2025 STATE OF SOFTWARE SECURITY

PUBLIC SECTOR SNAPSHOT

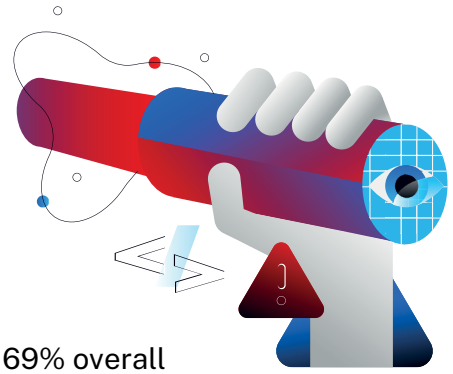
VERACODE

Discover trends about where the most software risk resides in the public sector and what metrics can be used to gauge progress against it.

Veracode recently published the 2025 State of Software Security (SOSS), marking the 15th edition of the annual report that has become a staple in the industry. This SOSS offered a new view of maturity, sharing insights drawn from analysis of 126 million security findings across 1.3 million unique applications. In particular, this edition establishes key AppSec risk metrics and benchmarks the performance of leading and lagging organizations.

While the 2025 SOSS findings are broadly applicable to all types of organizations, they didn't focus on specific industries. The role of this Snapshot is to feature insights specific to managing security debt in the Public Sector.

Finding flaws



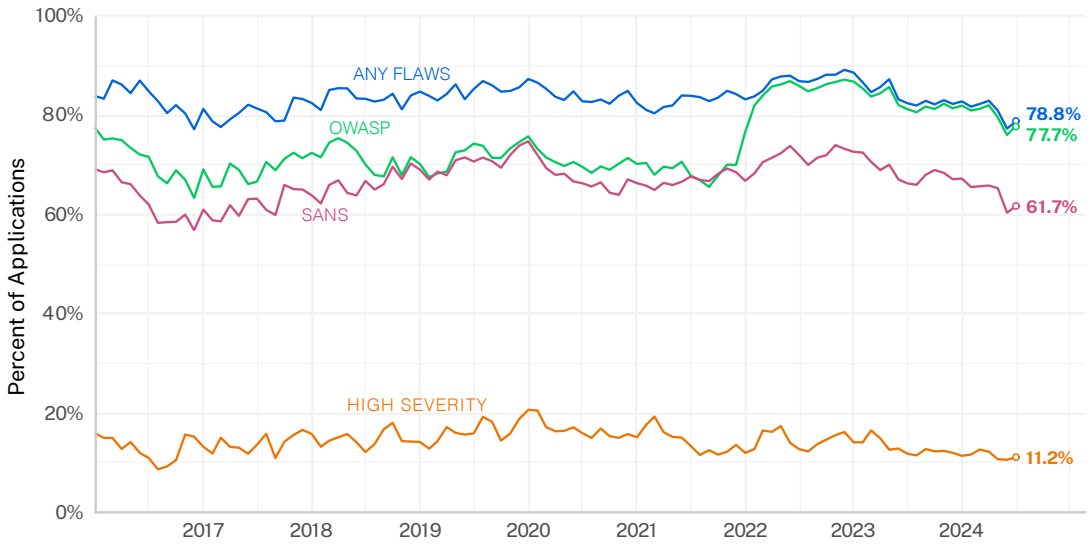
- 78% of apps have OWASP Top 10 flaws vs. 69% overall
- 11% of apps have high-severity flaws vs. 16% overall

As of their latest SAST scan, 79% of public sector applications had at least one flaw. Just under that proportion had flaws included in the OWASP Top 10 risks and 62% contained those considered most dangerous per the CWE Top 25. Thankfully, a much lower 11% exhibit high or critical severity flaws according to our own rating system.

If you're reading this Snapshot, it likely won't be surprising to see that most applications — including those in the public sector (which are organizations with the label 'Government') — have security flaws. The chart below puts a few qualifying statistics around that statement.

FIGURE 1

Prevalence of security flaws in public sector applications over time (SAST only)

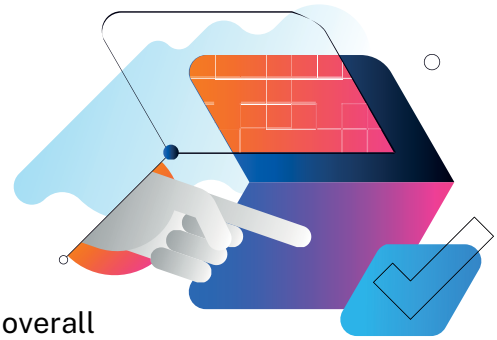


We'd love to be able to say that the prevalence of security flaws affecting public sector applications has decreased over time, but the chart clearly indicates that's not the case. The steady trend lines hint that public sector organizations are struggling to make headway in reducing AppSec risk.

