opentext™

WHITE PAPER

Developer-driven AppSec: Security at the speed of DevOps

Contents

The current application security problem 3

These problems will only Continue to grow 3

Why the traditional application security practices won't succeed 4

What is developer-driven AppSec? 4

Developer-driven appSec for your organization 4

Step 1: Develop with security in mind 5

Step 2: Test early, often, and fast 5

Step 3: Leverage integrations to make application security a natural part of the lifecycle 8

Step 4: Automating security as part of the development and testing processes 9

Step 5: Consider the future 9

Getting started 10

Why OpenText? 11

90%

of security incidents are from exploits against defects in the design or code of software.

Source: 2019 Application Security Risk Report

The current application security problem

In the past 10 years, software went from being a support function of business to an innovation center, becoming the essential competitive differentiator for most businesses in every vertical and size. With this shift in the role of software, businesses are dramatically increasing the number of applications and the frequency of releases. According to TechTarget, 45 percent of organizations are releasing once per week. Additionally, the complexity of code continues to increase as developers try to meet the business demand by utilizing open source and commercial code in addition to their custom code. Sonatype's State of the Software Supply Chain report shows that on average, 80 percent of an application's code comes from open source libraries. Not only that, but their report also showed that on average, each application contains 38 known open source vulnerabilities. This has huge implications on security teams to find and manage those vulnerabilities. As a consequence, some of the notable security breaches in recent years were due to vulnerabilities in third-party code components.

With business needs in the driver's seat, applications are proliferating via websites, social media platforms, mobile and cloud applications. Furthermore, some applications are driven by marketing teams and created with third-party software. These applications are often outside the normal business processes with little or no governance.

On top of all the challenges created by an increased number of applications, increasing complexity and faster releases, regulations the GDPR and capturing customer data for purposes, have has become the norm. Having multiple instances of customer data increases the likelihood and impact of a breach. This is especially concerning because the majority of security breaches today are due to application vulnerabilities. According to our Software Security Research's 2019 Application Security Risk Report, 80% of applications contain at least one critical or high vulnerability and 90% of security incidents are from exploits against defects in the design or code of software.

These problems will only continue to grow

As time to market continues to be crucial for business, organizations adopt DevOps or similar Agile methodologies for rapid development with increasing success. All this means that if security does not become an essential part of the software lifecycle, organizations will be releasing applications with more vulnerabilities at mind blowing speed.

Application security best practices and testing should be integrated into the developer's toolchain.

Why the traditional application security practices won't succeed

In many organizations, application security is isolated to a specific team that gets involved in the final stages of development and is perceived as an inhibitor of speed. These security teams can't keep up, as development teams are staffed at an 80:1 ratio to security teams. When security vulnerabilities are found in late stages, organizations face pressure, which results in friction between teams, missed release deadlines or worse. Releases with known security defects are also being pushed to production in order to meet project timelines, in which case the business and its customers risk being exposed to attackers.

Beyond missed deadlines and team dynamics, having a reactive approach to application security is costlier to organizations. According to NIST, the cost to remediate security flaws is 30 times more expensive in production and 10 times more in testing than if they were caught in early stages of development. These issues and potential risk indicate that the only way to secure applications without compromising cost is shifting security left and take a developer-driven application security approach.

What is developer-driven AppSec?

Developer-driven AppSec is about making application security an integral part of the software lifecycle without creating additional burden for the stakeholders. Whether it's taking a DevSecOps approach, or just creating a more effective security program, the need is thinking about security from the very early stages of the lifecycle. Application security best practices and testing should be integrated into the developer's toolchain. When executed the right way, this also means that you don't need to compromise on application security in order to achieve the faster release cycles that are being driven by the market.

Developer-driven AppSec for your organization

Success with developer-driven security takes time and effort, but the biggest hurdle to overcome is the culture change needed to include security throughout the entire software development lifecycle. It's important to remove the friction between security teams and developers. Many people believe that development and security teams have competing priorities that often become the biggest barrier to the success of an application security program. Developers are usually resistant to their organization creating an AppSec program for fear of being slowed down in delivering their code. This negative mindset about security is often due to security professionals dictating rules, workflows and tools on developers instead of creating strong partnerships, common goals, and tools that seamlessly integrate with the development toolchain.

1 in 8

source downloads have known risk.

