PARASOFT®

WHITEPAPER

Build Security Into Your DevOps Strategy

Addressing security as a top priority in software development drives the move to DevSecOps and ensures a secure DevOps pipeline. The term "shift left" refers to the desire to move critical software activities, such as verification and validation, earlier in the software lifecycle. Finding and fixing defects and security vulnerabilities sooner than later has both cost and risk reduction benefits.

Agile and (more recently) DevOps is typically adopted in the hopes of delivering software faster in an iterative manner that ultimately delivers functionality more aligned with the goals of the end customer. Realistically, with as many as 70% of IT projects failing or falling short of their goals, smart development teams are looking to improve their development practices so they can succeed with a project and create a repeatable process for future iterations and products. This paper investigates the processes and automation needed to improve software security while maintaining—and improving—quality.

It's important to think about the drivers behind the software development goals before jumping into details about the implementation and the Shift Left practices. These drivers may be budget and quality goals in a strict risk management framework. The important consideration here is weaving best software development and security practices in to support your agency's missions.

SUPPORTING EACH AGENCY'S MISSION

When focusing on accelerating delivery to end customers, be it a product or deliverables to other agencies, the first goal is to remove the bottlenecks in the process. The less hurdles in a DevSecOps pipeline, the more efficient each cycle becomes. Verification and validation—and the required testing at all levels of the system architecture—are the key bottlenecks to identify and make more efficient.

Improving the software development and delivery process is a multifaceted problem that juggles seemingly contradictory goals of accelerating development, reducing costs while increasing quality, and reducing security risks. In terms of reducing security risk, this is a further complex problem of improving confidentiality (protecting information), integrity (data can be trusted), and availability (application and data are available when needed).

The focus on security risk, mitigation, and improvement is an ongoing effort in the public sector. These risks can take many forms like leaking of sensitive information (for example, confidentiality) or disruption and denial of service (for example, availability). The challenge is that all these different risks are interrelated between traditional concepts of quality and modern concepts of security. Effectively, they are inseparable, so accelerating deliverables impacts security and quality. There's no point in reducing costs at the expense of either.

The key is to balance all these things together. This paper focuses on security and how organizations can implement a secure DevOps pipeline and "bake security" in from the beginning of the process.

SECURITY IS THE NUMBER ONE CONCERN OF PUBLIC SECTOR ORGANIZATIONS

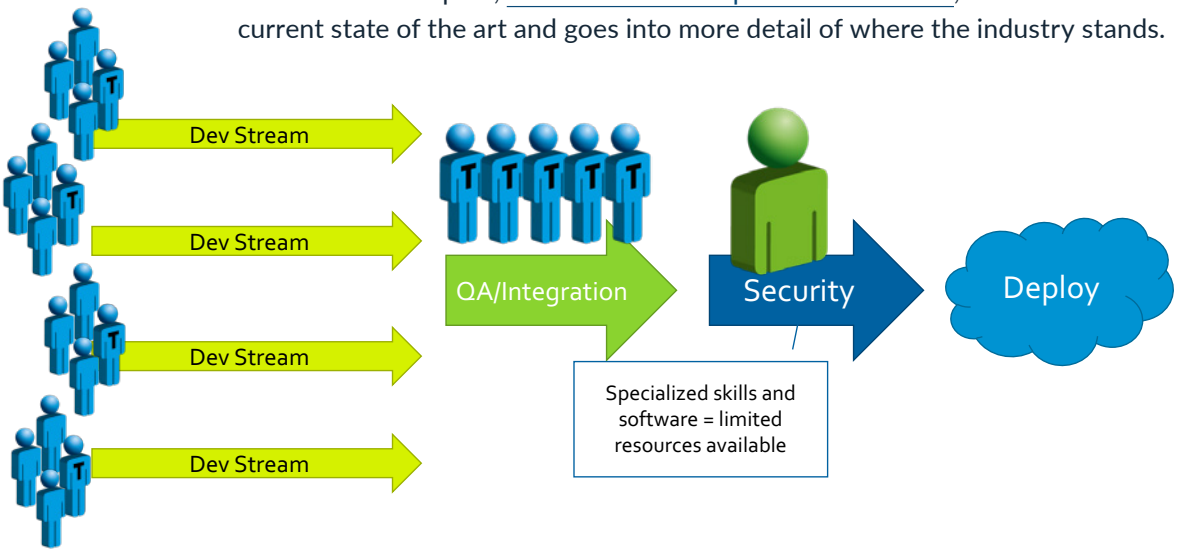
Security is an important concern within public sector organizations. As a result, there are industrial regulations to assure good security and privacy practices such as PCI DSS for payment processing, GDPR for privacy, and HIPAA for healthcare. These are focused on protecting information and making sure confidential information isn't leaked by accident or by attack. In the future, more regulation is likely. California with CCPA and New Jersey's data-privacy bill are additional privacy regulations to consider.

