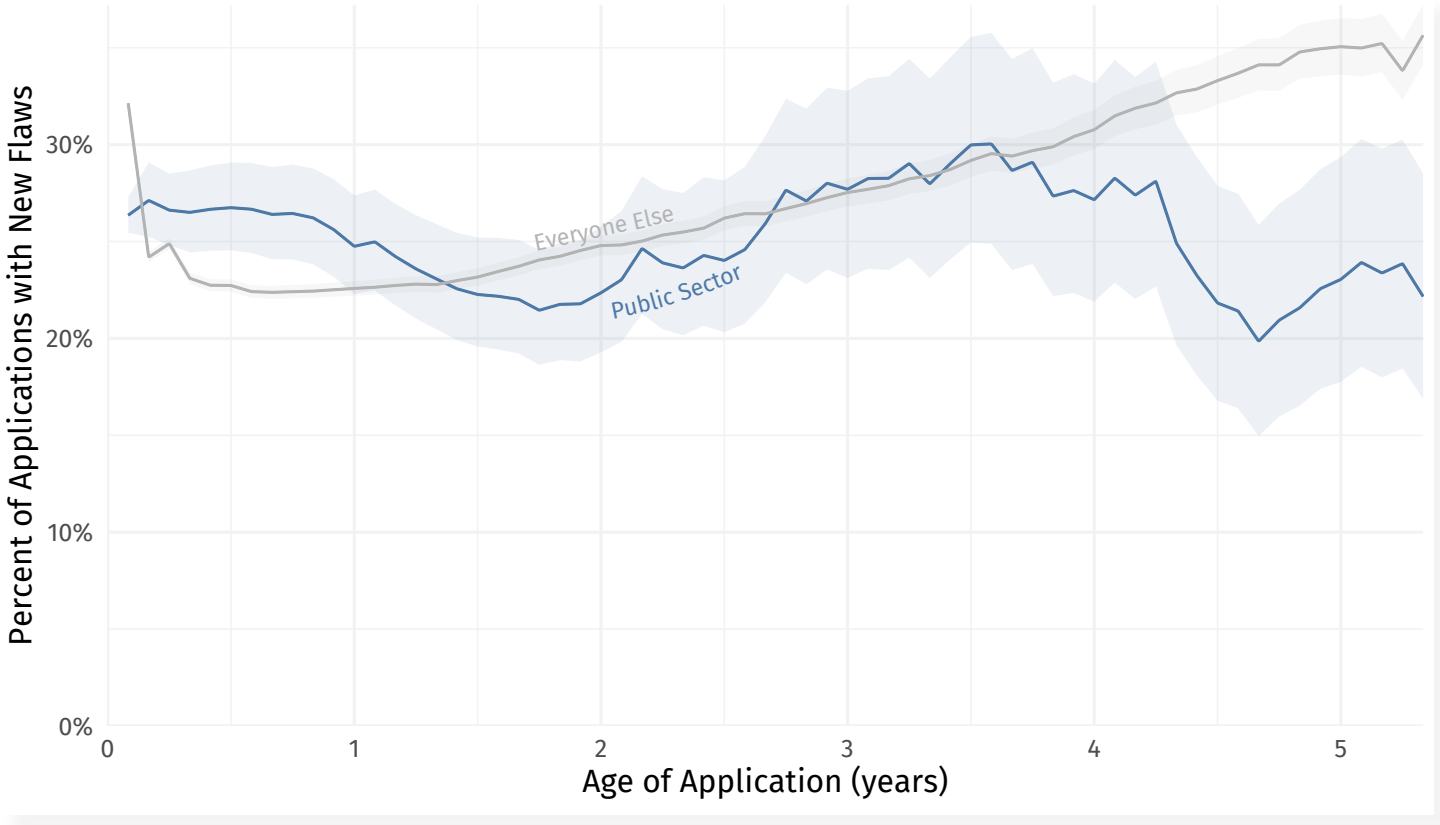

Figure 4: Flaw introduction by Age of Application

In the main SoSS 2023 report, we saw a combination of factors that influence the probability of flaw introduction, and then we examined those same factors for how many flaws were introduced (or reduced) if they were. In Figures 5 and 6 we will examine how that looks for the Public Sector. Given our examination of the lifecycle in Figure 4 we know that applications delivered by these teams have a different profile that is sometimes better and sometimes worse than applications delivered by the non-Public Sector organizations. Looking back once more to Figure 1 we know there is more to the story though. Sadly, completion of security training yielded a statistically inconclusive result for Public Sector applications so we have omitted that factor in this analysis. From the much larger "all customer" data set analyzed in SoSS 2023 we know that completion of security training has a benefit and it stands as a recommendation.

The baseline chance or probability that a flaw will be introduced is 27% in any given month.

To begin, note that the baseline chance a flaw is introduced in any given month is 27%, and the factors in Figure 5 influence that baseline probability. Initiating scanning via API (as opposed to API Scanning) is a rough measure of maturity. Teams that integrate scanning via API likely have more automation and control over the development pipeline. We see that the Public Sector development teams leveraging scanning via API reduce the chance of flaw introduction per month by 3.1 points. That is a full basis point better than non-Public Sector and given that the baseline chance of flaw introduction is 27% that's a significant reduction. In this case, launching scans via the API has a stronger correlation to reducing the probability that a flaw is introduced than for all other organizations.

