Contrast SECURITY

A Major Roadblock to Business Innovation

How Traditional AppSec Delays DevOps Release Cycles

Executive Overview

Developers are more important than ever to the success of their companies, but their time is still used inefficiently at many organizations. Security tools and processes are one source of wasted time for the development team, as the application security (AppSec) field has not advanced as rapidly as the software development discipline.

When fast-moving developers encounter outdated tools and processes, they are forced to delay releases in order to perform ineffective ceremonial exercises such as generating a scan. Software defenses that include static application security testing (SAST), dynamic application security testing (DAST), software composition analysis (SCA), and web application firewalls (WAFs) become ineffective monitoring devices that require "training" periods that take longer to learn than it takes to ship new code and reset the learning phase. It is time for a new generation of AppSec that matches the efficiency and optimized processes of methods like Agile and DevOps.

As the economy becomes increasingly digital, software developers have become more and more important to organizations. One recent study determined that access to developer talent is an even bigger factor in a company's success than access to capital.[1] Yet, the same study found that as many as 17 hours per week of developers' time is wasted because of inefficiencies—with a staggering 96% of executives believing that increasing the productivity of developers is a medium or high priority. This study predates the COVID-19 pandemic, but all indications are that this unprecedented disruption to the global economy will only accelerate the need for digital transformation at businesses of all types.[2]

In this fast-changing marketplace, developers are under constant pressure to push code even more quickly so their companies can remain competitive. The pressure comes from the very top of the organization. According to one study, 68% of organizations have a mandate from the CEO that nothing should be allowed to slow down the development process.[3]

> "While development teams are measured on speed and efficiency, workflows like security testing bring significant inefficiencies—17 hours per week for each developer.[4]

Contrast SECURITY

AppSec Practices Have Been Slow to Evolve

Of course, many such slowdowns are because of security issues. Legacy approaches to application security (AppSec) have lagged behind advances in the field of software development—particularly the move from sequential approaches like the waterfall model to more comprehensive and streamlined methods like Agile and DevOps. These newer approaches have resulted in vast increases in the speed with which applications are deployed. In fact, 27% of global developers now release new code monthly or faster; 25% say they build multiple times per day or during check-in.[5] In contrast, one study found that each vulnerability scan can delay development by as much as 164 minutes.[6]

At many organizations, the security team is accustomed to performing manual security processes across the different parts of the network. But in the highly automated and operationally efficient world of a development team, these processes can cause serious bottlenecks. And demands from the business often make such delays unacceptable. In fact, 52% of respondents in one study admitted to scaling back security measures to meet a business deadline.[7] Even when normal security processes are not bypassed, 60% of firms include security in two or fewer phases of the software development life cycle[8]—no doubt because of the delays it can cause.

While security-related delays create major headaches for development teams, they ignore security at their peril. Analysis of recent breach data finds that one-quarter of all breaches can be traced to the exploitation of web application vulnerabilities.[9] And remediating vulnerabilities becomes exponentially more expensive—and time-consuming—the later in the process they are discovered. In fact, the estimated cost of repairing a vulnerability for an application in production is 100 times that of vulnerabilities discovered in the design process.[10]

> In one test, vulnerability scans took code offline for as long as 164 minutes.[11]

Legacy Tools Do Not Do the Job

AppSec has been carried out in several ways over the years, but none of these approaches has been adequate in securing applications—individually or in combination with each other. More importantly for developers, each of these tools slows the development process by:

- Delaying code commits during security scanning
- Forcing developers to interrupt coding to deal with a barrage of security alerts
- Adding time and cost to the development process to address vulnerabilities not caught until late in the process

This white paper examines four legacy tools that together purport to cover the entire application life cycle, from the beginning of the development process to production:

- Static application security testing (SAST)
- Dynamic application security testing (DAST)
- Software composition analysis (SCA)
- Web application firewalls (WAFs)

Organizations have relied on each of these tools for a number of years. Never foolproof, they have become increasingly ineffective as the development process has accelerated, applications have become more complex, and threat actors have become more sophisticated in their tactics. And as time to market becomes increasingly crucial to organizations, these tools create interruptions that can significantly delay the process.

> 68% of organizations have a mandate from the CEO that nothing should be allowed to slow down the development process.

SAST Tools Use Lengthy Scans—and Create Excessive Noise

SAST has existed for more than a decade, and one of its advantages is that it can potentially find vulnerabilities very early in the development cycle, when remediation is quick and inexpensive. It does this by analyzing source code or binaries line by line, and this can be done before any part of an application can be executed. But SAST is a limited tool and creates or exacerbates problems for the development team.

