opentext™ | Cybersecurity

Position Paper

2022 AppSec Trend Report

The application security industry continues to evolve at pace as organizations recognize that software security risks need to balance with business imperatives that accelerate digital innovation.

Table of Contents

Top Application Security Trends

Application security continues to evolve from shifting left to shifting everywhere as we move further into a cloud-driven era. Our list for 2022 is a mix of larger trends and interesting new innovations. What would you add to the list?

98% of all code bases relied on open source components.

Introduction

In any annual trend list, there should be few surprises. After all, most trends are a continuation of what was important just a month or year ago. At OpenText™, we have a holistic AppSec vision that is based on being excellent on foundational elements. This includes broad and accurate language coverage; an integration ecosystem that allows minimum friction into the existing tools our customers use and love; and an end-to-end application security platform that takes into account that not every organization is the same.

The application security industry continues to evolve at pace as organizations recognize that software security risks need to balance with business imperatives that accelerate the speed of digital innovation. While this isn't new, the pace of technology transformation (encompassing an explosion of APIs, microservices, IAC innovation, and cloud technology mapped to the ever-increasing demand for faster time to market) is accelerating. Organizations are continually pushing boundaries while recognizing that the speed of AST delivery can't be traded for the depth and quality of code security analysis.

Many things on this list reflect what our customers are asking us to focus on and deliver—and most of these topics deserve more than a few paragraphs. We'd love to hear what trends or details you would have included.

With that, here's our list of key application security trends for 2022.

Securing the Software Supply Chain

In recent years, the severity and frequency of software supply chain attacks have increased significantly. Utilizing open-source components to accelerate the development process has proven to have great advantages, which is why a staggering 98% of all code bases rely on them. However, supply chains have many blind spots or cracks that attackers can take advantage of.

Some of the most recent software supply chain attacks, Log4J and SolarWinds, received wide press coverage, causing government and businesses to respond by scrutinizing their supply chains and putting in place the required processes to protect against risk. However, there are many emerging threats beyond software composition analysis, such as insider threat analysis (malicious code injection), insecure compilation (Trojan source), and Hacker Level Insights (third-party client-side JavaScript downloaded and executed at runtime in the browser).

For example, normal vulnerable components can create an exposure as soon as you put that software into production. Traditional composition analysis, which is done after development but before deployment, is an effective defense in this scenario. However, problems with malware components can do a lot of harm on the developer workstation itself. Software composition analysis isn't an effective defense for such an attack. You'll need traditional anti-malware software instead.

Gartner states that by 2023, over 50% of B2B transactions will be performed through real-time APIs, versus traditional approaches.

Future supply chain security will move beyond CVE scanning of the software you consume. It will encompass attack vectors such as malicious code injected in the source code you develop; the integrity of the code as it moves through the SDLC; the infrastructure driving deployment and operation; and the range of third-party code, components, and interfaces that your software interacts with at runtime. Equally important will be the ability to proactively find/select the best/most secure open source code for whatever application you build.

API Security Needs Are Growing Ever Larger

APIs are the most rapidly growing attack surface, but they still aren't widely understood and can be overlooked by developers and application security managers.

Modern cloud-native apps typically employ a distributed architecture, services/microservices, and serverless functions. These components communicate with each other, end users, and APIs, creating the need to assess security at the component and system levels. At the component level, interservice communication can use a variety of protocols, ranging from HTTP to SOAP to gRPC. At the system level, an API gateway is typically used to consolidate individual service APIs into a unified business app API, based on HTTP (usually REST). In recent years however, there has been an increasing popularity in GraphQL, the Facebook-created language that was released to the community in 2015.

API testing and discovery is a multi-step process. The first step in securing APIs is to incorporate SAST into the DevSecOps pipeline for each independent component. Then, API security incorporates DAST scanning at both the component- and system-level APIs, where HTTP is utilized. The next step is attack-surface discovery, which means providing the endpoints and parameters that constitute the API attack surface (the "what"). In addition to the "what," proper discovery also incorporates how the API is used (the "how"), which is important for the business logic workflows and sometimes complex authentication at the system level of the API gateway.