Despite these regulations in public and private sectors, there are still security breaches. Some are high profile. For example, the Equifax breach gained lots of media attention as most of their customers' personal information, including social security numbers, were leaked. As of the first quarter of 2020, there has been over 17 million leaked government records, an increase of 278% over the same period in 2019. The U.S. Small Business Administration (SBA) had a recent data breach that exposed personal details of over 7,000 individuals. Investigation of that data breach revealed the breach a result of an application bug. Unfortunately, these are not isolated incidents and many leaks and breaches slip under the news radar.

The current situation of deploying insecure software and patching it after a high-profile breach is unacceptable. Software developers need a new approach and need to ask themselves: "What is it that we're doing today? Why isn't it working? And what should we be doing to change that?"

THE TRADITIONAL APPROACH IS NOT WORKING

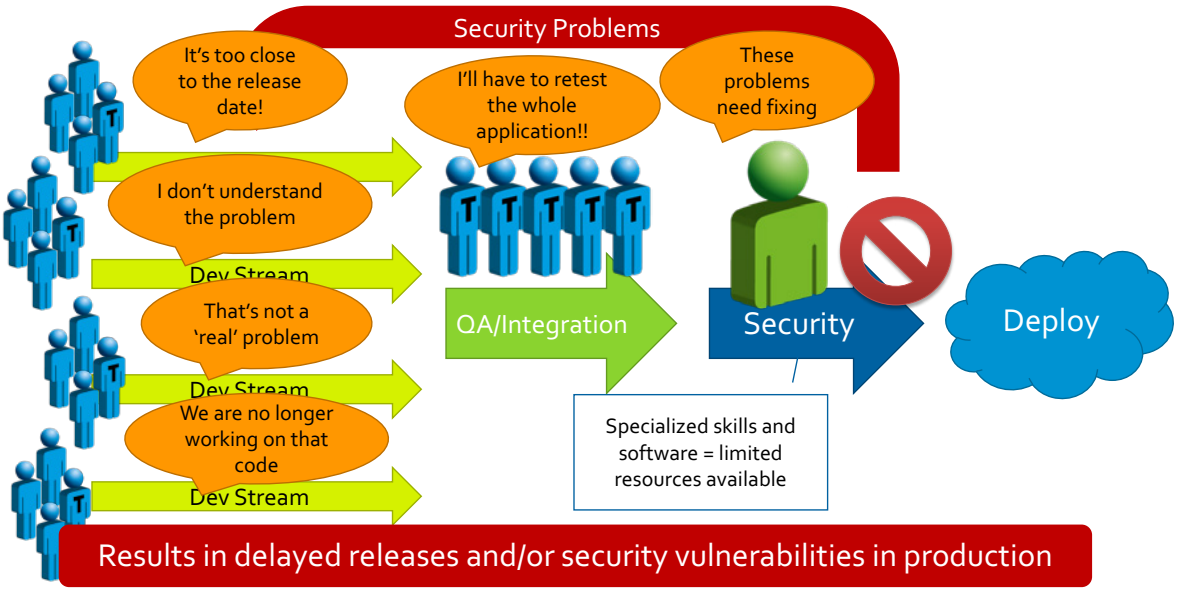
As the above data breaches show, security in public sector agencies needs improvement. Many organizations still, in 2020, are relying on an end-of-cycle security process. Basically, trying to bolt security in at the end of the process. The process if more like Development (Dev) then Security (Sec) then deployment (Ops). A waterfall approach rather than an iterative process. The SANS Institute report, 2018 Secure DevOps: Fact or Fiction?, discusses the current state of the art and goes into more detail of where the industry stands.