We also see stronger effects from application age in Public Sector than with the general population of applications. This caused some speculation that applications in this sector are not growing as fast as in others, but this is not the case. Applications delivered by teams in the Public Sector grow in line with other sectors, but over time, as you can see in Figure 4, the probability of new flaw introduction trends lower all the way out to the five-year mark. The probability-reducing influence of age overpowers the probability-increasing influence of growth over time. The way this works is 10% growth in size increases the probability of flaw introduction by .6 points and since the average application grows 40% year over year that means it is 4x for typical application growth. That results in a 2.4 point influence on flaw introduction. Since age has a higher than typical benefit for the Public Sector than applications in the general population, that results in the downward line you see in Figure 4. This is good performance and indicates that there is a maintained focus on keeping applications secure over time - not only in the first few years. By contrast, applications in the general population are observed having a gradual increase in new flaw introduction as time goes by. Returning to Figure 5. Scans last month, App Size, and Months since last scan track closely to the general population. However, once again we see an outlier with Flaw Density which cross-validates that Public Sector applications are subject to slightly more diligence because Flaw Density usually drives up the probability of flaw introduction by half a percent more. Again, that correlates with the gradual downward track over time.

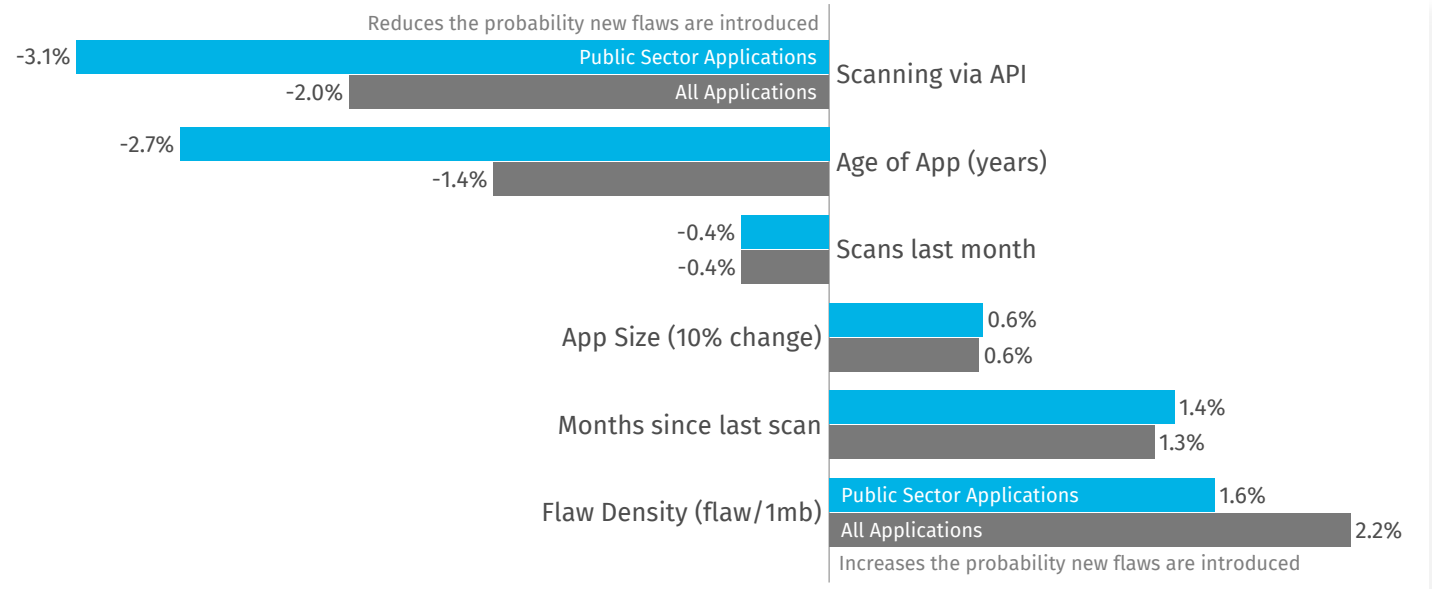

Figure 5: Factors Influencing the Probability of Flaw Introduction

Once we have adjusted our base chance with the positive and negative factors that drive the probability of flaw introduction in any given month, we see those same factors in Figure 6 and how they influence the number of flaws that are introduced. To clarify in Figure 5 those positive and negative factors influence whether any flaws are introduced. Then Figure 6 indicates how many are introduced when flaws are introduced at all. Many months may go by without any new flaws but code growth and months since the last scan invariably increases the probability that something will be introduced and then the next scan finds it.

