



Stop Vulnerabilities from Getting into Your Application Software Development Lifecycle

Integrate and automate security scanning and testing in the software development cycle.



Table of Contents

Executive Summary	3
Introduction.....	4
Integrating AppSec at Each SDLC Phase	5
Stopping vulnerabilities before they stop you	5
Seamless Application Security Testing	6
Secure development	6
Securing the supply chain and your applications	6
Security teams help reduce AppSec risk	7
Secure testing- embed scalable security into the development tool chain	7
Integration & Staging.....	9
Production – Continuous monitoring and protection	9
Identify, Verify, Remediate	10
Build security into the pipeline	10
Planning is the first step	13
Scanning the SDLC pipeline	14
Design application security integration into you processes	14
Coding is the fun part of a developer's work life	15
Verification.....	17
Integration into your test cycles	17
Remediate	17
Integration and staging.....	17
Production and deployment	18
Maintenance.....	18
Partnering with Security Experts	18

Executive Summary

Innovation is at the heart of every successful company and customers from large corporations to small sole proprietorships are accelerating their consumption of software applications to meet their business goals. If your company produces software applications that your customers rely on to provide the benefits they seek and be safe from cyberattacks, but you aren't sure how to implement application security (AppSec) best practices into your organization, this Micro Focus® Fortify eBook provides concrete guidance on the current state of cybersecurity and the steps to successfully integrate and automate AppSec into daily routines and developer environments.

Companies that aren't themselves in the business of tracking cybersecurity threats often don't have the capacity or resources necessary to identify, verify, and remediate weaknesses in software code to avoid sending out applications that are vulnerable to threats. We know the time is shrinking from software release to a known exploit. In 2006, it took an average of 45 days to exploit a vulnerability. It only took three days in 2017 to exploit the well-noted Equifax application software vulnerability.

Integrating and automating AppSec into your development and operations (DevOps) environment provides the assurance that your customers or your company won't show up in the paper—at least not for cybersecurity exploits. Around the globe, developers produce over 111 billion lines of new software code each year. Releasing software applications without critical or high-risk vulnerabilities requires integration and automation of AppSec testing tools into the software development lifecycle (SDLC).

And it not just for code developed internally. Scanning open source components before they get into the development environment is crucial to eliminating externally introduced vulnerabilities. In 2018, 87% of the applications inherited a critical severity vulnerability from referenced components, up by 22% since 2017. [Ninety-two](#) percent of modern applications contain open source and securing them becomes a top priority for your application security checklist.

AppSec integration into your SDLC requires cooperation between development, operations, and security teams. At each phase of the process, the right AppSec scanning testing tools identify scans to perform for optimal support in the least amount of time.

Research shows that flexible development and operations (DevOps) environments that implement the practices described by the [OWASP Software Assurance Maturity Model \(SAMM\)](#) achieve the highest level of application security and reduces the time to perform manual risk identification, verification, and remediation from 2 weeks to a couple of minutes or hours allows the daily sprints necessary to deliver high-quality software applications — free of potential vulnerabilities and exposure to cybersecurity exploits.

Introduction

If your business goals require generating revenue through software application innovation, new products, and feature releases, your innovation can't slow down. Your customers want the confidence that their applications are free of vulnerabilities. Yet companies around the globe are skeptical that application security isn't possible without putting the brakes on new ideas.

Yet, cyber threats are rapidly becoming one of the most significant existential threats to businesses, institutions, and governments, with implications for national security, economic prosperity, public safety, and organizational reputation.

An average of [130 large-scale attacks](#) occur yearly, yet [fewer than half of global companies report](#) they are sufficiently prepared for a cybersecurity attack. Combined with the sharp increase in using digital technologies in the workplace, organizations find it difficult to keep pace with the evolving cyber battlefield. This is especially true when talking about application security (AppSec).

Research from Cybersecurity Ventures estimates developers produce one hundred and eleven billion lines of new software code each year⁴⁹—introducing a massive number of vulnerabilities and potential exploits.

In a recent CSO Online security article by Dan Swinhoe, "[What is the cost of a data breach?](#)", U.S. organizations face the highest cyber-attack costs at an average of \$8.19 million per breach. This is driven by a complex regulatory landscape that varies from state-to-state, especially when it comes to breach notification. In the UK the figure is slightly lower than the global average, at \$3.88 million.

Data breaches average 25,575 records, a 3.9% increase since 2018. In the U.S. the average number of records is 32,434, followed by the UK with 23,600 records. Globally, each lost record costs an average of \$150. But in the U.S. that figure rises to \$242 while in the UK the cost is \$155 per record.

The alarm is just getting louder, and the time it takes for a vulnerability to be discovered and exploited is shrinking. In 2006, it took an average of 45 days to exploit a vulnerability. It only took three days in 2017 to exploit the well-noted Equifax application software vulnerability.

According to the latest [Micro Focus State of Application Security in the Enterprise](#) report, companies understand the need for application security "...with 30% of businesses rating their biggest challenge as delivering business-critical applications without vulnerabilities". Another 21% considered security program coverage as their greatest challenge. Overall, 72% of companies are at least slightly more concerned about application security than they were 12 months ago.

Companies doing business in the European Union (EU) must meet new compliance requirements under the General Data Protection Regulation (GDPR), highlighting the importance of including application security and vulnerability assessments to meet privacy and security compliance.

Throughout the paper, we will use the following standard definitions:

- **Exploit**—A successful attack against one or more vulnerabilities that violates the security posture of the company.
- **Vulnerabilities**—A property of software system requirements, design, implementation, or operations that could accidentally trigger or be exploited it, resulting in a security failure.
- **Weakness**—A defect in the system that may or may not result in a vulnerability.

Integrating AppSec at Each SDLC Phase

Application security doesn't have to come at the cost of faster innovation.

Stopping vulnerabilities before they stop you

We all know that the digital transformation demands speed and innovation to satisfy customer requirements for new software features accessible through web and mobile applications.

"Today's winners release greater software faster and more securely than their competitors." Michelle Duffy, product marketing manager, Sonatype.

Digital transformation also changes the way businesses need to secure their data. In the past, big network firewalls acted as the primary point of control. Data from external sources came through a single access point that identified suspicious data packages, not allowing them to pass into the internal network. Now, data traffic from the web and mobile applications are often routed around the firewall and no longer provides the same level of security.