There are several challenges with the traditional approach to security. The first challenge is a finite number of people and technologies that can be leveraged at the late stage of the process. Secondly, security inspection required of nearly finished software is a specialized skill. The sheer lack of people that can understand what the security problems are and uncover them, is a large risk to the organization. Thirdly, the security team becomes the gatekeeper and bottleneck to the final release. Security issues and vulnerabilities so late in the process can be difficult to fix because developers may already be reassigned.

Leaving security until the late stages of development is risky and expensive. Issues found are more complicated to diagnose and might be difficult to remedy—it's nearly impossible to improve the security of an application by testing. Vulnerabilities are discovered but might not be fixable at such a late stage. Developers are not always security experts and complex vulnerabilities are challenging to map to problems in the code.

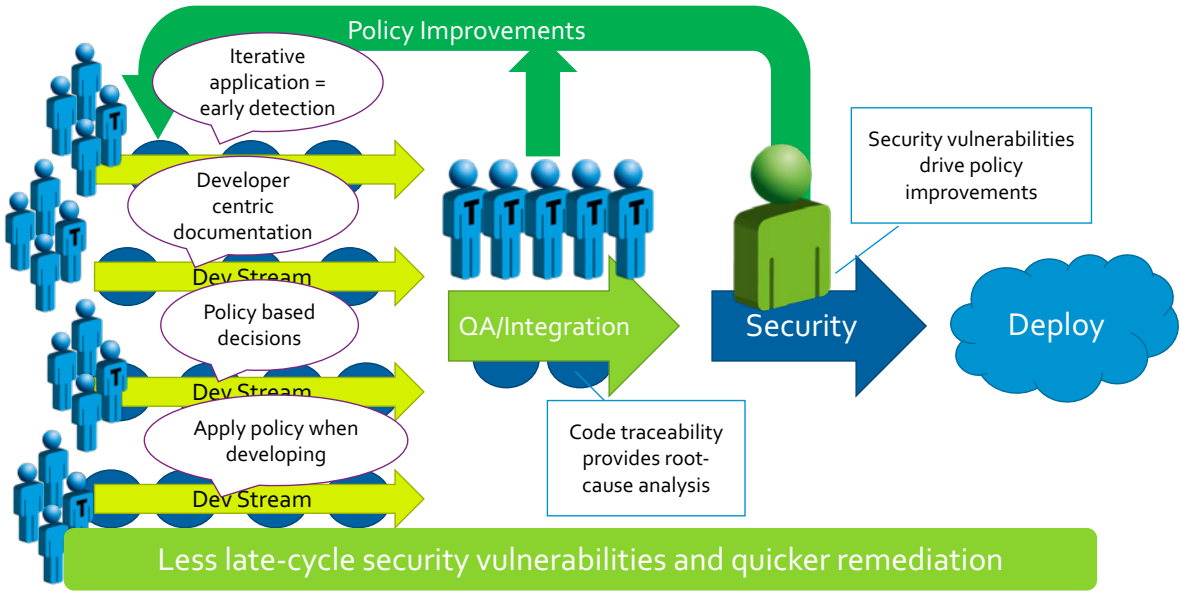
Using automation tools like static application security testing (SAST/static analysis) to find vulnerabilities in nearly complete code base is difficult. Inevitably, it's hard to prioritize and sift through many reported issues so late in the process. In addition, any fixes for issues requires integration teams to retest the whole application. Because security has such a broad global impact, the project ends up with either a delayed release or poor quality and security, which is a more common outcome. In too many cases, software ends up with security issues leaking out into a production environment with a promise to fix them at some point in the future.



THE SHIFT-LEFT APPROACH TO SECURITY—SEC>DEV>OPS

The traditional approach and the problems created are confirmed in the 2018 SANS Institute's report. In response to this, Parasoft proposes that rather than starting security at the end, start with security as early as possible. Shift security to earlier in the lifecycle—with the first line of code—and to left of the development timeline.

This security-first approach means doing more work upfront—such as threat modeling, which feeds into development policies—and eventually defining how the code is constructed. The desired workflow is defined early, as are the testing practices, to ensure the creation of high quality, secure, and reliable applications. Defining processes, tools, and techniques early provides the ability to apply security practices throughout the development process, leveraging different teams to perform appropriate secure coding practices to collectively contribute towards a secure application.