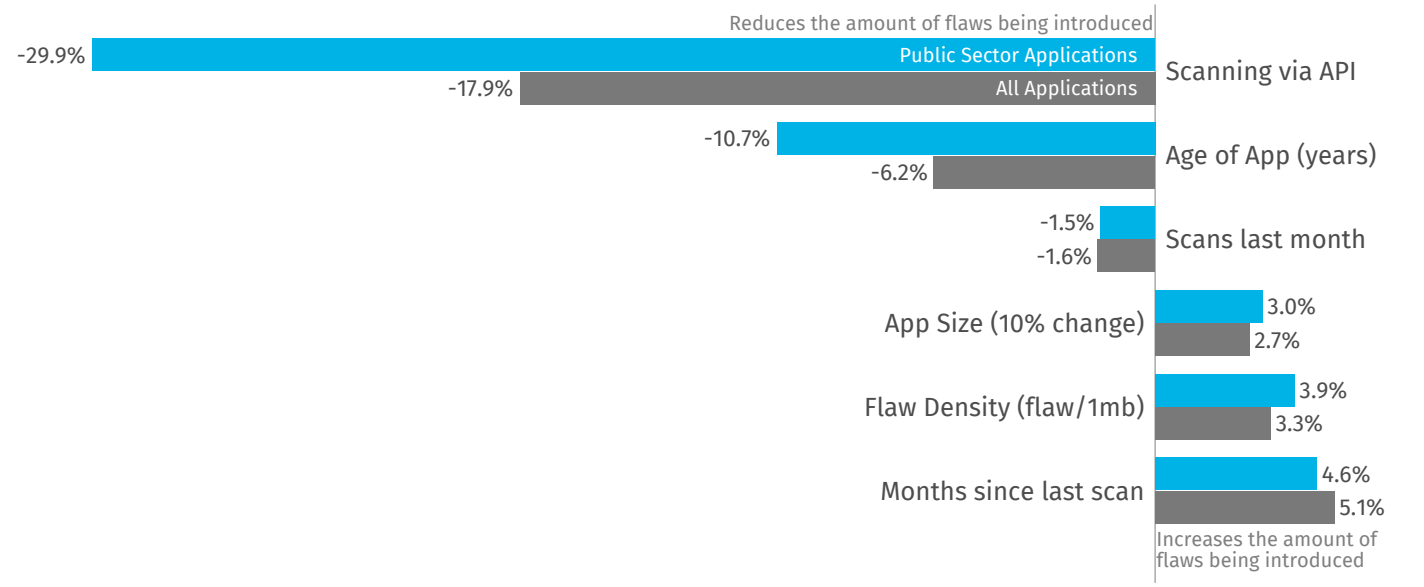

Figure 6: Factors Influencing the Amount of Flaws Introduced

Once again we see that initiating scanning via API has a profound effect, and even more so in the Public Sector with an almost 30% reduction in the number of flaws introduced. Age again reduces the number of flaws introduced at almost double the general population. The other factors track closely to the non-Public Sector metrics.

Recommendations

Fix the backlog

If we look back to Figure 1 and compare it with Figure 4 something is not right. Why do such a high percentage of applications carry flaws when so few are being introduced? A possible answer from our analysis is that flaws are not being remediated. Given that we are talking about 77% and 68% of applications carrying OWASP Top 10 and CWE Top 25 flaws respectively, it is worth looking into. The performance on High Severity does get a nod. Either way, if flaws are found, fix them. If they are not reachable flaws, then mitigate them.

Keep the scan cadence regular

In combination with the previous recommendation to fix the backlog, let's look at one way this might be made easier to do. Figure 4 looks good over time, but the shading on the top and bottom of the smoothed-out line is a bit of an area for concern. One probable cause is a scan cadence that is not as regular as it should be, so when flaws are discovered they are found in bunches. Given that age in Figure 6 really pulls down the number of flaws introduced, scanning regularly should make the workload of finding and fixing flaws more predictable. Optimistically speaking, given that the teams are doing well over time, some tactical improvements like regular scan cadence should improve the overall standings in Figure 1 so that flaws are not carried forward.

Automate

Initiating scans via API has a great correlation with reducing the chances that flaws will be introduced, and reducing the number of flaws that are introduced. Why is that? Programs that keep control over the CI/CD pipeline and leverage automation eliminate ad hoc changes that have not been vetted through the processes. This could be processes like code review, application security testing, change management, and many other steps. Allowing developers to commit outside the guardrails of the application delivery guidelines has perils. A goal for the next three years is to increase maturity in this area—increase automation, and the benefits will follow.

Add DAST to the mix

SAST is on par and SCA looks good, but what about DAST? Different types of scanning are simply better at picking up different classes of flaws. DAST will catch vulnerabilities that nefarious users interacting with the applications will be able to potentially exploit. Given the mission of Public Sector organizations to serve the public, and the modus operandi of various threat actors, it would be a very good idea to scan running applications with DAST.

VERACODE

Veracode is intelligent software security. The Veracode Software Security Platform continuously finds flaws and vulnerabilities at every stage of the modern software development lifecycle. Prompted by powerful AI trained by trillions of lines of code, Veracode customers fix flaws faster with high accuracy. Trusted by security teams, developers, and business leaders from thousands of the world's leading organizations, Veracode is the pioneer, continuing to redefine what intelligent software security means.

Learn more at www.veracode.com, on the Veracode blog and on Twitter.