REPORT REPRINT

Application security shifts left in system development lifecycle

DANIEL KENNEDY

12 DEC 2018

Application security is undergoing a shift to the left in the system development lifecycle, moving backward through the testing box and into the code creation box. This necessary change has crawled forward for years, but has finally passed a tipping point.

Application security is undergoing a necessary shift left in the system development lifecycle (SDLC), moving backward through the testing box and into the code creation one. This necessary change has crawled forward for years, but according to 451 Research's recent Voice of the Enterprise Information Security Vendor Evaluation study, has finally passed a tipping point. The reasoning was always clear – fixing a security vulnerability in software is at its lowest impact in cost and effort when it is fixed shortly after creation. Otherwise, it goes through other's hands, like testers, or in the worst case is exploited by an attacker in production.

Application security testing (AST) tools always belonged in the hands of developers in addition to information security. But the speed of modern application development and the sheer number of builds and releases, facilitated by the proliferation and some level of standardization of DevOps tools, has forced the issue.

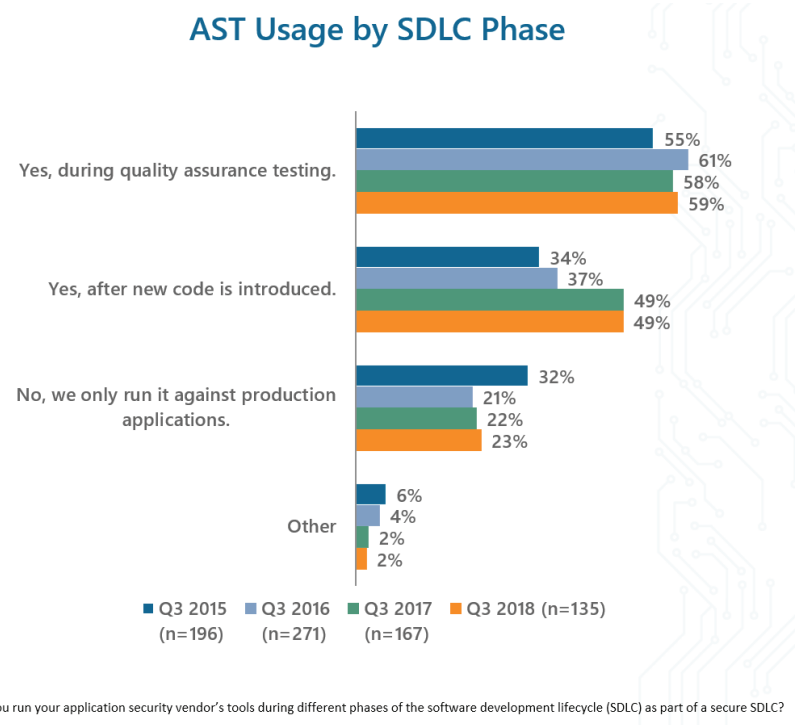
THE 451 TAKE

The efficiencies gained through DevOps tool chains and agile methodologies that stress iteration over heavy up-front planning have conspired to superheat application development delivery, and that means more releases. Application security programs that rely on the information security team to test large product releases directly before release were never particularly efficient, and are fast becoming untenable. Application security vendors who stress the role of the developer, keeping up and integrating with the methods used and supplying the right guidance quickly at code construction, while still meeting the audit needs of information security, will be in the best position to exploit a trend that is finally at fruition.

MAPPING APPLICATION SECURITY TOOL USE TO SDLC PHASE

FIGURE 1: SDLC PHASE OF APPLICATION SECURITY TOOL USE

Source: Voice of the Enterprise: Information Security, Vendor Evaluations 2018



Q13. Do you run your application security vendor's tools during different phases of the software development lifecycle (SDLC) as part of a secure SDLC?

"...we purchased a great number of licenses in the last 18 months because prior to that the information security team was doing the static code analysis instead of the developers. So in the past 18 months, we've put the tool in the developer's hands and tell them to do it themselves." – *Senior management, >10,000 employees, $5-$9.99bn, finance*