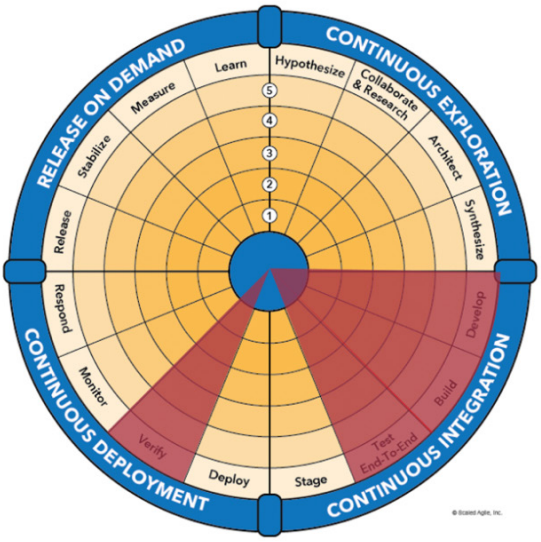
When a security vulnerability is found using a security-first, Sec>Dev>Ops approach, it's caught early during development. It's quicker and cheaper to fix. In addition, if the vulnerability is serious enough, it can drive improvement to the security policy to prevent or identify subsequent occurrences as well as other occurrences in the application. This continual feedback into the process reduces repeat vulnerabilities. It's also important to leverage modern testing practices that the teams may already be using to provide information and traceability, making it easier for the development and QA teams to identify the root causes associated with the security vulnerabilities.

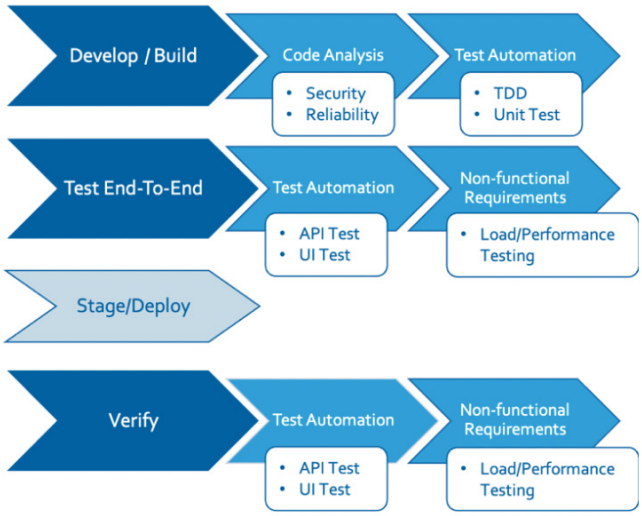
This proposed approach seems reasonable, especially if it provides a developer-centric workflow. Over time, security policy improves based on feedback from developers and testers.

The next question is: "How to make this work?" The rest of this paper discusses how to build a secure DevOps pipeline that not only improves security but also bakes quality into the application with improved efficiency to ensure the ongoing delivery of high quality and secure applications.

INTEGRATING SECURITY AND QUALITY INTO THE DELIVERY PIPELINE

There are different phases of the development process. They are represented differently depending on the audience, but one approach that's gaining traction for helping software teams is the Scaled Agile Framework (SAFe). SAFe is popular in organizations that are trying to scale their Agile practices across an enterprise, beyond individual teams and silos. SAFe identifies these different phases of software development and clearly identifies best practices that are appropriate for each phase (represented as sectors) of the process.

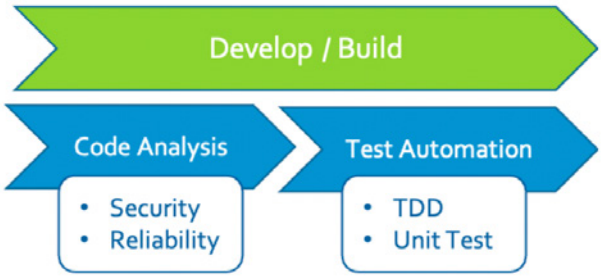




The above diagram shows specific testing techniques indicated on the continuous development cycle representation used by SAFe. These testing techniques are used continuously as part of a DevOps pipeline, leveraging infrastructure that can be created dynamically, that is itself secure and assembled in a secure way. This approach is where containerization along with Kubernetes to orchestrate containers, really shines giving software teams a flexible and powerful framework for deploying quickly and securely. A secure "software factory" from the start.