Source: Sonatype, 9th Annual State of the Software Supply Chain, 2023

Just like in DevOps, teams have to break down the silos between them, embrace transparency, and collaborate. While that's easier said than done, having executive buy-in and some security key champions within the organization can help drive this initiative. Beyond the culture change needed, here are some important steps to make your developer-driven AppSec transition successful:

Step 1: Develop with security in mind

With the developer to security specialist ratio around 80:1 ratio, empowering developers to take responsibility for their own code is a must. By finding and fixing security defects during the coding process, developers can eliminate potential security vulnerabilities before they reach testing and production, saving the organization time and money. This change in thinking requires training developers to code with security in mind and arming them with the right tools to get real-time feedback about their code. There are plenty of options for developer security training, but tools providing real-time security feedback about the code (such as the OpenText™ Application Security Assistant plugin by OpenText—which acts very similarly to a security spell checker, providing real-time security insight about the code as it's being developed) or integrated gamified developer training such as Secure Code Warrior, simplify adoption and accelerate training.

It's also important for security teams to assist in enabling developers by sharing information on known threats, providing feedback and having transparency and visibility into their work. Having development leads trained in application security and teaming up with them as security champions yields positive results. This way, dev leads bring in the security perspective early on in the development lifecycle in addition to the traditional functional and quality aspects.

Step 2: Test early, often, and fast

During the software development lifecycle, there are several approaches to follow in order to maintain the speed needed to keep up with releases today. These approaches are testing early, often, and fast.

Test early

Static Application Security Testing (SAST) identifies the root causes of security issues and helps remediate the underlying security flaws starting from the early stages of development. To maintain the speed of releases, developers need to be able to submit code quickly and easily by having the intelligence at their fingertips. OpenText™ Static Application Security Testing (Fortify SAST) leads this method because it:

- Identifies and eliminates vulnerabilities in source, binary, or byte code.
- Covers languages that developers use with support for 27 languages and counting.
- Delivers early defect detection and remediation, which leads to lower costs of remediation.

More intelligent scanningmeans DAST validation of SAST findings, and DAST tuning by SAST results.

- Reviews scan results in real-time with access to recommendations and line-of-code navigation to find vulnerabilities faster and enable collaborative auditing.
- More of a "shift left" approach—having analysis available everywhere, including developer IDE and CI/CD pipelines.

Fortify Security Assistant plugin by OpenText takes this one step further by giving developers real-time insights and rec-ommendations on code vulnerabilities as the code is being written. This not only serves as a developer's security "spell check" for common known vulnerabilities, but it enables them over time to stop making those mistakes to begin with.

Beyond static analysis, there is still the growing concern around known vulnerabilities within open source components. For almost 10 years, using known vulnerable components has been on the OWASP Top 10 list. The DevSecOps Community recently found 1 in 10 open-source component downloads contain a known security vulnerability. There has been a 71% increase in verified or suspected breaches between 2014 and 2020, and 1 in 5 organizations experienced at least one open-source breach in the last 12 months.[1]

While this is alarming, many organizations have been using software composition analysis to offset these risks. However, prioritizing open-source findings is still a major challenge with software composition analysis. Like with SAST findings, manually auditing findings is a time-consuming process that increases time to remediate for developers. Based on a report from Sonatype, organizations will spend 20 minutes on average manually researching an open-source finding and the average application contains 38 open-source issues. With most organizations having hundreds or thousands of applications, this could potentially lead to thousands of hours spent investigating open source findings that may not have any actual security impact on your application. Teams need to be able to focus on issues that are not only vulnerable, but also exploitable.

Susceptibility analysis means quickly illustrating vulnerable components that are directly or indirectly being invoked and thus exploitable or "susceptible." Being able to prioritize open source issues saves time on investigation of known issues, and even more time spent upgrading a library that has almost zero security benefit.

At OpenText, we partner with Sonatype to accomplish this, collecting methods and function signatures based on the requests that are received for Sonatype indications of known components. As Sonatype scans various open source components, OpenText understands that for any of those particular known vulnerabilities that have had updates, meaning that they have been patched, OpenText generates a signature for that function or method so that we can see that the function is actually in your own custom code and that you are using that vulnerable component of the dependency. This means developers know not just that they have the dependency on their class path but they actually used it in a way that makes them susceptible to this particular vulnerability.

[1] Sonatype, 2020 DevSecOps Community Survey

With automated static or dynamic analysis, you can efficiently identify security vulnerabilities in source code, minimizing the labor-intensive nature of security assessments.

Test often

Dynamic Application Security Testing (DAST) simulates attacks on a running web application or to identify exploitable vulnerabilities. This provides a comprehensive view of application security by focusing on what's exploitable and covering all components (server, custom code, open source, services). By integrating DAST tools into development, quality assurance and production, it can offer a continuous holistic view. OpenText™ Dynamic Application Security Testing (Fortify DAST) offers an effective solution by:

- Quickly identifying risk in existing applications.
- Automating dynamic application security testing of any technology, from development through production.
- Meeting security compliance standards with pre-configured policies and reports for major compliance regulations.
- Validating vulnerabilities in running applications, prioritizing the most critical issues for root-cause analysis.
- Crawling modern frameworks and APIs.

