State of Software Security 2023: EMEA

Securing the Future: Unveiling the State of Software Security in Europe, Middle East, and Africa

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 (SoSS). 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 in other geographies and yields an aggregate view.

Within this report we refer to a horizontal based view of organizations in Europe, Middle East, and Africa or EMEA. This means that by its nature the view is essentially an average of all industry verticals and all countries, which introduces some forcing. However, given the sample size of applications and organizations that produce them, we were thankfully only left with a few curious question marks in the data, which we will call out when we get to them.

A glance down at Figure 1 reveals that EMEA is in the middle of the pack in most areas but significantly behind the Americas regarding the proportion of applications with any flaws. In the Americas about 73% of applications carry security flaws in their last scan over the last 12 months, whereas in EMEA that number is just over 80%. APAC is performing worse, except in High Severity Flaws where EMEA drops to the bottom of the pack. The reason for the positions of the three Geographies is unclear. Some clues can be found in some of the later figures, but some conflicting data as well. Such is the state of things, but it is an interesting view to see North America so far ahead of the other Geographies. That should not be mistaken for praise though since all Geos have applications with a very high percentage of OWASP Top 10 and CWE Top 25 flaws. Note that this is a slice in time: an application's last scan in the last 12 months. Note also that there is not a count here in this view, but we'll get to that later.

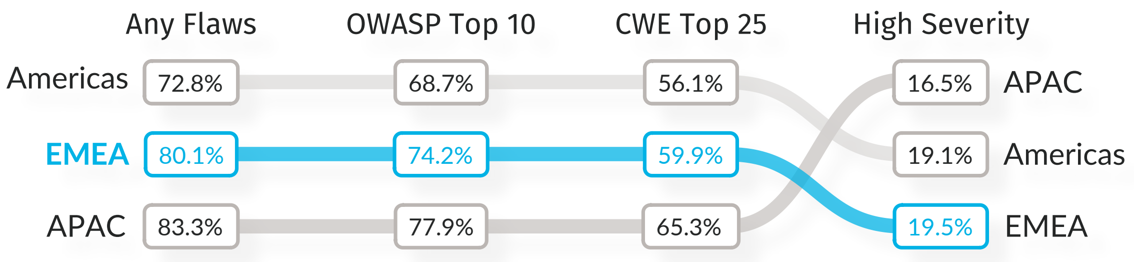

Figure 1: Percent of applications that had a flaw found in their last scan over the last 12 months, by category. Lower numbers are better.

When it comes to language preference the EMEA region is the top user of Java (see Figure 2). In some early discussions we considered that this might be one of the reasons for the higher percentages of flaws in the last scan (see Figure 1.) If we remember the Remediation Timeline in the original research (State of Software Security 2023), application teams using Java tended to remediate at a slower rate than those using .NET or JavaScript, causing flaws to hang around for a lot longer. By proxy that would leave flaws to be found. At least that was perhaps an early explanation. This theory had a hole in it though. In our Financial Services cut though, those application teams use Java overwhelmingly as their preferred language, and their Figure 1 numbers were a good 10% better in almost every category except High Severity. This means we have a potential correlation, but probably a false flag that they are directly related.

In any case we cannot shift the responsibility to the programming language in use. So the burden of responsibility for improving the numbers in Figure 1 lies with the teams and how they deliver code, maintain it, and remediate the flaws that are discovered. Sadly this statement is validated further on with Figure 4 with a look at EMEA performance for application lifecycle. To wrap up Figure 2, the other programming languages aside from Java are closely in line with other geographies and in general are within a percent of the Americas. APAC has a unique profile though with almost a full 22% in the "Other" category which includes any language not in the top 3 in other geographies and less preference for Java.

	Java	.NET	JavaScript	Other
EMEA	47.2%	25.7%	13.5%	13.6%
Americas	44.0%	26.7%	14.5%	14.7%
APAC	38.1%	23.0%	17.0%	21.9%

Figure 2: Development Language Usage by Geographical Region

Having a look at Figure 3, which shows the top flaws by scan type, we see that EMEA generally aligns to the overall flaw types and proportion of applications, except the proportion of applications with flaws reported from software composition analysis (SCA). Here being further right indicates a higher proportion of applications of flaws reported in EMEA than outside. The knee jerk reaction is to say, “Well Java is almost 95% open-source code. That must be it.” And this time you might be right. We saw a similar thing in the Financial Services cut.