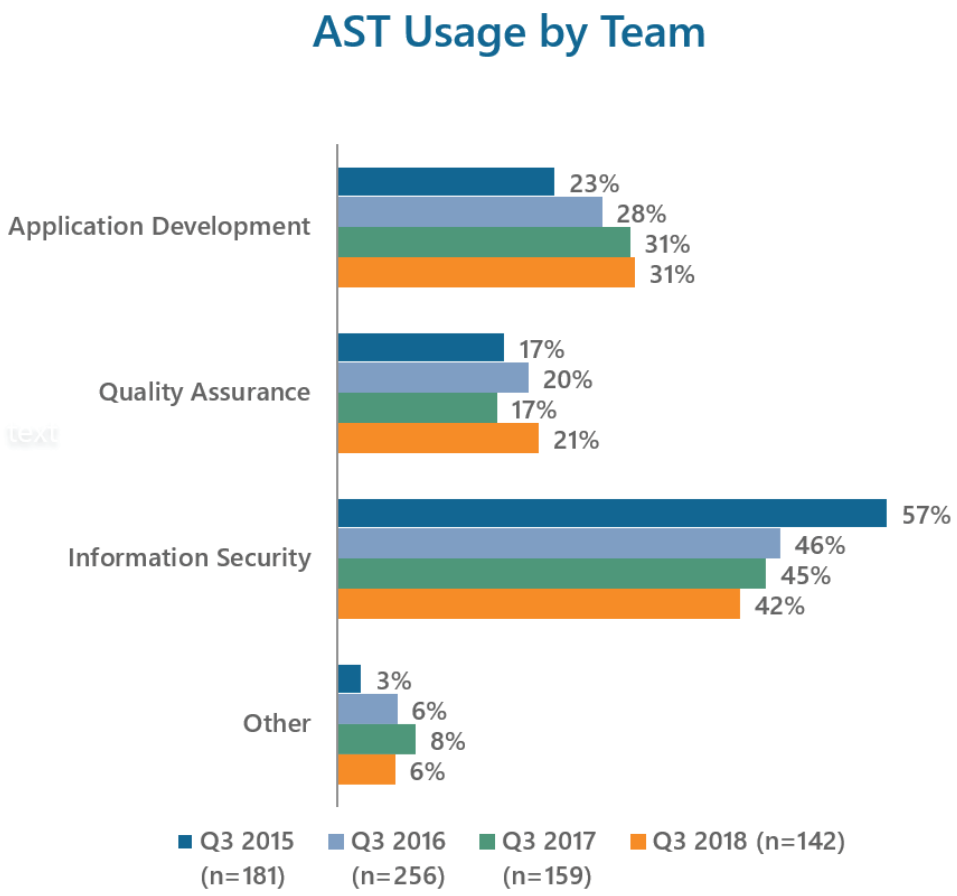
The first clear indicator is the percentage of respondents with AST that now note running what is in most cases likely static application security testing (SAST) at the point of code creation. SAST, or code/binary scanning, caught up to dynamic application security testing (DAST) in 2017 as the most common method of AST. In 2015, 34% of organizations ran these tools after new code was introduced; in 2018, it's 49%. The percentage of organizations running these tools only against production environments dropped from 32% to 23% over the same period.

The other flavor of application security that has grown the most over the same period is software composition analysis (SCA), trailing only DAST and SAST, and is an indicator of the prevalence of open source in modern application building. Many of the major application security vendors known for robust SAST offerings have built some form of SCA into their product. One-fifth, or 20%, of respondents with AST in place also noted using either interactive AST or IAST and/or runtime application self-protection (RASP), both methodologies that look at the data flows and transactions that occur as an application runs. In the case of IAST, this allows for the type of data gathering a classic DAST test might produce, and in certain scenarios more, but in an automated fashion running continuously in the background via an agent. RASP similarly looks at an application's data flows in context, but can take preventative measures to stop an attack based on what it sees.

MAPPING APPLICATION SECURITY TOOL USE TO TEAMS

FIGURE 2: TEAM ALLOCATION OF APPLICATION SECURITY TOOL USE

Source: Voice of the Enterprise: Information Security, Vendor Evaluations 2018



Q14. How is usage of application security tools allocated across the following teams in your organization?

"...you can submit code snippets and they can make recommendations on changing the code to help you, which is a significant benefit, because otherwise I've had developers come back to me, and I'll go, 'Hey, you've got a problem with buffer overflows.' And they'll go, 'Well, okay, so I just increased the size of the buffer.' Oh, my God, no. That's not how you fix it...This is an age-old problem, but...when you can go, 'Oh, yes, if you just do this with the code,' or 'So here's a library that will help you to solve this problem,' it's less painful, because the biggest impediment to application security tools is the elongation of the development timeline. Because developers, you screw with their release cycle and they lose their minds." – *Mid-level management, 1,000-9,999 employees, $100-$999.99m, information*

The second indicator is team usage of AST. In the survey, respondents are asked to allocate 100 points of usage across four teams: application development, quality assurance, information security and an 'other' category to capture usage outside of these three roles. While information security is still the largest user at 42%, that is down from 57% in 2015. At the same time, application development as a user went from 23% to 31%.

IMPLICATIONS

The implications for application security vendors and enterprise security are multiple; the first most obvious one is an increasingly complex sales cycle and evaluation process. The information security team in large organizations still has the greatest motivation for spending resources on application security and retains the role as the largest user of these tools, although the trend line on that is clear. Rather than attempting to run scans and start advising developers on their code, a difficult proposition even when the information security team has coding expertise (which they don't always have), and which can unfortunately lead to dumping generic vulnerability reports over the fence, the security team should embrace an ideal role as application and process auditors: finding the issues that have escaped the code construction process and monitoring efficacy of vulnerability identification and remediation overall. That means a combined approach between both teams in evaluating service offerings.

Another implication is the push toward integrated developer education in these tools, resolving the secure coding instruction shortfalls of many college or university computer science programs. The key for AST vendors has been tight IDE integration that provides personal development opportunities in secure coding in short, easily digestible, in-context bursts in reaction to vulnerabilities the AST tool is identifying.

Finally, integration into DevOps tools, including build tools, ticketing systems, collaboration or notification tools, and the like have become increasingly common, to avoid developers doing any context switching as problems are flagged for their attention. While stand-alone testing has its place, full application security coverage will be achieved only by testing thoughtfully built into the development process as opposed to on an ad hoc basis.