All development processes start with defining work items, the things that are designed, coded, and tested. They go by many names such as requirements, stories, backlogs, or maybe defects or problem reports. Whatever the terminology is, there is always something that defines a unit of work.

These units of work are defined, uniquely identified, and stored in a repository. For example, Altassian JIRA is very popular with the Agile community. Other examples are VersionOne or RallyDev. Parasoft tools integrate with many tools in this ecosystem including custom integrations with proprietary systems. The bottom line is, once a work item is defined (and uniquely identified and stored), the development teams start working on them within some development process. It is at this very early stage where it's best to start to build security and quality into the application. An effective and efficient way to do this is using static code analysis.



Using SAST as Early as Possible

Advanced static analysis tools (also known as static application security testing or SAST) are more than simple code beautification tools for making sure that source code looks nicely formatted. Modern SAST tools find deep reliability and security issues that are hidden within the code, that are often overlooked by human inspection or unit testing. Automation helps to uncover these hard to find problems. There are open source tools available, but they often lack the deep analysis needed to uncover reliability and security issues. For a detailed discussion, read our whitepaper, How to Choose a Modern Static Analysis Tool. Advanced SAST tools like Parasoft Jtest, C/C++test and dotTEST provide this level of analysis for Java, C and C++ and .NET technologies.

Easing the Adoption of SAST

Static analysis tools ship with many built-in checkers. For example, Parasoft C/C++test has over 4,000 different checkers. This seems like a lot, but these are mapped to different coding standards to make adoption easy when implementing an industry standard. When first using static analysis, it is common to get many warnings, perhaps thousands. This can overwhelm the development team so it's important to ease your way into adopting the tools. More details on adopting static analysis are provided in our whitepaper, Getting Started with Static Analysis.

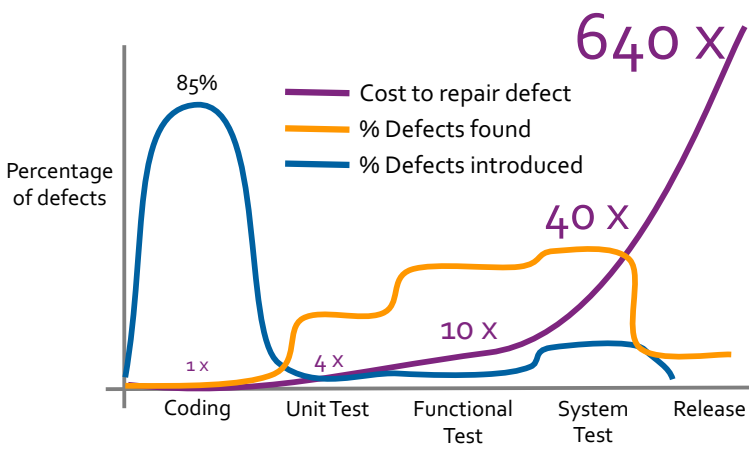
Adopt a Secure Coding Standard

Part of the adoption process should focus on the adoption of a relevant coding standard. For regulated industries, coding standards are already defined (for example MISRA C 2012 or AUTOSAR C++14 for safety critical applications). For other types of applications, it's a good idea to consider coding standards focused on secure code development. The popular options are SEI CERT, OWASP Top 10, and CWE Top 25.

SEI CERT C/C++ differs in that it's an engineering standard that defines the more secure subset of each language to use (MISRA and JSF are similar in this regard) to build secure code in the first place. The other standards such as OWASP and CWE are guidelines of what software weaknesses to avoid and more suited for testing code. Luckily, with whatever choice made, Parasoft tools support the most popular coding standards out of the box with checkers aligned to these standards, allowing for quick creation of compliance reports. For more details on adopting secure coding standards, see our whitepaper, How to Select and Implement the Right Secure Coding Standard.

Shift Testing to the Left

The well-known graph, shown below, based on work by Capers Jones illustrates how defects are introduced during the early phases of the software development process. However, these defects are identified during the late stages of development, often when integrating the application and system testing starts. The costs to repair defects at this late stage are significantly more due to the complexity of debugging the problem, cost of impact of the change, and cost of the number of resources to remediate it.