Java applications are overwhelmingly (>95%) made up of 3rd party code (see SoSS version 12 Figure 6), and Financial Services and EMEA organizations are big users of Java. Given that Software Composition Analysis (SCA) picks up flaws in the composition of open-source code included in applications, the probability of finding publicly reported flaws using SCA rises commensurately with that higher percentage of open source code usage. This is not merely saying that these applications are full of flaws though. It only indicates that scans found libraries or packages containing flaws and reported them. After the report of flaws is delivered, an SCA product will also advise on available upgrades to versions of the libraries that do not contain those flaws. An interesting correlation to be sure, and one we saw again in an inverted fashion where fewer flaws were found with .NET and Public Sector. In those organizations .NET was in higher use and Java lower than in the broader market. This seems to be a strong theory and is worth testing further in the next SoSS report.

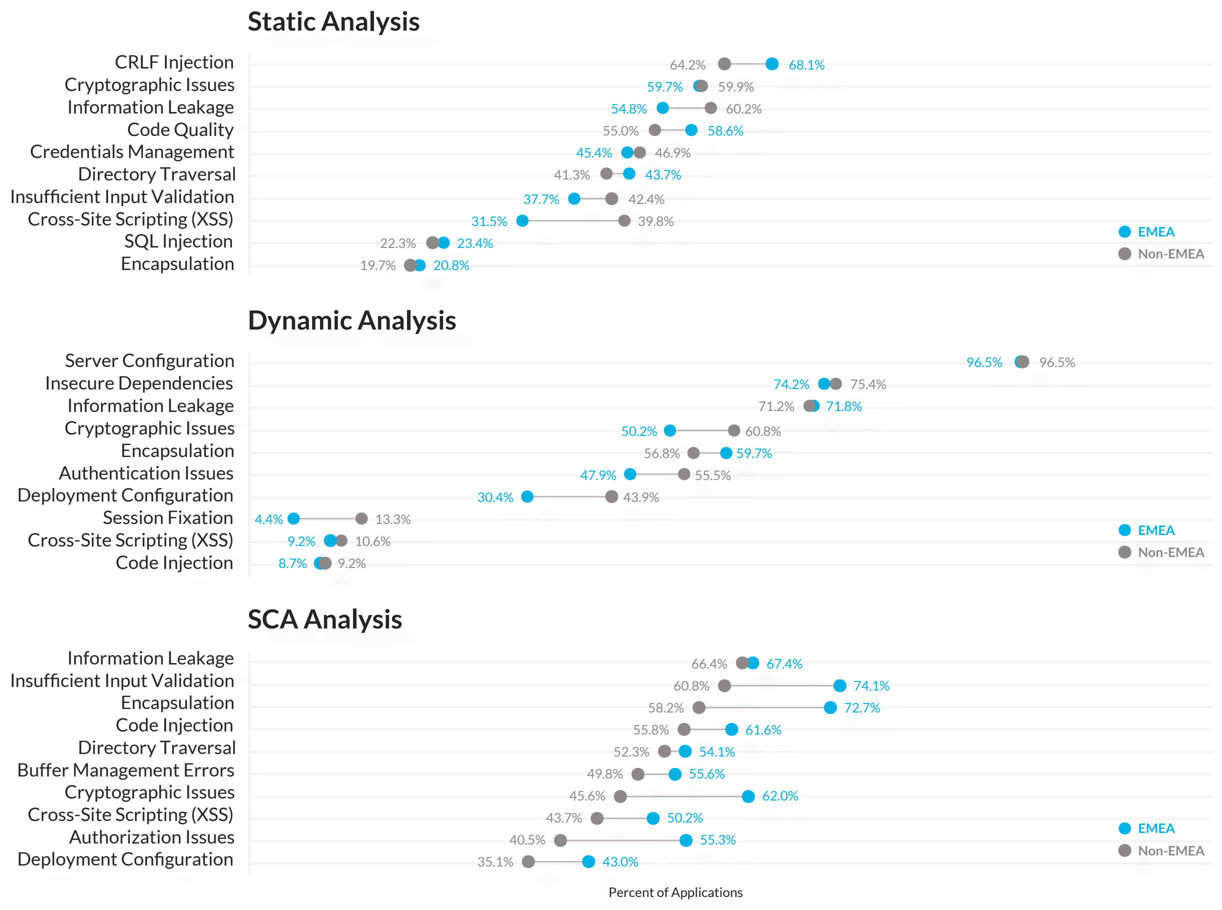

Figure 3: Top Flaw Types by Scan Type

Application lifecycle management is something that not a lot of organizations do , and as a disciplined practice even fewer. Planned obsolescence for applications is a bitter pill to swallow. It's expensive in terms of development hours and net cost to replace the functionality that was provided by an aging application. It seems that many organizations in EMEA keep updating applications but with less focus on quality, as is demonstrated by Figure 4.