AppSec Is Evolving from Shift-Left to Shift Everywhere

Shift-Left has affected not only *where* in the SDLC application testing and security is being implemented, but has also had a profound impact on *who* is responsible for security testing. Developers are increasingly becoming the primary drivers when it comes to the purchase and implementation of AppSec testing.

"When increasing the speed and frequency of scans and prioritizing SCA tickets, we found enterprises that tightly integrate security testing within their CI/CD pipeline fix 91.4 percent of new issues."

The reality is that business usually trumps security. Developers are incentivized to deliver functionality with as few bugs as possible, as quickly as possible. So, the trick is figuring out how to insert security into the developer pipeline to enable developers to fix vulnerabilities without slowing them down.

Seamless integration into every stage of the SDLC is continuing to become the norm for AppSec tools. AppSec teams continue to have less influence when it comes to tooling in the DevOps toolchain. As development organizations pushed back, many commercial vendors started to offer hyper-convenient scanning. Early offerings resulted in tools that found only a fraction of the vulnerability issues of a more robust AppSec tool, but the convenience and cost savings helped organizations check the compliance box.

The tug-of-war between convenience and robustness has pushed the entire AppSec industry toward tighter integrations throughout the software development lifecycle. As top-tier AppSec tools become "seamlessly" integrated into the CI/CD pipeline, we're seeing the "shift left" mentality become a reality in organizations with mature AppSec programs. In fact, the "shift left" pendulum is swinging to "shift everywhere." It's really about finding the right tool for the right job, for better defense in depth.

Security has unequivocally become a critical component in DevSecOps. As vendors and tools mature, the integration and enabling experience is becoming table stakes. Quality of results and the enablement of fixing/reducing risks efficiently will once again matter more than just a quick scan/check of the box.

Cloud-Native AppSec

There's a broad IT industry trend towards the cloud, with organizations being in various stages. The modern software stack includes cloud-native elements of the architecture. CNFC defines cloud-native as "technologies [that] empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach."

AWS, Azure, and GCP dominate the market. Each offers similar functionality and SDKs for interacting with the cloud infrastructure components, along with a further level of abstraction with serverless functions. While cloud computing offloads many tasks to the cloud provider, the end user organization retains responsibility for securing the data that goes into the cloud. This affects AppSec in many ways. One of these is that there are specific AppSec requirements for apps running in the cloud.

More than half (57%) of organizations use three or more cloud platforms.

What these are depends on how the application is architected. When putting an application in the cloud, the simplest approach would be to "lift-and-shift," running on cloud infra. Few things change. Another approach is to go "cloud-native," leveraging cloud technology to the maximum extent possible. Then, many things change. This is a continuum, rather than a binary choice. Bigger organizations will have large application portfolios that are spread across this continuum.

In the cloud-native case, we have identified the following specific aspects of AppSec:

- **Use of cloud SDKs.** Example: Instead of using standard Java JDBC to connect to MySQL, you might use the AWS Java SDK to connect to S3. This has its own good and bad practices and risks. This is a huge area, given { cloud providers } x { services } x { SDK languages }. It's challenging for AppSec tools, in particular SAST, to keep up with that.
- **Infrastructure as Code.** Some specifics thoughts on this space:
  - Not truly "AppSec." Our point of view is that for static analysis, AppSec naturally extends to IaC because it's also code analysis and is often in the same repos. There is also an aspect of dynamic testing: What is the actual configuration in the cloud (configuration drift/production monitoring)? This seems further away from AppSec.
  - This space is still rapidly developing. There are cloud-vendor specific solutions (CloudFormation, ARM Templates), abstraction layers over the configuration (Ansible, Terraform), and abstraction through a container management layer (Kubernetes).
- **Containers**
  - Various approaches to assess security:
    - Checking the creation process files (Dockerfile), close to AppSec.
    - Image scanning, further away from AppSec.
- **Serverless/Function-as-a-Service**
  - Cloud-specific deployment model where you simply deploy code without bothering about the appserver/container/VM (that's all handled by the cloud provider). Not automatically recognized by any AppSec tool.
- **Cloud Secrets**
  - Cloud and IaC trend create the additional risk of storing hard-coded secrets. The impact of disclosure of a cloud secret can be huge.
  - Opportunity: Many cloud/API secrets tend to have fixed formats and can be found using relatively simple analysis to find regexes. Multiple vendors have jumped into this space.