In a recent presentation from Sonatype, in 2018, 51% of enterprises experienced a security breach in twelve months, 43% coming from external persons. Of these, 68% target web applications. Every time your employees log in to a web application, without properly integrated security measures, your company is vulnerable to a security breach.

The [International Council of E-Commerce Consultants \(EC-Council\)](#), the world's largest cybersecurity technical certification body, identified ten essential steps to ensure your applications meet stringent security compliance requirements that help avoid security breaches, reduce lost productivity, and preserve hard-earned business revenue.

[Ten steps to AppSec](#): Design an end-to-end software development lifecycle (SDLC) framework.

- Develop a secure SLDC framework for software development.
- Develop knowledge of OWASP top ten threats.
- Capture security requirements at the start of the SDLC.
- Define, maintain, and enforce industry application security best practices.
- Implement secure application design and architecture.
- Implement secure coding practices for validation, implementation, and authorization.
- Teach a culture of secure coding, cryptography, session management, and error handling.
- Rate the severity of found defects and publish comprehensive reports of associated risks with each identified vulnerability.
- Use well-known application security tools like Fortify, WebInspect, etc., to scan and test applications.
- Post deployment and maintenance to ensure security at different worlds.

Automating and integrating AppSec testing into your SDLC helps speed up the entire development cycle. Research shows that flexible development and operations (DevOps) environments that implement the practices described by the [OWASP Software Assurance Maturity Model \(SAMM\)](#), achieve the highest level of application security in the shortest amount of time. But no matter what development model you

use, Waterfall, Agile, Spiral, or DevOps, the same steps apply to ensuring that your code and software applications are secure.

Seamless Application Security Testing

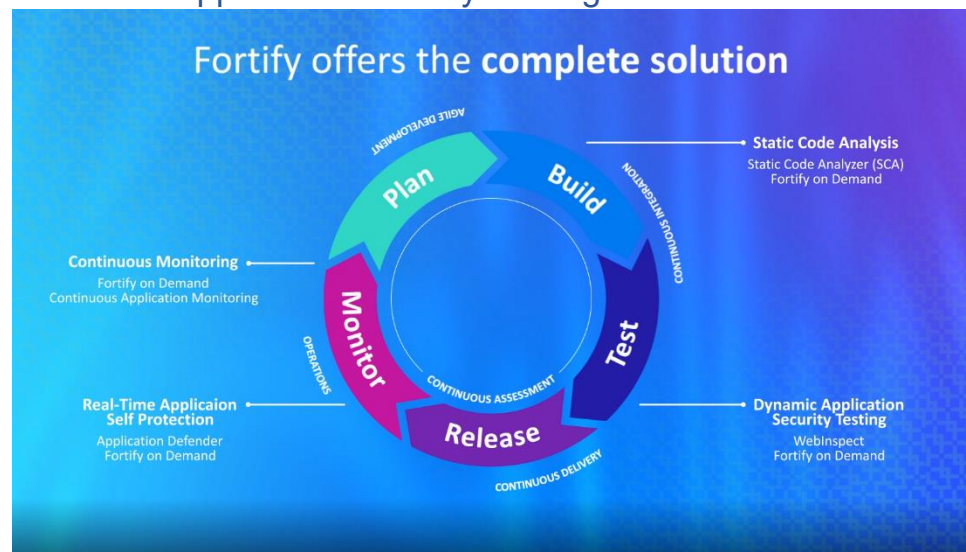


Figure 1: <https://www.microfocus.com/en-us/trend/security-risk-governance>

Secure development

Optimal application security starts with the planning phase and the architecture review. During this stage, developers, security, and business teams collaborate on the policies, software library framework, artifacts, bill of materials (BOM), and AppSec tools selection required to scan each phase and component for weaknesses and vulnerabilities.

Most software security vulnerabilities are the result of [security bugs](#) or defects within the software. Two primary causes of these potential vulnerabilities:

1. Non-conformance, or a failure to satisfy requirements
2. An error or omission in the software requirements.

Development teams identify which code must be from an open-source, third-party, commercial, or customized to meet the business requirements. Collaborating early in the process allows all parties to identify weaknesses, vulnerabilities and define agreed-upon remediation techniques.

Securing the supply chain and your applications

Previously deployed applications, as well as custom and open-source application codes, make it easier to deliver new and updated features faster. Most developers use open-source application code for part of their development. In 2018, over 146 billion Java components were downloaded, and approximately 85% of released applications used open source libraries. The more extensive public library of JavaScript tools Nothing Plus More (NPM), gets a staggering 7B download requests a week.

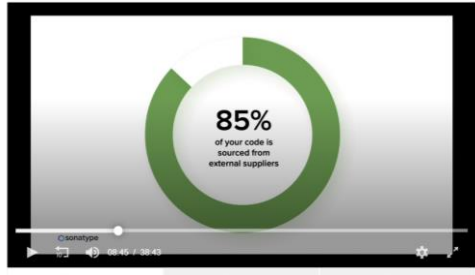


Figure 2: Percent of open source code used in development

Using open-source code reduces the time required to deliver an application, but it can also introduce security risks. Supply-chain attacks are increasingly becoming part of the open-source ecosystem. For example, attackers are accelerating activity aimed at polluting the open-source software supply chain. At least six incidents in 2018 targeted open-source libraries with backdoor code, according to the software-security firm Sonatype. At least six incidents in 2018 targeted open-source libraries with backdoor code, according to the software-security firm Sonatype.

Ensuring that your processes are not creating vulnerabilities is a major headache when application programming interfaces (API) used to accelerate software application delivery can't be tested as a whole. Deprived of integrated security policies across your SDLC and your company leaves holes and weaknesses that can be easily hacked. According to Sonatype, about 60,660 thousand JavaScript packages are downloaded annually, with about 51% known vulnerabilities—allowing faster and more frequent attacks to occur.

Security teams help reduce AppSec risk

Developers and project owners release about fifteen thousand new releases daily, and the appropriate tools are necessary to automate testing of the integration and delivery pipeline. Performing static scans on open-source components, searching the [OWASP Top Ten](#), and the [Common Vulnerabilities and Exposures \(CVE\)](#) database for known vulnerabilities helps remove vulnerabilities and potential exploits before the code is entered into the software library.