As applications age, the knowledge of the inner workings tends to disperse to other teams and other projects (or it simply fades from memory). The more hands that touch an application, the more varied the methods of accomplishing those functions becomes. This can be minimized with disciplined style guidelines and code reviews. The beginning Figure 4 shows that 40% of applications in EMEA introduce new flaws. The pay down of the flaw debt is rapid though and falls in line with everyone else for about 2 years. After that, the introduction of new flaws diverges rapidly, and clearly more attention to this portion of the application lifecycle is needed. Whatever choices are made after the first few years should be examined to get back to a baseline of fewer new flaws.

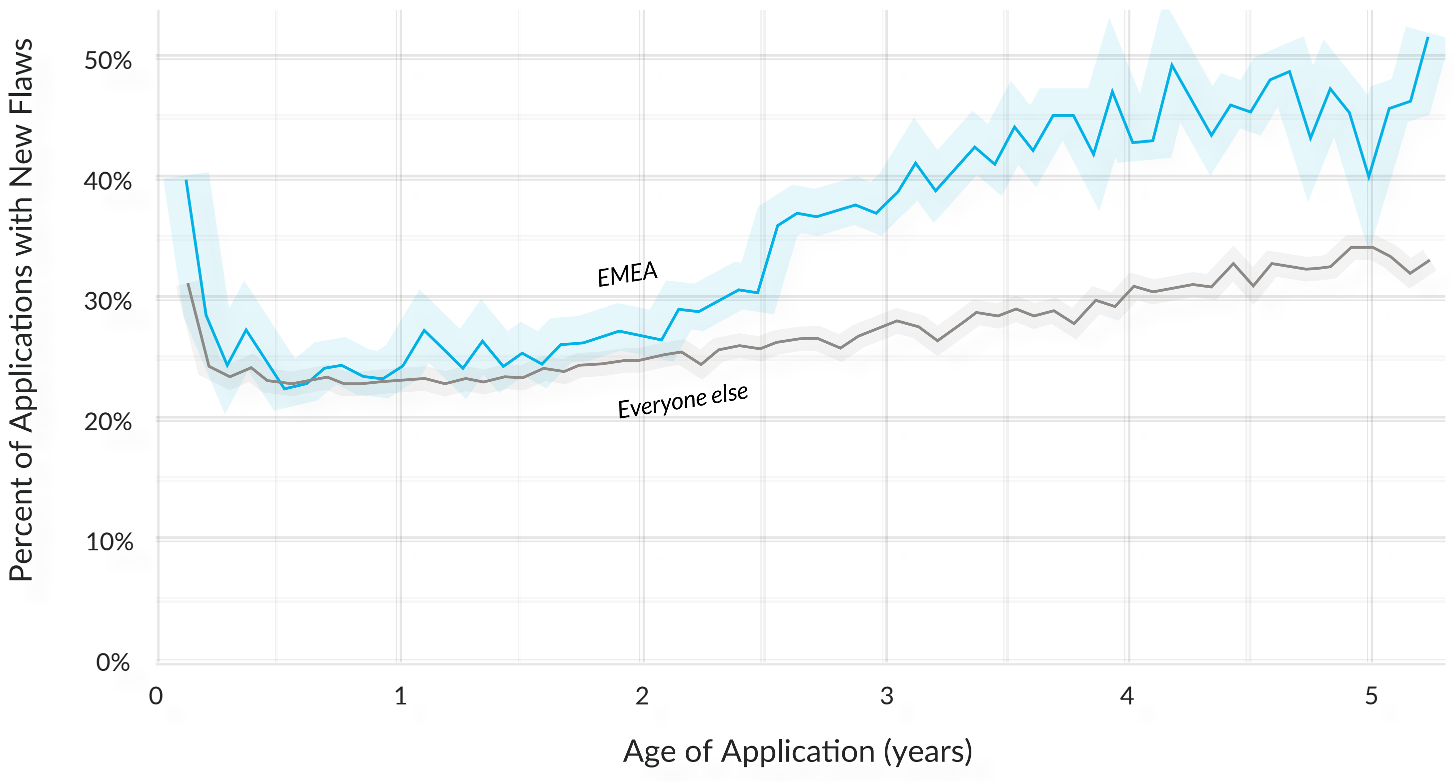

Figure 4: Application Size by Age of Application

In the main SoSS 2023 report, we saw a combination of factors that influence the probability of flaw introduction, and we examined those same factors for how many flaws were introduced (or reduced) if they were. In Figures 5 and 6 we will examine this for EMEA. Given our examination of the lifecycle in Figure 4 we know that applications delivered by these teams have a different profile that is sometimes on par with but later worse than applications delivered by organizations outside of EMEA. 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 EMEA 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 with, the baseline chance that 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 EMEA development teams leveraging scanning via API reduce the chance of flaw introduction per month by 2.3 points. That is almost on par with non-EMEA organizations launching scans "and" - "API, and has a slightly" better correlation to reducing the probability that a flaw is introduced than for all other organizations.