- **Cloud CI/CD**
  - Hosted DevOps systems (ADO, GitLab, GitHub, AWS CodeStar, etc.).
  - Have their own (supply chain) risks, in particular injection in workflows.

Looking at the consequences of these new aspects of AppSec:

- All of them place new demands on AppSec tools.
- Demarcation between AppSec and InfraSec is becoming blurry.
- Many niche vendors are popping up that do one thing really well.
- At the same time, organizations are concerned about tool sprawl and prefer consolidated solutions.

AppSec Orchestration and Correlation

Application security orchestration and correlation has increasingly become a hot topic in the industry. These are often spoken of simultaneously, but it's really the combination of two separate topics. For the sake of discussion, we have split them into separate sections.

Orchestration

With the continued speed and complexity of modern development, the demands on AppSec teams continue to grow. Many organizations utilize numerous different tools from various vendors to cover their AppSec needs for SCA, SAST, DAST, and more. Attempting to manage each of these tools separately creates complex problems and bandwidth issues. From a broader standpoint, one security professional might only have access to security tools utilized in the applications they cover. AppSec orchestration plays an important role in enabling these small teams of AppSec professionals to meet the increasing demands and deliver scalable, dynamic, and static scanning solutions to large teams of developers across the organization. This comes from utilizing a single source to schedule automated and scalable scans across numerous tools used throughout the organization.

Correlation

Development organization leaders and executives mainly care about the risk of the environment. Risk management provides a comprehensive view of risk from applications and their supporting infrastructure. With a focus on this approach, executives can get a clearer picture of their assets, business context, and ROI. Solutions such as SaltMiner from Saltworks Security do this by pulling in contextual data beyond just AppSec.

With vulnerability management, we are seeing the continued evolution of solutions that aggregate, analyze, and report results into a single pane of glass—providing visibility into all of the application security initiatives within an organization. This provides a holistic view for organizations to assess their AppSec data at an executive level.

To further expand the basic idea of AppSec correlation as expanded vulnerability management, there is the aspect of systematic problems and patterns that can emerge. By layering the results of dynamic analysis on top of static analysis, customers gain a valuable additional risk metric that allows them to see a more complete real-world risk picture. While it is important to identify vulnerabilities early in the SDLC using technologies such as static analysis, it is also 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 such as XSS early in the SDLC, and continues to detect those issues in production, can focus their training and development resources on addressing systemic problems.

A unified application security vulnerability management platform is critical not only 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.

Next-Generation DAST

DAST is continuing to integrate earlier in the pipeline. Historically, the turnaround times of DAST scans have precluded their integration into stringent DevSecOps workflows. However, we are starting to see developer-driven DAST testing expand—extending the use of DAST beyond the hands of AppSec/QA and fully within the Dev CI/CD automation pipelines.

This enables DAST to be included in faster testing cycles. With automated security scans in the pipeline, it yields many benefits that lead to faster discovery and fixes:

- Developers are alerted to any new vulnerabilities before they hit production, optionally breaking the build to ensure that a review happens before the release.
- Testing can be run against underlying services and APIs instead of being limited to the customer-facing application, leading to faster identification of the underlying issue when a bug is found.

With DAST scans aligned with functional test scripts, only the portions of the application that are being worked on remain in the context of the code they were working on. Scans that run automatically and integrate with existing processes and tools keep security and development teams moving quickly. They remain focused on fixing critical issues, not scheduling scans. This approach typically yields better results than the recent increase in an IAST method. Passive IAST doesn't crawl the application and is dependent on the user creating functional testing scripts and manually exercising the application. DAST not only has these capabilities, but is also effective at discovering the attack surface of the application on its own. This becomes a question of how much you trust QA to create a script for every scenario and code path. Unless they can cover those scenarios 100%, you will still need DAST to find all of the attack surface.