Your security teams provide the right resources to ensure that open-source and custom code don't contain critical, high, or medium vulnerabilities. Working together, security and development teams contribute to the increased success of your software applications. As part of the SDLC team, security specialists use AppSec scanning tools to scan the code, identify vulnerabilities, perform risk assessments, and remediation techniques are documented and enforced.



Figure 3: Integrate security scanning into your SDLC pipeline

Secure testing- embed scalable security into the development tool chain

Just what is AppSec testing? While some think that security software testing is a phase of the SDLC, it is a continuous process that ensures the code and software applications are free from any vulnerabilities,

threats, and risks. AppSec testing looks for all possible loopholes and weaknesses that expand the attack surface and attract attackers, potentially resulting in a security breach. See Figure 4.

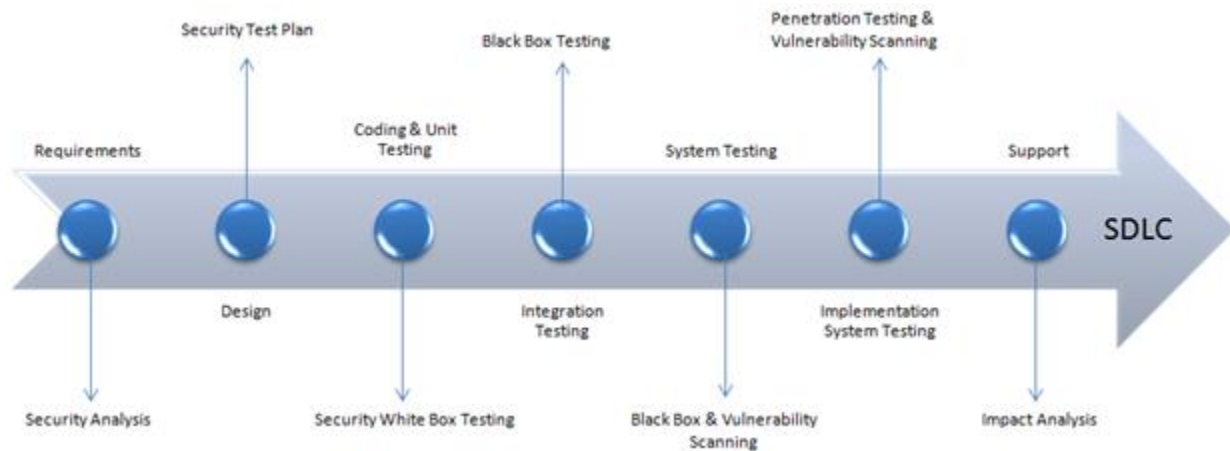


Figure 4: Software Development Life Cycle

Reliable, ongoing application security testing programs should include both static code analysis and regularly scheduled dynamic scans, and supports faster software innovation.

- Static assessments help developers identify and eliminate vulnerabilities in the source, binary, or byte code to build more secure software.
- Dynamic assessments mimic real-world hacking methods and attacks. It uses automated, interactive, and manual techniques to provide a comprehensive analysis of complex web applications and services.

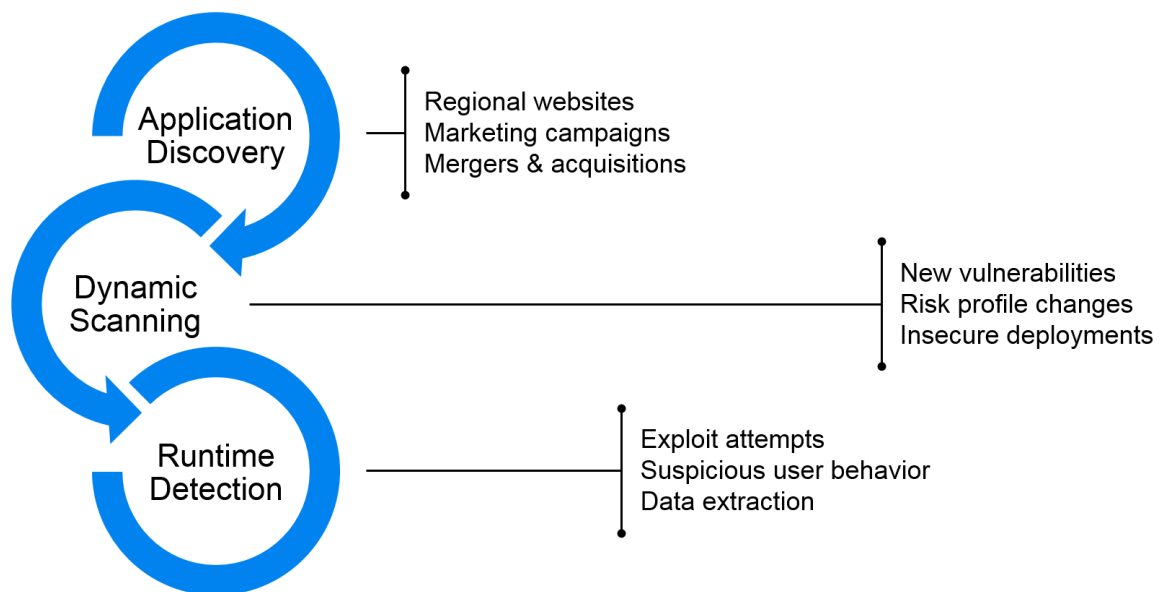


Figure 5: Real-time discovery process

- Vulnerability scanning—Automatically and manually scan code and applications against known vulnerability signatures.
- Security scanning —Identify network and system weaknesses, and later provides solutions for reducing these risks. This scanning can be performed for both manual and automated scanning.
- Penetration testing—Simulate and analyze an attack from an external malicious hacker.
- Risk assessment—Analyze and classify security risks as low, medium, and high, recommending controls and measures to reduce high and critical risks.
- Security auditing—Internal inspection of applications for security flaws. An audit can also be done via line-by-line inspection of code.

Integration & Staging

Up to now, software development may have been done in a silo. But at integration, developers work from a common environment where all developers' commit code changes are captured. The goal of this environment is to combine and validate the work of the entire project team so it can be scanned and pen testing performed before being promoted to the staging environment.