One of the big reasons for this is because AppSec is not a static environment. Code is written, flaws are found, flaws are fixed, new code is written, and the cycle begins anew. Or, perhaps more accurately, it never ends. This is why it's critical to view software security as a lifecycle.

1. Most of the analysis in the 2025 SOSS as well as this Snapshot are based on a combination of static code analysis (SAST), dynamic code analysis (DAST), and software composition analysis (SCA). We use SAST only here because sufficient volume and history of data exists to support longer trends over time.

Fixing flaws



- Half-life of 315 days vs. 252 days overall
- 33% of flaws still open after 2 years vs. 28% overall

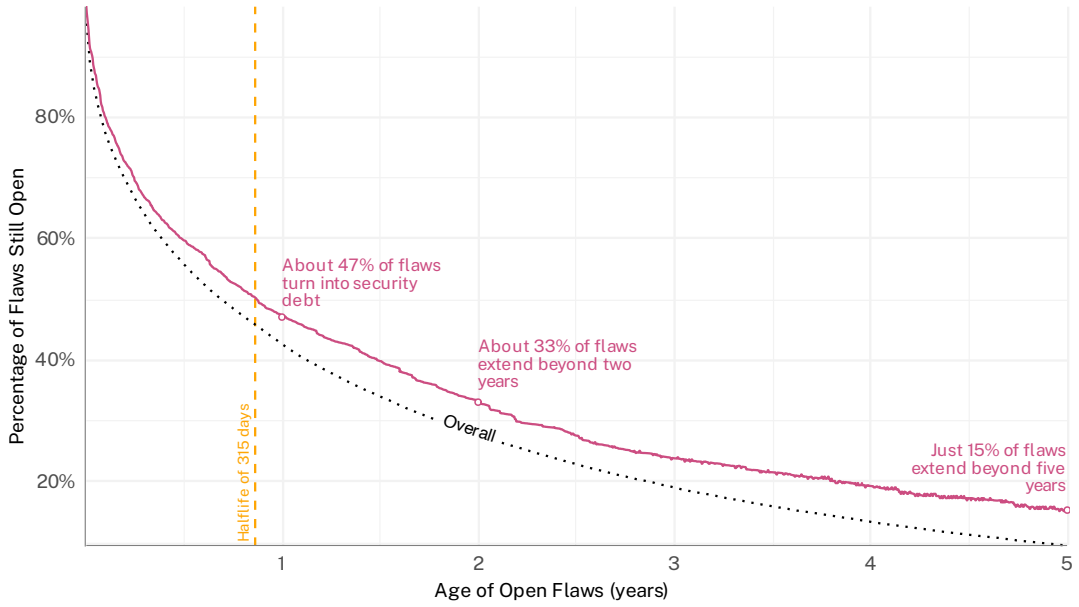
Finding flaws is easy; fixing them is where the challenge lies. There are many ways to measure the speed at which flaws are fixed across active applications. We use a statistical technique called survival analysis for this metric. A flaw's lifespan begins at discovery and ends when scans confirm that it has been fixed (which doesn't always happen).

The chart below depicts the survival curve for all flaws across public sector applications. You can determine the survival rate at any point along the curve based on where the x and y axes intersect. Half-life is a key statistic for fix speed, measuring the time it typically takes to eliminate 50% of flaws.

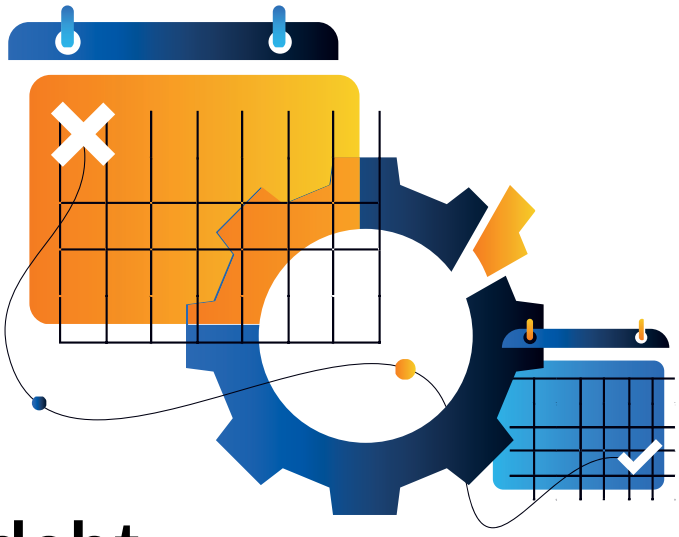
For the public sector, half-life stands at 315 days — about two months longer than the overall average. After two years, two-thirds of the flaws have been fixed, and 15% remain open for more than five years. Note how the survival curve for the public sector is consistently above the overall curve, indicating a consistently slower rate of remediation.