Testing earlier means organizations don't need to re-orient their entire development process to a late-stage security gate as they did before. This allows for better scalability of DAST, which typically has been a major hurdle for security teams. Solutions that centralize the scanning are a key element of making DAST work at scale in DevSecOps pipelines.

A great way to set up DAST for both fast feedback and comprehensive scanning is:

- For every check-in, run any functional tests through DAST. This enables developers to get quick feedback on their changes, in the same way as IAST.
- On nightly builds, run the more comprehensive scan that crawls the entire application, giving you full and complete coverage.

Only 29% of organizations have automated most (75% or more) of their security testing. Fewer than half of organizations (44%) have included security tests and reviews as part of coding workflows.

Machine Learning and AI Are Key to the Next Evolution of Automation

Automation is one of the biggest drivers empowering shift-left security. This is backed up by studies showing that companies who use automation are twice as likely to implement security testing. While many organizations know there is a need for automation, and some automation has taken place, there is still more room for improvement. Gartner states that while 95% of respondents use automation, only 33% fully automate their deployment pipeline. Furthermore, Gartner indicates that 32% of organizations manually integrate their security tools.

While the challenges and push to automate more of the implementation and tools used throughout the development process continues, we are also seeing more benefits in the form of automated remediation utilizing existing data and machine learning. For example, we are seeing this in software composition analysis with automated pull requests. We have innovated in this space with our Fortify Audit Assistant by OpenText tool as well. Our application security as a service offering, Fortify on Demand by OpenText, runs thousands of static, dynamic, and mobile scans per week, scanning billions of lines of code. Fortify on Demand takes customer application source code, runs the scan, and then (as a value-added service) passes these raw scan results to a team of auditors who are subject matter experts. These auditors identify and prioritize the noteworthy findings while removing the noise from the results.

Consequently, Fortify on Demand customers receive actionable results that enable them to focus on fixing the most critical issues. The Fortify Audit Assistant service uses machine learning algorithms to feed off the hundreds of millions of anonymous audit decisions from Fortify on Demand experts. These decision models are actively used and developed for Fortify on Demand, but can also be automatically applied on premises to Fortify Static Code Analyzer by OpenText results by using Fortify Audit Assistant. This innovative and patent-pending technology has been available to the Fortify product line by OpenText's customers for the past five years.

In the future, we will see more AI-assisted auditing for other AppSec vulnerability types, likely starting with a subset of SAST findings (configuration/IaC style, etc.). In addition, there are numerous use cases for machine learning advancements. Our software composition analysis product does this with Debricked Open Source Select by OpenText, which utilizes it to compare and analyze the health of all open source on GitHub to make better decisions when researching a library or a framework.

Final Thoughts

The application security industry continues to evolve at pace as organizations recognize that software security risks need to balance with business imperatives that accelerate the speed of digital innovation. Fortify products' Software Security Research team finds that a vast majority of web of applications have at least one critical or high-severity issue (79% in our latest AppSec Risk Report). Combine this with the known vulnerabilities within open-source components, additional attack surfaces associated with the move to the cloud, and more, and it is clear that having an application security partner you can trust is critical for success.

The trends we've discussed here fit into a modern development framework where security is developer-driven and focused on actionable results that enable digital innovation.

We'd love to hear from you. Which of these trends interests you the most? Are there other AppSec innovations that you're watching closely?

About OpenText

OpenText enables you to build software resilience from an industry-leading AppSec partner you can trust. OpenText static, dynamic, interactive, and open source security testing technologies are available on premises, SaaS, or as a managed service—offering organizations the flexibility they need to build an end-to-end software security assurance program.

Learn more at
www.microfocus.com/appsecurity

Connect with Us

www.opentext.com

opentext™ | Cybersecurity

OpenText Cybersecurity provides comprehensive security solutions for companies and partners of all sizes. From prevention, detection and response to recovery, investigation and compliance, our unified end-to-end platform helps customers build cyber resilience via a holistic security portfolio. Powered by actionable insights from our real-time and contextual threat intelligence, OpenText Cybersecurity customers benefit from high efficacy products, a compliant experience and simplified security to help manage business risk.