Once the integrated code is scanned and any additional vulnerabilities have been identified and remediation strategies have been applied, the application is ready for staging.

The staging environment is as identical to the production environment as possible and is used to simulate as much of the production environment as possible. The staging environment can also be used for demos and training without jeopardizing the production code.

Beta testing with key customers is commonly done at this time. It is an opportunity to verify the application meets the feature and design requirements prior to production.

Software Security Assurance (SSA) is the process of ensuring that [software](#) is designed to operate at a level of security that is consistent with the potential harm that could result from the loss, inaccuracy, alteration, unavailability, or misuse of the data and resources that it uses, controls, and protects.

The software security assurance process begins by identifying and categorizing the information that is to be contained in or used by the software. The information should be categorized according to its sensitivity. For example, in the lowest category, the impact of a security violation is minimal (i.e. the impact on the software owner's mission, functions, or reputation is negligible). For a top category, however, the impact may pose a threat to human life; may have an irreparable impact on the software customers' missions, functions, image, or reputation; or may result in the loss of significant assets or resources.

Production – Continuous monitoring and protection

AppSec scanning tools support operations and development folks in quickly responding to last-minute changes. New requirements are swiftly incorporated, scanned, and resubmitted in preparation for release. Updated systems allow customers to safely purchase the new software or update their existing software. Once released, the application is scanned and monitored for defects, security vulnerabilities, attacks on the application, new requirements, and any other changes.

Fortify on Demand, Fortify Static Code Analyzer, and Fortify WebInspect secure your applications and protect your business with the most comprehensive, accurate, and scalable application security solutions designed to work with your DevOps processes.

Identify, Verify, Remediate

Integrate and automate security scanning and testing measures throughout your application software development cycle.

Not all data breaches are caused by a software code vulnerability. But weaknesses in code that ultimately lead to a data breach require mitigations, legislation, and new standardization rules. Excluding software consumed from third parties, the National Vulnerability Database (NVD) showed that 12.8% of companies reuse software apps. In 2018, [cross-site scripting \(XSS\), remote code execution, authorization bypass, and cross-site request forgery \(CSRF\) topped the list of exploits²](#). ...[11.1% of Java download requests have a known vulnerability, 51% of Java script downloads have a vulnerability](#)...

Micro Focus' Software Security Research team recently analyzed public data and vulnerabilities detected in over 11,000 applications using its Fortify on Demand platform. The analysis showed that 16,517 application security issues were reported to the NVD in 2018—the most ever in a single year.

Build security into the pipeline

Just how is application security (AppSec) built into the entire process without interrupting code development and meeting critical timelines?

In fast-paced business cycles, successfully securing software applications requires that development, operations, run-the-business, and security teams collaborate, integrate, identify application security metrics, remediation plans, monitoring, and reporting policies into the software development life cycle (SDLC).

AppSec scanning tools ensure that the software code is automatically scanned at each phase—reducing the time and resources required to meet your AppSec goals with confidence. Building security into the seven phases of the SDLC used in this e-book: plan, design, code, test, integrate and stage, operate, and maintain requires integration with the tools that developers count on. Tools such as integrated development environments (IDE), build tools, code repositories, bug tracking, and ticketing systems, guarantee that your software application is free of potentially harmful vulnerabilities.

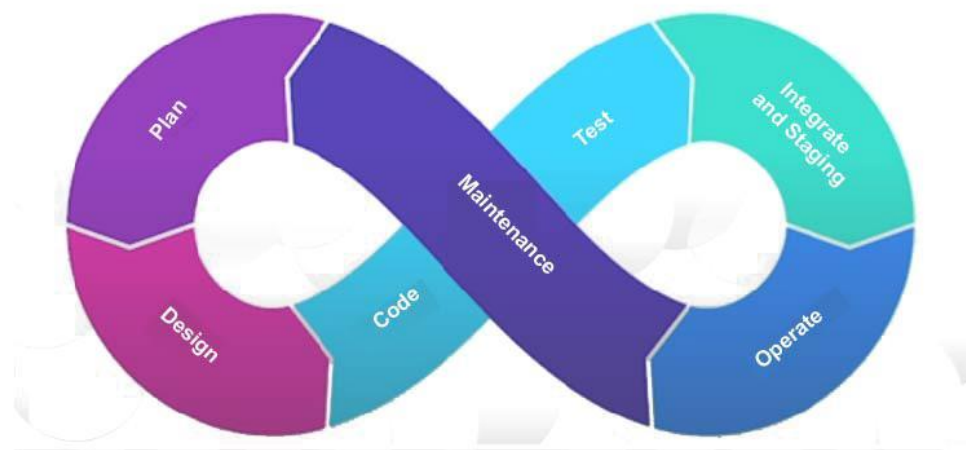


Figure 6: SDLC phases

Team engagement in the overall process

- Development
 - Uses threat modeling to ensure policy compliance requirements are factored into the design.

- Tracks and reports on the number of vulnerabilities or weaknesses and how many have been fixed.
- Enforces practices such as minimum privilege and minimum data.
- Identifies open source and custom components.
- Operations
 - Manages process changes and builds security into the CI/CD (continuous integration/continuous delivery) workflow.
 - Collaborates with the teams on business impact assessments (BIAs), data classification, and management.
 - Implements centralized policies around business criticality and risk.
 - "Owns" the layer of security tools closest to the applications, including critical operational technologies such as web application firewalls (WAF).
- Run-the-business
 - Provides oversight on how well application security is being implemented; is the application secure and can it go live?
 - Understands blind spots such if code is being built but not tested, you'll want to identify the holes.
- Security
 - Defines and communicates service levels and policies.
 - Identifies, monitors, and reduces the business risk from an application's source code.
 - Scans open source, third-party, and custom code.
 - Manages demand management (workload).
 - Provides recommendations to remediate issues.
 - Helps remove security blockers in the software supply chain.

Automating and integrating security scanning and testing into each phase of the SDLC requires the [right testing tools at the right time](#). Integrating tools such as SAST, DAST, IAST, RAST, WAF, and SCA that automatically scan the code, identify internal and open source risks, verify custom code, enforce policies, and remediate any found vulnerabilities promotes internal coding, compilation, and management of secure libraries and artifacts without slowing down the build.