Since SAST analyzes code line by line, one problem from the developer's perspective is that coding cannot be done during vulnerability scans. This can result in both production delays and operational inefficiency for the development team. To make matters worse, the entire code base must be rescanned every time changes are made—even when the vast majority of the code has not changed. The result can be hesitancy on the part of developers to make important adjustments on the fly because of the new, time-consuming scan that would be necessitated.

Another serious problem with SAST solutions is that they are notorious for false positives. According to research by the Open Web Application Security Project (OWASP) Benchmark Project, the average SAST tool had a nearly 23% false positive rate.[12] The result is that developers often find themselves combing through irrelevant alerts when they should be coding.

Another level of inaccuracy in SAST tools compounds the problem. Indeed, the OWASP Benchmark Project finds that the overall accuracy score for the average SAST solution is just 20%.[13] Despite the excessive number of alerts that developers must deal with, SAST tools also miss a significant number of legitimate vulnerabilities. Some of these false negatives result in risky vulnerabilities being discovered only later in the development process or even runtime, when remediation is more expensive and is more likely to create big delays in development.

The cost of fixing a software vulnerability after the design phase:

- 6x more, if found during implementation
- 15x more, if detected in testing
- 100x more, if identified in production[14]

DAST Tools Add More Delays at a Critical Juncture in the Cycle

DAST—also known as black box testing—is in many ways the mirror image of SAST, and the two are often used in conjunction with each other. This is because DAST analyzes an application by executing it, and unlike SAST, cannot identify vulnerabilities before that is possible. DAST emulates the activities of a hacker, feeding malicious data to the software as it runs. It analyzes how the application responds to the simulated attack and looks for security gaps that could be exploited. DAST can identify problems that only appear when the application is running or when a known user logs in—and thus would be missed by SAST.

But DAST tools also create significant headaches for developers. DAST regularly takes codebases offline for testing, but these delays are often more consequential at this later stage of the development cycle. Additionally, vulnerabilities discovered during this phase are costly and time-consuming to remediate and can potentially delay time to market significantly—at no fault to the development team.

DAST tools also have a very low overall accuracy score, owing to a large number of false negatives. In fact, the average DAST solution, with an 18% accuracy score, is even less accurate than the average SAST solution. As with SAST, this translates into a potential for vulnerabilities to be discovered even later in the process—during final testing or even in production. This creates major delays for the development team that could potentially cascade to future projects.

Another bottleneck for developers can come when a vulnerability identified by a DAST tool has been addressed, as there is no automated way to verify the fix. Having to do this manual verification generates yet another delay in coding while the developer chases down verification of remediation.

Finally, DAST is further hampered by the recommendations of security teams to limit error messages being sent to the scanner, the types of error messages that would inform a DAST scanner of what is happening. This results in more false positives, which produces more coding roadblocks and release delays.

> "Because legacy SAST, DAST, and PEN testing only provide a snapshot in time, they can't keep up with today's agile software development lifecycle processes."[15]

SCA Tools Create More Delays—and Have Incomplete Results

Keeping track of open-source dependencies is a priority for the security team, and SCA tools examine software to determine the origins of all components and libraries within the software. They help organizations track the sources of an application's codebase and locate new entrants in the common vulnerabilities and exposures (CVE) database as they come up. They are particularly helpful in supporting the ongoing security of applications that include a lot of open-source code.

However, SCA tools can add further frustration for the development team. For one thing, SCA tools also scan line by line, meaning further delays while code sets are taken offline for SCA scanning. Making things worse, for all the delays the scans cause, SCA tools do not provide complete information on vulnerabilities. They only examine lines of code against the CVE database, and they do not identify cases when dependencies are present but not used. This results in false positives—and wasted developer time as they chase down more extraneous alerts instead of creating code.

Finally, SCA tools do not differentiate between vulnerabilities that truly pose risk to an application and those that do not. Overall, only 0.6% of CVEs are ever exploited in the wild.[16] For the development team, a lack of risk prioritization of alerts translates into significant time dealing with extremely low-risk vulnerabilities— when they should be pushing code.

> Only 0.6% of all CVEs are ever exploited in the wild.[17]

WAFs Bring Limited Security—and a Lot of False Positives

When an application is released into production, the development team typically moves on to the next project. However, if a vulnerability is discovered—or, worse yet, exploited—after the software is in production, developers must drop what they are doing for an urgent remediation project. Vulnerabilities discovered in production are 100 times more costly to remediate than those identified during the design process—and much of that cost consists of developer time.[18] This means that current projects can be severely delayed, often with an adverse effect on the bottom line.