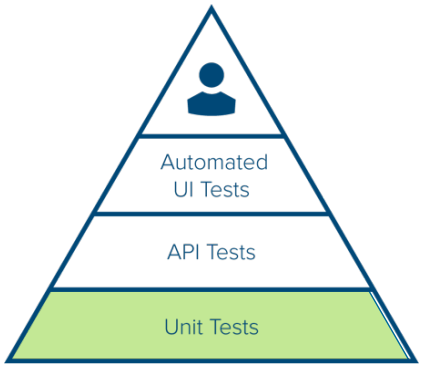


Parasoft aims to help customers reduce the overall cost by shifting left the identification of defects much earlier in the process where they are cheaper and easier to fix. This is the key to integrating security into any development process—techniques and tools needed to address security issues early in the lifecycle where it's both cost effective and lower risk.

Early Introduction of Unit Testing

In the early stages of development, it's important to start unit testing and adoption of test-driven development (TDD) practices. Building upon established open source frameworks (such as JUnit for Java), test automation tools (like Parasoft Jtest) augment this base functionality to accelerate creation and maintenance of unit tests. Note: For C and C++ development where there is no clear solution that supports for both C and C++, Parasoft provides a source-based unit testing framework with Parasoft C/C++test.

A foundation of unit tests ensures that code is functional, well designed, and meets requirements. One measurement of unit testing "completion" is code coverage. This is an important metric, but public sector organizations unduly focus on code coverage, assuming higher is automatically better. Ideally, there should not be a sole focus on code coverage metrics. Rather, the emphasis should be on proactively using unit testing from the beginning, which results in better code, that is easy to maintain, debug, and regression test. Trying to add unit tests later in the process and aggressively increase coverage is difficult and inevitably more work.

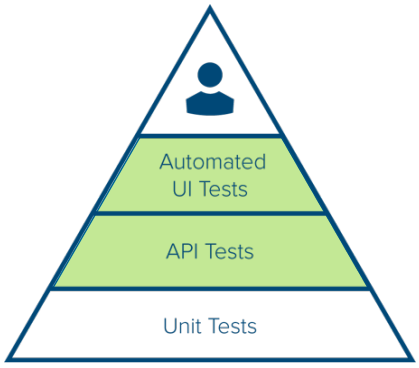


Unit tests form the foundation layer of referred to as the testing pyramid. This is a popular way to describe the hierarchy of testing in Agile development. Unit tests are quick to diagnose, quick to execute, and adopted right from the beginning of the development phase. Unit testing is the ultimate shift-left component to focus the use of tools and test automation. Code coverage is one of many important metrics to measure progress at this level. Equally important is understanding how tests correlate back to the original work items. In other words, the requirements coverage—how much functionality is tested and validated.

Focus on API Testing

Moving further down the software development timeline, end-to-end testing of the assembled application begins. At this point, it's time to start testing different components of it by seeing how they work together. These tests can be deployed in a virtual or production environment, as appropriate. Using functional test automation, it is possible to validate the application at the API level, whether it's REST APIs, web services, or any other protocol message level testing.

Testing at this level can be thought of as use case testing without the need of the user interface. User interfaces are subject to frequent changes and automation at this level is harder to maintain and reuse. API level testing provides the ability to test system functionality with more reusability and less impact from changes at the UI level.



Automating API or service level testing is where Parasoft SOAtest comes in. API tests represent a valuable sweet spot. It turns out API tests are an efficient way for testers and developers to communicate while being a flexible, reusable, and easier-to-diagnose way of automating the validation of business logic. It's also possible to reuse API tests for performance and penetration testing.

UI testing is still very important although manual testing should be reserved for those types of tests best suited for human interaction, such as usability tests. Automating UI testing with Selenium, for example, is a common approach.

Selenium UI test cases suffer from common UI automated testing challenges of maintainability, stability, and long execution times due to the complex and brittle nature of UI tests. Parasoft Selenic improves Selenium testing with automated test creation from observed UI interactions and AI-powered self-healing test execution to make tests less susceptible to UI design changes. The tool offers recommendations, post-execution, to improve UI tests.