How do these tools work, and when do I integrate them in the SDLC?

- Static application security testing (SAST) is the front line of secure code development. Integrating SAST into the developer IDE scans the source code looking for known patterns of weaknesses that lead to vulnerabilities. Most SAST tools support the major web languages, Java and .NET, and generally support some form of C, C++, or C#. The tools should also support your DevOps team's software delivery and integration pipeline. Running them as often as practical provides feedback directly to the developer, helping eliminate security defects.
- Dynamic application security testing (DAST) runs against compiled or production code. Also known as "black-box testing," dynamic analysis locates various types of vulnerabilities in running applications. For risk-free software applications, organizations should run both dynamic and static testing.

- Interactive application security testing (IAST) performs what is often called “glass-box” testing. IAST is designed to catch attacks that the other approaches cannot by running an agent that collects event data from running applications. Automated DAST tools can provide the same level of detail.
- Run-time application self-protection (RASP) works “inside” an application’s runtime environment — detecting changes that may indicate an attack is underway. RASP is effective for legacy applications when fixing vulnerabilities by modifying the source code may not be an option. RASP tools expand the attack surface and run inside the runtime production environment.
- Web application firewalls (WAF) detect intrusions at the perimeter of an application server’s network. WAFs are signature-based devices and apply “a set of rules to an HTTP conversation,” as described on the OWASP website. “Generally, these rules cover common attacks such as cross-site scripting (XSS) and SQL injection.”
- [Software Composition Analysis](#) (SCA) helps your development teams run automated security checks and reporting throughout the SDLC—identifying and building an inventory of all open source components in the environment and detecting which identified vulnerabilities can put your applications at risk. Basic information includes the open source license and if a security vulnerability is associated with that component.
- Penetration testing (pen testing) is a manual process and isn’t covered in this document. However, it should be part of the integration and staging process.
- Mobile security testing consists of static code analysis for mobile application source code, customized dynamic testing methods for compiled mobile apps (such as fuzzing) and testing of the server backend/services. The [Mobile Vulnerabilities Security Verification Standard from OWASP \(MVSVS\)](#) provides a good model to govern the verification needs for mobile apps.

Advanced tools offer [automatic policy enforcements](#) by cross referencing every open source component found in your code with your organizational policies, triggering different responses from initiating an automated approval workflow to failing the build.

“The pressure companies are putting on developers to create and deliver changes faster and faster is a big reason for this pattern,” says Amy DeMartine, Principal Analyst, Forrester Research. “All they have to develop is a thin slice of proprietary code, with 80-90% of the rest consisting of open source components that serve as building blocks—functions that can build a graph or call a form, for example—then you can be first to deliver that new chunk of functionality through your DevOps pipeline....”

“It’s not that we’re writing worse code these days,” says DeMartine. “It’s just that the people writing open source components are not always scanning before they release, and it’s critical that the business consuming these components look at these new versions and understand where the vulnerabilities lie.”

Automating security testing processes provides a competitive advantage when products can be released faster without including known vulnerabilities. Integrating security intelligence into the DevOps pipeline optimizes the power of automation for agility, speed, innovation, and delivery.

Armed with a complete vulnerability description and actionable recommendations, your security experts and developers can accurately mitigate any found security flaws. Using dashboards and reports, AppSec programs provide visibility to simplify managing your risk profile, confirming remediation, tracking metrics and trends, and reporting progress.

Planning is the first step

Application security is an iterative process, and throughout each phase of the SDLC, successful AppSec integration and automation start with setting clear program objectives, security policies, deployment timelines, scope, and success measurements. AppSec practices are simpler to integrate, and friction between teams is removed when application security policies are included during the planning phase (known as shifting left). The [NIST Cybersecurity Framework](#) (CSF) provides a common taxonomy and mechanism for organizations to:

- Describe the current cybersecurity posture.
- Describe the target state for cybersecurity.
- Identify and prioritize improvement opportunities as part of a continuous and repeatable process.
- Assess progress toward the target state.
- Communicate among internal and external stakeholders about cybersecurity risk.

Identifying the technologies used throughout the SDLC allows you to define architecture standards and “secure by default” design patterns to follow. Knowing your organization’s application architectures is essential for both managing your technical risk exposure and constructing other portions of your program.

Establish and document organizational, development, and security policies. Define the architecture, technology, and security enforcement strategies required for the SDLC process to succeed.

Table 1: Asking the right policy questions

Example policy questions	
Who issues the policy, and who is responsible for its enforcement and maintenance?	What are the remediation strategies and solutions, and how will they be enforced?
Which security programs will be implemented for the environment, such as antivirus, firewalls, anti-malware, and anti-exploit software?	How are updates and patches applied to limit the attack surface and plug up application vulnerabilities?
How often is open source and third-party code patching done?	What encryption methodology will be used?
How will data be backed up? Is it automatically backed up to an encrypted cloud server with multi-factor authentication?	Who manages access privileges?
Who responds to and resolves security incidents, and how?	When will pen testing be performed and how often?
Who is responsible for testing and validating open-source code?	How often will automatic code scans occur?
What are the acceptable levels of risk, false positives?	What are the parameters used to perform build/break stress testing?
How will assets be tracked?	Who will train users on the application?
How many open Jira tickets are acceptable at release?	What type of email warning is acceptable for critical highs, mediums, and lows?
Are the license versions the latest?	How will your containers be managed?
What compliance and regulatory standard must be met and who will audit?	What is the technical debt associated with an easy fix versus a more thorough approach?

Scanning the SDLC pipeline

The right tools help developers and operations automatically enforce these policies and control risks across the SDLC, so you can stop vulnerable components from entering the SDLC in the first place.