SAST and DAST truly complement each other. By layering dynamic analysis on top of static analysis, customers gain a valuable additional risk metric which allows them to see a more complete real-world risk picture. While it is important to identify vulnerabilities early in the SDLC using technologies like static analysis, it is critically important to create feedback loops that can identify when those findings surface in running environments via a DAST scan.

An organization that identifies findings like XSS early in the SDLC, and continues to detect those issues in production, can focus their training and development resources on addressing systemic problems.

True SAST and DAST integration means SAST and DAST tools integrate into a single developer-centric platform with a single management console. Unified vulnerability management creates feedback loops. A unified vulnerability management platform is not only critical in terms of the simplified prioritization and triage workflows that it introduces, but also in terms of the patterns that can be gleaned from the data. More intelligent scanning means DAST validation of SAST findings, and DAST tuning by SAST results.

Test fast

Interactive Application Security Testing (IAST) is a form of application security testing that combines dynamic application security testing (DAST) and runtime feedback from the tested application as the tests are being run. But even with an IAST approach, finding vulnerabilities is only one-third of the effort. The other two-thirds of the effort can often times be spent on false positive validation and remediation. Another counter argument for IAST is the fact that this testing method is likely to miss true positives because of technical limitations with this approach. As a more efficient approach, applied machine learning algorithms and audit automation can save time and auditing effort while improving accuracy for static analysis.

Audit Assistant is our machine learning technology. Offered both on premises and in the cloud, Audit Assistant leverages scan result metadata to predict and remove false positives thus reducing the time to remediate by as much as 50%. One customer saw 8,000 Java issues reduced to about 3,000 based on this technology. Our recent release further automates the process for customers by adding auto prediction at the application version to automatically request automated predictions when new issues are added.

Audit Assistant streamlines the most time-intensive phase of security testing—the auditing of scan. Audit Assistant applies extensive security knowledge and machine learning to automate the removal false positives, prioritize findings and identify the relevant security vulnerabilities to the organization. This means that after a static scan is initiated, validated scan results can be obtained in minutes and be pushed to development for fixes.

Step 3: Leverage integrations to make application security a natural part of the lifecycle

Application security must seamlessly integrate into your SDLC and CI/CD pipeline for success. By integrating into the tools your organization and developers use to develop and test your applications, you find issues early and often and fix them as part of the development testing cycles. OpenText has an integration ecosystem that is easy for developers to use, leverages your investment in current tools, and reduces friction by embedding security into your processes. OpenText application security is built into your DevOps process. DevOps speed at enterprise scale doesn't mean sacrificing security and putting your business at risk.

OpenText leverages Swagger throughout our APIs to provide documentation/API self-reference. OpenText application Security page has several projects with examples of how to leverage our various APIs to perform frequently requested tasks. The API reference is built into the products and can be accessed through the web interface of the respective products.

Faster software deployment

With automation options for static and dynamic scans and available integrations to the most popular development tools such as Visual Studio, Eclipse, and Jenkins, development teams save time and reduce friction. Integrations with defect management systems, such as JIRA or BugZilla, improve handling and remediating security issues and make sure they can be handled the same way the organization handles functional issues. This efficient approach results in faster software development and deployment that meet the business needs for speed.

Reduced risks

By shifting security to the left and covering the entire software development lifecycle in an integrated and automated way, organizations reduce their risk and associated costs because it's less costly to fix vulnerabilities earlier in the process. Fortify Security Assistant plugin and automation of security scans driven by Jenkins or Azure DevOps help the development organization adopt security testing earlier and throughout the process.

Improved return on investment

Fortify works with existing development tools to protect your existing investment and allows development teams to continue using their favorite tools. With Fortify Security Assistant plugin, for example, developers don't need to learn a different tool to run security scans on their code as it works from within their existing IDE. Or with static scan integrations, security scans are run as part of the build process and developers receive the security issues within the defect management system, without introducing any complexity to the existing tools and processes.

Step 4: Automating security as part of the development and testing processes

Automating development, processes, the provisioning of servers, and deploying applications is the key to being efficient with the DevOps initiative. Automation enables organizations to develop and release higher quality applications faster. For develop-driven application security, automation can be used in the same way with security testing in order to maintain the same quality at higher speed. Automation is about including security as part of the DevOps toolchains. This can occur in the IDE while coding, at the commit, build, and testing phases.This is a major emphasis of every AppSec program. By automating security tests, you can create and run automated security tests just like you would unit tests or integration tests.