The foundation of security for applications in production has long been the WAF—a technology that has existed for close to two decades. Like SAST tools, WAFs are notorious for false positives. The result can be that developers are mistakenly pulled off a current project to remediate a supposed vulnerability—which winds up not being a legitimate vulnerability at all.[19]

> 25% of data breaches in the past year exploited web application vulnerabilities.[20]

Conclusion: A More Comprehensive Approach is Needed

Development teams are under constant pressure to meet business-mandated deadlines, and these time limitations are often critical to the success of the business. While SAST, DAST, SCA, and WAF tools each provide some security benefit, they result mostly in headaches for the development team. The security that these tools provide often comes at the expense of repeated delays: Codebases are taken offline for scanning and developers waste further time performing manual processes to comb through irrelevant alerts and verify remediations.

In today's rapidly evolving marketplace, delays in development cycles are unacceptable. Gaps in security protection are also out of the question from a risk management perspective. Unfortunately, as applications become more complex and development processes more agile, old-school application security processes result in both of these unthinkable outcomes. A more comprehensive, holistic, automated approach is needed.

> "Apps have become the business imperative, the key conduit to customers and the essential business enabler."[21]

Contrast SECURITY

[1] "The Developer Coefficient," Stripe, September 2018.
[2] Jim Boland, "COVID-19, Pandemics and the Accelerated Need for Digital Business Transformation," Cohen & Company, March 13, 2020.
[3] "52% of Companies Sacrifice Cybersecurity for Speed," Threat Stack, March 13, 2018.
[4] "The Developer Coefficient," Stripe, September 2018.
[5] Amy DeMartine and Jennifer Adams, "Application Security Market Will Exceed $7 Billion By 2023," Forrester, updated March 29, 2019.
[6] Michael D. Ernst, et al., "Boolean Formulas for the Static Identification of Injection Attacks in Java," University of Washington, accessed April 14, 2020.
[7] "52% of Companies Sacrifice Cybersecurity for Speed," Threat Stack, March 13, 2018.
[8] "2019 State of DevOps Report," Puppet, CircleCI, and Splunk, accessed April 9, 2020.
[9] "2019 Data Breach Investigations Report," Verizon, April 2019.
[10] Mukesh Soni, "Defect Prevention: Reducing Costs and Enhancing Quality," iSixSigma, accessed April 9, 2020.
[11] Michael D. Ernst, et al., "Boolean Formulas for the Static Identification of Injection Attacks in Java," University of Washington, accessed April 14, 2020.
[12] "Accurately Assessing AppSec With the OWASP Benchmark Project," Contrast Security, December 2016.
[13] Ibid.
[14] Mukesh Soni, "Defect Prevention: Reducing Costs and Enhancing Quality," iSixSigma, accessed April 9, 2020.
[15] Jeff Williams, "SAST, DAST, and IAST: Why the Difference Matters," Contrast Security, May 1, 2019.
[16] Roger A. Grimes, "Are Zero-day Exploits the New Norm?" CSO, February 21, 2019.
[17] Ibid.
[18] Mukesh Soni, "Defect Prevention: Reducing Costs and Enhancing Quality," iSixSigma, accessed April 9, 2020.
[19] "Perimeter Security Noise Leaves Applications Vulnerable to Attacks," Contrast Security, April 2020.
[20] "2019 Data Breach Investigations Report," Verizon, April 2019.
[21] Jo Peterson, "DevOps: The Secret to Doing More, Faster, Better and for Less Green," Channel Futures, February 23, 2018.

Contrast Security provides the industry's most modern and comprehensive Application Security Platform, removing security roadblocks inefficiencies and empowering enterprises to write and release secure application code faster. Embedding code analysis and attack prevention directly into software with instrumentation, the Contrast platform automatically detects vulnerabilities while developers write code, eliminates false positives, and provides context-specific how-to-fix guidance for easy and fast vulnerability remediation. Doing so enables application and development teams to collaborate more effectively and to innovate faster while accelerating digital transformation initiatives. This is why a growing number of the world's largest private and public sector organizations rely on Contrast to secure their applications in development and extend protection in production.

240 3rd Street
2nd Floor
Los Altos, CA 94022
Phone: 888.371.1333
Fax: 650.397.4133

Contrast SECURITY

contrastsecurity.com