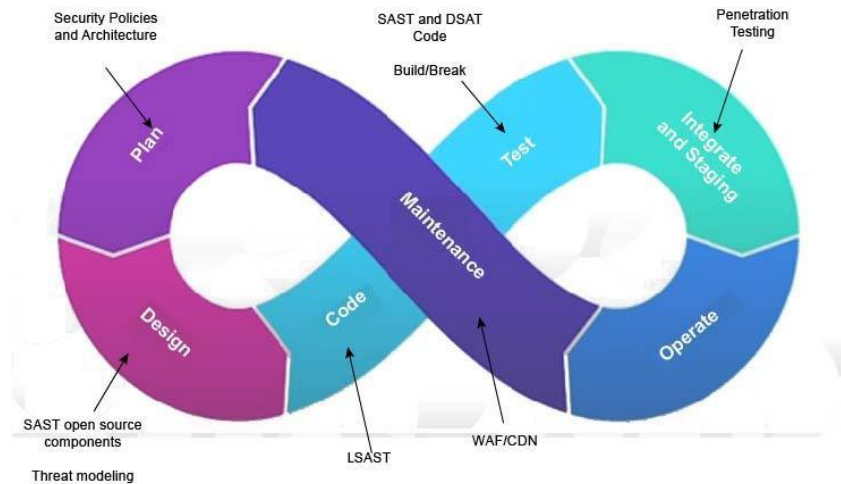


Figure 7: Scanning the SDLC pipeline

Design application security integration into you processes

Collectively, security and development teams provide a significant contribution to the success of your software applications.

An essential part of supporting the business goals is deliver secure, risk-free software applications or upgrades that meet your customers' needs and requirements. Development teams define the components, libraries, artifacts, and sourcing from open source, third-party vendors, or custom code, and other details required to deliver high-quality and defect-free software applications.

Security teams help define AppSec enforcement measures such as defining how securing open source code policies are enforced when proxying to public repositories. For example, if your policies block components with a CVE of 7, any flagged component won't be downloaded.

The following tasks are performed during this phase:

- Identify and specify the goals and how they fit with the business objective.
- Establish the development and testing environment.
- How and where will the application reside.
- Describe how this application will be developed and tested; including responsible parties.
- Define the software components, open source, third party, or custom code needed to build the software application.
- Conduct a security risk assessment to address the following components:
 - Assets
 - Threats
 - Vulnerabilities
 - Likelihood

- Consequences
- Safeguards.

Establish operations goals.

Together, software developers and security specialists perform risk assessments that evaluate compliance with baseline security requirements, identify threats and vulnerabilities, assess alternatives for mitigating or accepting residual risks, and deliver the following:

- Software Security Assurance (SSA) plan
- Bill of materials and software artifacts
- Weaknesses and vulnerabilities
- Remediation strategies

Establish a secure central repository for documenting, tracking, storing, and managing libraries and artifacts. Automated tools like the Nexus Repository from Sonatype and manual methods can be utilized to produce an initial inventory.

- Libraries–Repositories for data and programming code used in the development of software programs and applications. They assist programmers and programming language compilers in building and executing software.
- Software build artifacts–Produced during software coding, software artifacts can include project source code, dependencies, binaries, resources, use cases, class diagrams, Unified Modeling Language (UML) models, requirements, and design documents. They help describe the function, architecture, and software design associated with the application. Stored in a repository, they can be retrieved on-demand and shared across teams. Done early in the process and at each sprint, threat modeling helps improve application security and defines security policies for each application. Identify and prioritize potential threats or actual malicious exploits such as DoS and code execution attacks. Adverse events and the actions taken in each case are documented.

Performing threat modeling is a planned activity that identifies and assesses application threats and vulnerabilities. This rigorous process is not typically done every day, but it is an important part of design. Performed early, it answers these questions:

- What are we working on?
- What can go wrong?
- What will we do with the findings?
- Did we do a good job?

Coding is the fun part of a developer's work life

Integrating SAST into your procurement and delivery pipeline, find weaknesses before they lead to a vulnerability.

[Ninety-two](#) percent of modern applications contain open source code, and it must be a top priority on your application security checklist. [In 2018, 87% of the applications inherited a critical severity vulnerability from referenced components, up 22% since 2017.](#) Scanning open source components before they get into the development environment eliminates externally introduced vulnerabilities.

Automatically scanning inbound requests and enforcing policies when using proxying to publicly available code through tried and tested software security tools stops risks from entering the SDLC. Auditing applications creates actionable metadata, identifies risks, enforces policies, and remediates issues.

SAST also finds potential vulnerabilities (like an SQL injection error) in your proprietary code (built in-house, outsourced, and third-party) using different analysis methods such as taint, structure, or control flow analysis. Tools like Nexus Firewall and SAST from Sonatype prevent potential vulnerabilities in open source code from entering the SDLC based on defined policies.

[Audit Assistant comes with Micro Focus® Fortify Software Security Center](#) and enables machine-learning-assisted auditing, an important part of the recipe for successful automation. Audit Assistant leverages millions of previous audit results from static scans (part of SAST) and automatically determines the likelihood that found vulnerabilities are critical, false positives, etc., reducing the amount of manual auditing performed, which would otherwise be a 100% manual process.

Developer benefits of running SAST:

- Identify and eliminate vulnerabilities in source, binary, or bytecode.
- Review scan results in real-time to find vulnerabilities faster and get access to recommendations using line-of-code navigation and collaborative auditing.
- Fully integrated with the Integrated Developer Environment (IDE).

Tools like Fortify Static Code Analyzer (SCA) automatically scans the actual source code in the early stages and discovers security vulnerabilities in all your applications whether they are built in-house, outsourced, third-party, or mobile. It identifies security vulnerabilities in source code and prioritizes them by severity and importance, pinpoints the root cause of the vulnerability, correlates and prioritizes results in the source line of code detail, and provides best practices to help developers remove exploitable vulnerabilities that pose the greatest threats.

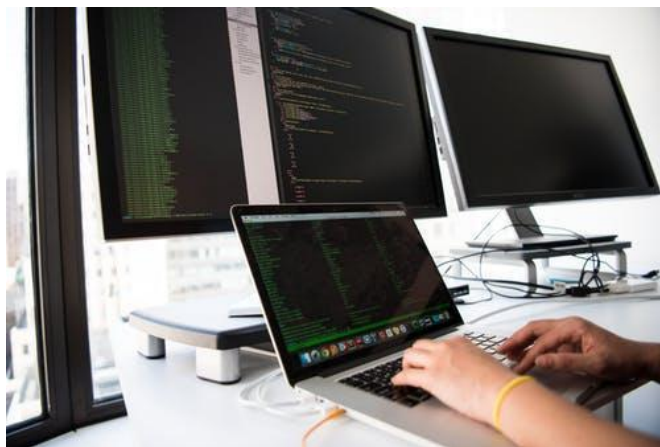


Figure 8 Eliminate security defects

Developer productivity and accuracy increase using tools that are integrated with your IDE and checks the code as its written. Fortify's Security Assistant checks (like a spell checker) the code in real time—identifying and removing software vulnerabilities as coding progresses.