FIGURE 2

Public sector flaw remediation timeline based on survival analysis



The concave shape of the public sector survival curve makes it clear that the process of fixing flaws begins in earnest but tapers off over time. There are numerous reasons for this phenomenon, but the result is that applications gradually become bloated with old, unresolved flaws that we term security debt.



Managing security debt

- 78% of orgs have security debt vs. 74% overall
- 55% of orgs have critical debt vs. 50% overall

Security debt refers to flaws that remain unfixed for over a year. It's quite pervasive — about three-quarters of all organizations have accrued some level of debt. The level of security debt in the public sector is slightly above par, affecting 78% of organizations. Even more concerning, over half (55%) of government organizations have critical debt — the risky combination of highly severe and long-unresolved flaws. That's 5% higher than the overall rate.

FIGURE 3

Percentage of public sector organizations with security debt

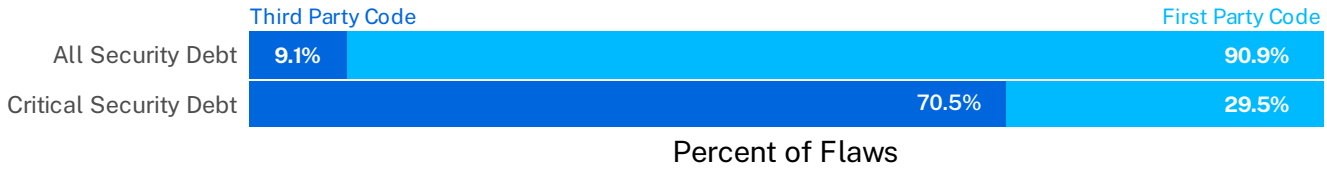
Wondering where to start to eliminate debt in your organization? Our analysis shows that debt tends to be most concentrated in older, larger applications. Unsurprisingly, there is a relationship between software security debt and broader forms of tech debt that degrade productivity, efficiency, and resilience in many organizations.

Open-source debt

Not only does debt affect the code your developers write, but it also creeps in via open-source libraries imported into your applications. Of all security debt in the public sector, less than 10% stemmed from third-party code. However, if we focus on critical security debt, that ratio exceeds 70%. Any debt-fighting strategy that doesn't extend beyond your team's own code is not one that will ultimately be successful in driving down AppSec risk.

FIGURE 3

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



It generally takes about 50% longer to fix flaws in third-party code compared to first-party software. Evaluating open-source libraries and avoiding those riddled with flaws before incorporating them into your codebase can help prevent major issues across applications.

Benchmarking performance

At the outset of this Snapshot, we promised to benchmark public sector organizations in terms of how they're managing security debt. Here's where we make good on that promise for 5 key metrics that we recommend using to measure the maturity of your AppSec program:

- Flaw prevalence: Percentage of applications with at least one unresolved security flaw
- Fix capacity: Average percentage of open flaws fixed each month
- Fix speed: Time it takes to fix half of all discovered flaws (half-life)
- Debt prevalence: Percentage of applications that have accrued debt (flaws >1y old)
- Open-source debt: Percentage of all security debt that exists in third-party code

The stats below compare the performance of "leading" (top 25th percentile) versus "lagging" (bottom 25th percentile) public sector organizations for each metric. On the whole, there's a marked difference in how leading public sector organizations are driving down risk. They have fewer flaws, they fix more of them in a timely manner, and maintain a much lower rate of security debt across their applications.

Leading Organizations	Lagging Organizations
< 33% of apps have flaws	100% of apps have flaws
Fix capacity: above 9% of flaws monthly	Fix capacity: 0.1% of flaws monthly
Fix speed: half of flaws in 3.3 months	Fix speed: half of flaws in 11.4 months
Security debt in less than 26% of apps	Security debt in more than 85% of apps
84% of apps have open-source critical debt[2]	100% of apps have open-source critical debt

2. This number is much higher than what we observe outside the public sector (15%). We suspect this is due to a comparatively low number of public sector organizations using SCA. Those actively doing SCA are likely doing it out of necessity because they heavily utilized open-source libraries, and so are more likely to discover flaws at a higher rate.

How do leading public sector organizations achieve this level of performance? More importantly, how can YOUR team join them?

It starts with having visibility into your SDLC to minimize flaws at the source. Do this through continual scanning as developers write their code before flaws go into production. This is the most cost-effective time to remediate flaws, and it's a great use case for responsible-by-design AI.

Next, correlate and contextualize findings in a single view to prioritize work to burn down security debt. Your AppSec tools are flooding you with information about what's severe, but you need a way to see what's exploitable, reachable, and urgent to help you prioritize further. This requires an open and tool-agnostic Application Security Posture Management solution.

VERACODE