With automated static or dynamic analysis, you can efficiently identify security vulnerabilities in source code, minimizing the labor-intensive nature of security assessments. Having an automated analysis of code reduces not only the code review, security assessment and testing times, but it leads to reduced costs in remediation by finding vulnerabilities earlier

Step 5: Consider the future

With the ongoing shift where modern development is more dynamic than ever, with increased velocity and complexity, there is continued migration to APIs, microservices, IaC, and more. Ensuring the security of this changing landscape will become more and more crucial in the coming months and years.

Getting Started

Developer-driven application security, integrated throughout the entire software development lifecycle creates measurably reduced risk and controlled processes, which ultimately results in reduced costs, improved time to market and optimized effort. Having a clear path to integrated and automated application security with measurable KPIs, will increase your organization's opportunity to succeed. Application security provides returns that are easier to demonstrate compared to other cybersecurity investments. Demonstration of the progress made and the return on investment will guarantee continued investment in application security.

Here are a few important considerations when building the roadmap for that journey:

- Identify your champion(s) for application security.
- Develop your strategy and main processes before implementing.
    - Define the initial scope and key metrics, such as: Which applications and development teams to start with
    - Whether to use SAST, DAST, or both
    - Which integrations to leverage
    - Whether to use application security tools on premises, on demand or a hybrid approach
    - What are the expected improvements in 12 months compared to the baseline.
- Find the right tools for your organization.

With everything, measuring your success is crucial. Proper KPIs allows your organization to not only effectively measure their security posture, but to justify spend and continued investment into your security program. KPIs should align with business/program goals. However, here are a few to consider:

- **Weighted risk trend**—A business-based representation of risk from vetted web application security defects over a specified time period, or repeated iterations of application development.
- **Security defect remediation window**—The length of time from when a vetted web application security defect is identified until it is verified closed. Can be referenced as Mean Time to Remediation (MTTR).
- **Rate of security defect recurrence**—The rate, over time, at which previously closed web application security defects are re-introduced into a given application, organization, or other logical unit.
- **Security to quality defect ratio**—The ratio of security defects to the total number of software quality defects being generated (functional + performance + security).

Why OpenText?

People, process and technology are the essential components of developer-driven application security. OpenText has the experience and the resources with the technology, people and processes via OpenText™ Core Application Security (Fortify) to help you every step of the way.

OpenText provides a flexible end-to-end application security solution with on-premises, on-demand, and hybrid models. With measurable benefits such as 30x faster time to market, 95% fewer positives, 10-15 times faster scans, 10 times faster remediation and 2 times more vulnerabilities found, OpenText continues to be the industry leader in AppSec tools.

Choose OpenText for:

Easy startup: You can get started in a day with OpenText Core Application Security.

Intuitive integration to existing processes: OpenText application security easily integrates with what your developers use and love, making security a seamless addition to their existing tools and processes.

Fast automation and scale: Most OpenText scans complete in minutes and you can get machine assisted audit results in minutes for raw scan results. Automated scans can be initiated as part of code check-ins, commit, builds, releases or other components of the CI/CD pipeline. OpenText customers can scale easily on premises using centralized scanning techniques, using OpenText Core Application Security, or taking a hybrid approach.

Accurate findings and coverage in programming languages: OpenText customers report more true positives (more validated findings) and fewer false positives (less noise) compared to other products. OpenText also supports more than 27 programming languages—the broadest coverage available.

Continued industry recognition: OpenText (formerly Fortify) has been recognized as an application security leader over the past 15 years, including being recognized as a leader in the Gartner Magic Quadrant for Application Security for the eighth straight year. OpenText is trusted by the top companies in multiple verticals around the world.

Build secure software fast using OpenTextwith these key features:

- **Security Assistant plugin** provides real-time-as-you-type security analysis on code. Fix each issue with confidence knowing that only high confidence issues are flagged.
- **GitHub Actions and GitLab CI templates** allows integrating and automating Static Application Security Testing (SAST) into your CI/CD pipeline workflows.

Resources

Contect with us

opentext.com
LinkedIn ›
X ›

- **Susceptibility Analysis** enables developers or security professionals to check whether someone has invoked a vulnerability in your custom code. More importantly, they can see whether attacker-controlled input reaches the code's function.
- **Speed** Dial in OpenText Static Application Security Testing gives developers more control of the depth and speed of their static scans.
- **Commit Scan** gives developers automated, light-weight scans into their workflow—integrating static testing into the Git commit process, providing them immediate feedback on the code that is being checked in for GitHub, GitLab and Bitbucket.
- **Fortify Audit Assistant** minimizes auditor workload with machine learning to identify the vulnerabilities from OpenText Static Application Security Testing results. This reduces the number of issues that need deep manual examination.
- **Smart View** in Audit Workbench helps developers quickly understand how multiple issues are related from a data flow perspective, with the ability to sort security issues and then fix issues at the most efficient point.