Fortify products provide clear explanations of each vulnerability, technical details, remediation guidance of your scan. Integration with [Secure Code Warrior](#) provide developers clear explanations of each vulnerability result through gamified training.

When using containers to manage your code, libraries, artifacts, etc. ensure that your container images are pristine. At build time, crawl through all the components using a tool like Fortify WebInspect or Fortify on Demand to determine what has been included. Container scan setup/upload and container scan reporting are provided within the existing reports section automatically.

Verification

Integration into your test cycles



Figure 9: Testing environment

Your code is ready, and now it is time to do verification testing. But evaluating the code at the end of each sprint slows down the release process. Fortify's Software Security Center Audit Workbench shows your test results in a filtered and viewable way, helping you become more efficient in auditing and remediation of vulnerabilities. Using Smart View, you can pinpoint data convergence issues for more rapid remediation.

In most cases, organizations should run both dynamic and static testing. Static analysis tools give developers

feedback and educate them at the same time, while dynamic analysis tools can give security teams a quick win by immediately pinpointing exploitable vulnerabilities in either production or pre-production environments.

SAST products that are capable of detecting deserialization weaknesses will report and flag an issue. Calmly, check your application portfolio through your software security dashboard. This allows you to see which applications are at real risk and help you program and prioritize your patch release process. Report new vulnerabilities to the Common Vulnerabilities and Exposure (CVE) organization.

Arm yourself with a comprehensive audit of all vulnerability classes. A dynamic or mobile assessment of the running application in a QA, test, or staging environment simulates the real-world hacking techniques and attacks employed by the bad guys.

Dynamic application scanners such as Fortify WebInspect crawls modern frameworks and web technology, expanding the attack surface coverage and providing thorough, easy-to-read reports on found vulnerabilities. It thoroughly analyzes today's complex Web applications and web services for security vulnerabilities. Mimicking real-world hacking techniques, you get comprehensive dynamic analysis and runtime visibility to detect new potential exploitable code that would otherwise go undetected before an application release is deployed to production. Furthermore, automatically running DAST scans with Fortify's runtime agent supercharges dynamic testing to find more vulnerabilities—and fix them faster.

Build and break stress testing is an important part of evaluating your software application. Implementing your application security policies such as stopping the build if critical or high-risk vulnerabilities are found ensures that the code is free of risks and is stable. Remediation guidance identifies the vulnerability category, risk level, and should be written an easy to digest way that developers clearly understand issue and resolution. Fortify integration with the build tools allow you to break the builds based on identified severity levels such as the OWASP Top 10.

Remediate

Stop vulnerabilities going out the door in your software

Integration and staging

You have been progressing through each phase, and now it is time to view the application in its entirety. Staging environments are made to test software codes, builds, and updates to ensure quality under a production-like environment before application deployment. Static assessments are often integrated into

the software toolchain as an automatic step in the continuous build and integration pipeline. Run SAST as often as practical to provide feedback directly to the developer, allowing managers and the security team to monitor the progress of developers in eliminating security defects.

SAST tools report potential vulnerability findings using integrated and automatic testing methods, including taint, structure, or control flow analysis. Experts audit the findings using details specific to their organization, such as the context of the application and deployment. When auditors determine that a potential software security vulnerability is *not an issue* during lengthy, manual audits, the verification is non-value-added time. Manual audits are time-consuming, often taking two weeks or more. Traditionally, they come at a significant cost, exposing a fundamental challenge in delivering secure applications. Tools and technology do not always immediately produce actionable intelligence.

Automating SAST audits of software applications proves the value through verifiable savings of time and effort. Fortify Audit Assistant predicts the exploitability of raw findings with a 97% average accuracy. Fortify customers using Audit Assistant reaped significant benefits, such as a 58% reduction in manual audit time, and amplified the SAST return on investment in three major ways:

- Reduces the number of issues needing deep manual examination
- Identifies relevant issues and removes false positives sooner
- Scales application security with existing resources

Production and deployment

You've achieved a final major milestone prior to releasing the application software to the public or individual consumer. Tracking status of requests, code, and activities is crucial to success of pipeline release. In these final stages verify the following:

- Are initial scans completed and the results documented?
- Have false positives been resolved?
- Has a clean (no critical and no highs) scan be achieved?

Maintenance

Don't stop now. Your application is in production and you have happy customers, but keeping the application free from exploits takes an ongoing commitment. But now you have the necessary and automation in your SDLC. AppSec testing tools provide you, your team, and the company with the knowledge that your software applications are free of potential harm. Monitoring the performance of the software is also included during the maintenance phase. It includes making changes to improve a system's performance, correct problems, enhance security, or address user requirements. Incoming design change requests, upgrades, found bugs, and fixes are a routine part of the maintenance phase of the SDLC.

Making changes to improve a system's performance, correct problems, enhance security, or address user requirements. To ensure modifications do not disrupt operations or degrade a system's performance or security, organizations should establish appropriate change management standards and procedures. Make SAST and DAST part of your ongoing maintenance policies.

Partnering with Security Experts

Whether applications are built using DevSecOps, Agile, Scrum, Waterfall, or a single developer, all need one view of the development cycle across the enterprise. Partnering with security experts like Micro Focus Fortify ([a leader in the Gartner Application Security Magic Quadrant](#)), Sonatype Nexus, and Saltworks lets your organization implement a complete software security assurance program. Integration

with your IDEs, CI/CD, and other development tools allow you to run automatic scans with actionable results, line of code detail, and specific fix locations with static testing; exploitable and overall security information with dynamic testing; and automatically push secure scans to your IDE defect management and governance tools.

Micro Focus Fortify makes building AppSec into your organization easy, flexible, and ensures high-quality, secure application software for you and your customers. Fortify reduces software risk by identifying security vulnerabilities that pose the biggest threats to your organization. It pinpoints the root cause of the vulnerability, correlates and prioritizes results, and provides best practices so developers can develop code more securely. The Fortify suite uses open APIs to embed application security testing into all stages of the development tool chain: development, deployment, and production.