We also see slightly weaker effects from application age in EMEA than with the general population of applications. We double checked and applications in this region are growing as fast as in others. Applications delivered by teams in EMEA grow in line with other regions, but over time, as you can see in Figure 4, the probability of new flaw introduction trends higher all the way out to the five-year mark. This mutes the usual probability-reducing influence of age and is overpowered by 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. 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. EMEA applications curiously buck the trend and age has a lower than typical benefit than applications in the general population. That is clearly visible by the upwards line you see in Figure 4. As mentioned this performance indicates that there is a dwindling focus on keeping applications secure over time. By contrast, applications in the general population are observed having a less marked 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 EMEA applications are subject to less diligence because Flaw Density drives up the probability of flaw introduction. Again, that correlates with the upward track over time that we saw in Figure 4. We know that teams that scan regularly bring their flaw baseline down and it stays there, and this is a common recommendation for how to mature an AppSec program.

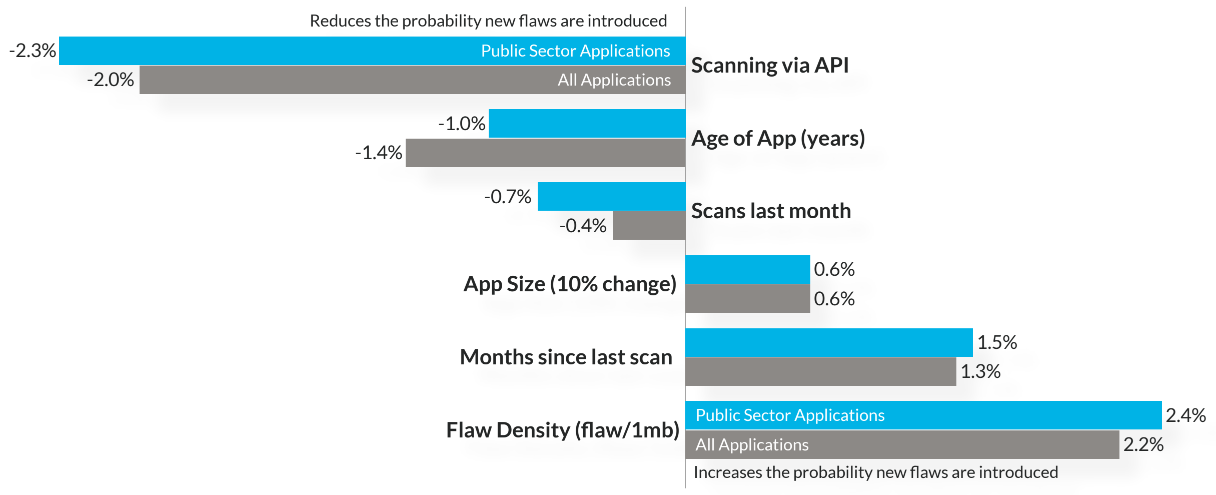

Figure 5: Factors Influencing the Probability of Flaw Introduction

If you look back at SOSS volumes 9, 10, 11, and 12 you'll see that applications that are scanned at a regular cadence fix more flaws faster than those that are only scanned periodically. Security seems to prefer agile development.

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 at all. Then Figure 6 indicates how many are introduced when flaws are introduced. 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.

Once again we see that initiating scanning via API has a profound effect, but curiously despite re-examining the data and looking for outliers it seems that EMEA applications do not benefit as much as non-EMEA applications. Sadly we have to take the data as it is, but getting control of the pipeline and launching scans via API correlate to good performance everywhere else, so it remains a recommendation.