Micro Focus Fortify is available on-premises or on demand. It provides a comprehensive suite of software security solutions, including application security testing, software security management, and application self-protection. IDE plugins currently include: Azure DevOps, Eclipse, IntelliJ, and JDev, improving integration with CI/CD environments such as VSTS, Jira, ALM, Octane, and Jenkins

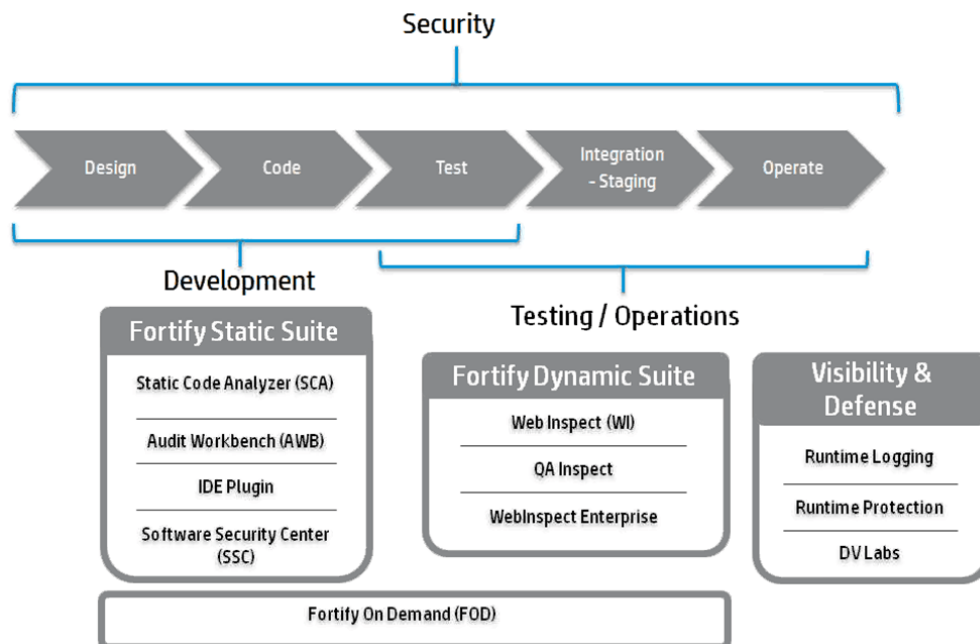


Figure 10 Integrating Micro Focus Fortify security products into your SDLC

Use Fortify On-Demand (Static and Dynamic scanning), or Fortify Static Suite (SCA, Audit Workbench, IDE Plugin, and Software Security Center), as well as Fortify Dynamic Suite (WebInspect and WebInspect Enterprise) to hold your results, pull reports and analytics, etc. on all the security testing you are doing.

- [Micro Focus Fortify on Demand Security as a Service \(SaaS\)](#) offers dynamic, static, and manual analysis to find security vulnerabilities in applications. It is essential for teams that don't have a security program in-house or need to supplement theirs. Our security experts do all the assessments, auditing, scanning, identification, reporting, etc. providing a complete analysis of vulnerabilities and recommended, remediations.
- [Fortify Static Code Analyzer](#) (SCA) – this suite of SAST tools scans the actual source code as part of the design, code, and test phases. SCA features includes:

- [Security Assistant](#) – this is our integration into the developers' IDE tools, Eclipse, Visual Studio, and IntelliJ IDEA (more IDE environments to come), and performs vulnerability checks as you type.
- [Software Security Center](#) – a centralized management repository providing visibility to your entire application security testing program, whether your software is deployed using traditional networks, the cloud, or mobile technology. It reviews, manages, prioritizes, and tracks software security testing activities, allowing an organization to measure and control its security posture across many teams and applications.
- [Audit Workbench](#) – Allows you to show your converged data test results in a filtered and viewable way by priority.
 - [Audit Assistant](#) – Automatically performs auditing, typically a lengthy process, leveraging the knowledge and experience of previous audit decisions gained through machine learning, and applies them to scan results with similar patterns.
- [Fortify Dynamic Code Analyzer](#) – Scans your running applications during test and production. The suite of tools includes:
 - [WebInspect](#) and WebInspect Enterprise – Workflows use build automation tools to manage the dynamic scanning ecosystem, including QA testing and cloud deployments. [Automating WebInspect](#) into existing DevOps systems and processes allows security tests to be run simultaneously and at scale. WebInspect employs an extensive arsenal of attack agents designed to detect and analyze today's complex web applications and web services for security vulnerabilities automatically

Mimics real-world hacking techniques using thousands of HTTP requests and delivers comprehensive dynamic analysis and runtime visibility to detect new vulnerabilities that might otherwise go undetected.

 - [Simultaneous crawl, audit, and concurrent scanning](#) make powerful scanning technology accessible to even novice security testers.
 - Perform an automatic qualitative analysis and generate Compliance Reports by grading how well your application complies with certain government-mandated regulations or corporate-defined guidelines. For example, the Health Insurance Portability and Accountability Act (HIPAA) requires healthcare providers using web-based applications to provide "procedures for creating, changing, and safeguarding passwords."
 - WebInspect Enterprise – Manages the scheduling and polling to identify the availability of a sensor. WebInspect Enterprise also automatically publishes results to Fortify Software Security Center.
- [Security Fortify Application Defender](#) – A runtime application self-protection (RASP) solution that helps you manage and mitigate risk from homegrown or third-party applications. It provides centralized visibility and actively defends production applications from use and abuse while protecting from software vulnerability exploits and other violations in real time.

Finally, be confident that you can deliver vulnerability-free software applications on time by automating and integrating comprehensive AppSec scanning tools into your development process.

Table 2: Your complete solution

Micro Focus and its Partners Offer a Complete Application Security Solution			
Technology Partners		Consulting Partners	
Logo placeholder	Sonatype —Open source governance platform	Logo placeholder	Saltworks Security —Providing an integrated approach to application security
	Secure Code Warrior —An innovative approach to secure coding		

Contact the Micro Focus Fortify team to learn more.

Follow us on Twitter, Facebook, Instagram, LinkedIn