Again we see the positive and negative sides of the cadence coin in Scans last month (good!) and Months since last scan (bad!) Age again doesn't reduce the number of flaws introduced as much and application size increases the number of flaws more than the general population. Those two together are likely due to what we see in the application lifecycle in Figure 4. Flaw Density is much more influential than non-EMEA and likely driven by the upward trend in Figure 4 as well. It bears repeating that something needs to be done to maintain focus on applications delivered by EMEA teams as they age.

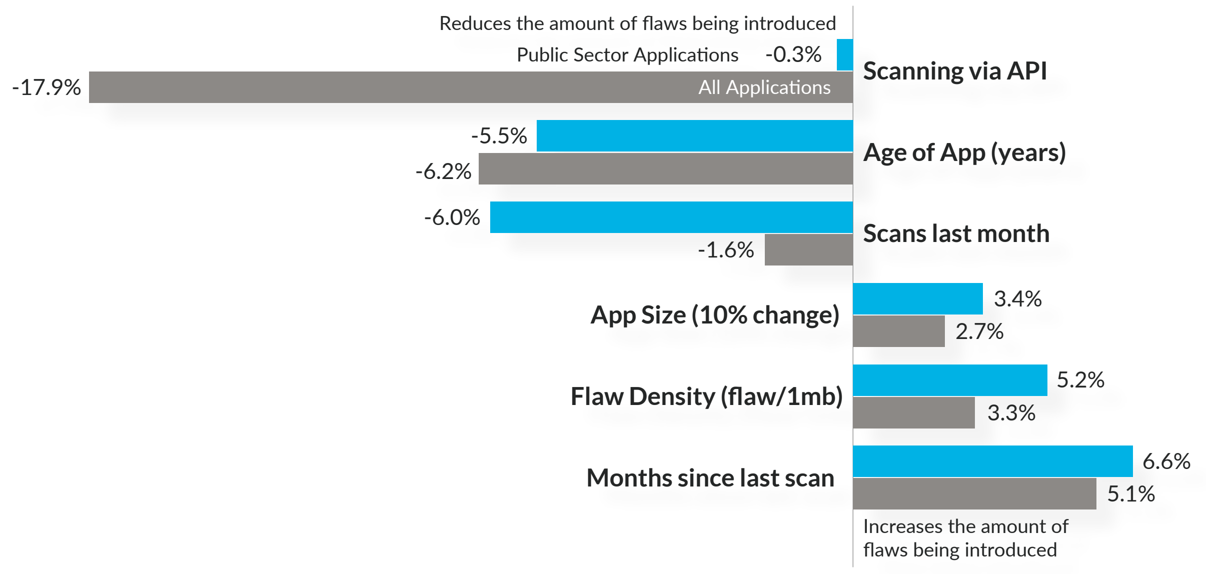

Figure 6: Factors Influencing the Amount of Flaws Introduced

Recommendations

Examine the application lifecycle

Our data shows that EMEA applications increase their new flaws at the beginning of the second year, and we cannot say for sure why that is happening in any organization. However, it is not optimal and there must be reasons why it occurs. Application delivery teams and AppSec should take a look together and build in processes such as style guidelines, better documentation, and code review if they are not in place. If they are in place, audit them for effectiveness. These things make code more maintainable over time. Finally explore lifecycle management and the idea of planned obsolescence. This may be preferable to maintaining code that has become truly unmaintainable and yields many flaws each time it is touched.

Keep the scan cadence regular

In combination with the previous two recommendations to fix the backlog and look at the application lifecycle, let's look at one way this might be made easier to do. Figure 4 looks bad over time. 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 drives up the number of flaws introduced, scanning regularly should make the workload of finding and fixing flaws more predictable. Optimistically speaking some tactical improvements like regular scan cadence should improve the overall standings in Figure 1 so that flaws are not carried forward.

Get automated

Despite our results in the data, we know that initiating scans via API has a great correlation with reducing the chances that flaws will be introduced, and then of course 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.

Write simple code yourself

We saw the SCA outlier in Figure 3, and that also occurred in our Financial Services data cut. Both EMEA and Financial Services organizations are big Java users. We plan on testing in the next SoSS research to find out if this is indeed a Java thing. For now a recommendation from our Open Source section of the State of Software Security 2023 is a safe bet. Since Java applications are overwhelmingly open source, teams need to discuss a purposeful way of when they should include relatively simple libraries that bring dependency chains of questionable value. If it is simple code, write it yourself, but don't roll your own crypto or dive into a proprietary database. Fewer dependencies by its nature should help.

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.