Use Service Virtualization to Manage Dependencies

In a complex application, the challenge becomes the test environment. Some types of tests are difficult to support due to missing or incomplete dependencies (like other systems, databases, services.) It might be prohibitive to include production systems in the testing or there may be security or privacy concerns using external services.

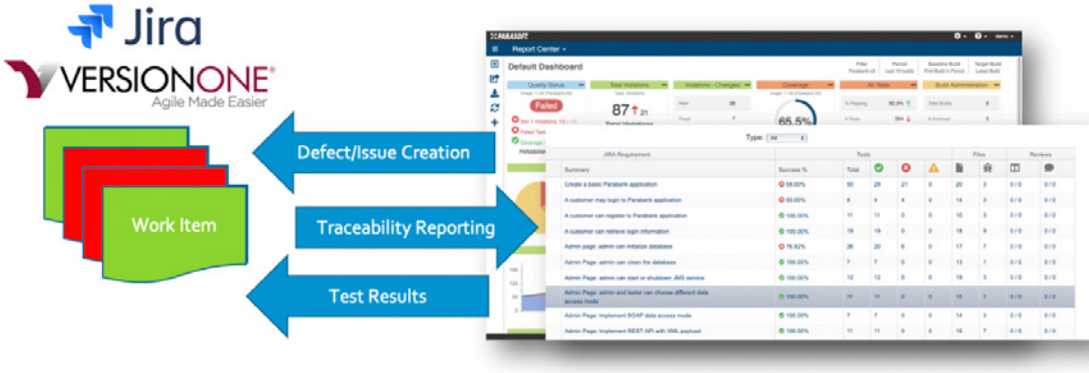
This is where a technique called service virtualization comes in. It's popular with the public sector where it's used to shift-left performance testing—helping organizations do early-stage performance testing as part of their Agile delivery process. There are other powerful applications.

Service virtualization is intelligent simulation of the behavior of a dependency as a stand-in for the real system or service. Despite the use of containers as backend components, they may still need virtualization due to difficulties setting the correct starting state or further external dependencies.

Maintain a Centralized View of Quality

Combining several test automation techniques results in a lot of quality information. This data is useful, but the sheer volume makes it difficult to pause and make some decisions about the application. To support decision making, this data must be aggregated into a centralized view of quality. Parasoft includes a reporting and analytics dashboard as part of Parasoft DTP included in our test automation products. It aggregates data from each of these testing practices and activities, bringing in coding standard compliance reports from our SAST tool, unit test results with code coverage information, and API test results. In addition, it offers cross correlation (traceability) of test results back to the original work items.

As a manager, these graphical dashboards provide a top-level view of testing progress. Additional work items can be created based on review of the data, identified defects or other issues. They also provide a true bi-directional traceability for a full view of your quality initiatives, the DevOps process, and the status of quality and security practices in one place.



INTEGRATING SECURITY INTO DEVOPS

The recommended techniques were introduced and the problem that DevSecOps is tackling was discussed at a high level. This next section discusses how to apply these to build security and quality into the product line—migrating from DevOps to DevSecOps. Before prescribing an approach, it is important to understand the issues that lie ahead. Let us summarize some of the key challenges facing the adoption of DevSecOps.

The SANS Institute 2018 Secure DevOps: Fact or Fiction report has some interesting findings. Less than half of the respondents who responded to the survey used in the report have shifted left their security initiatives. In other words, the other half of organizations are still taking a right-sided approach to security, looking at security just before product release. To summarize the three key challenges:

- **Security knowledge is limited and requires skilled resources/tools.** A common challenge faced by software organization is the shortage of application security personnel and skills. The combination of adding security in the last minute "crunch time" of the process and a lack of qualified personnel means that security is not addressed adequately.
- **End-of-cycle security testing results in delays and vulnerabilities in production.** Many respondents (53%) wait until the product release before engaging their security resources in their organization. That is a concern for many reasons, but it's clearly a resource constraint. Waiting to engage those security resources, probably means that they're not prepared nor ready to be part of the process and certainly not very Agile. In addition, these organization discover that corrective action from security investigations is in the hands of the developers (65%). Meaning the remediation to a security vulnerability is likely in the code developed near the beginning of the project. However, the problems are being identified right at the end of development.
- **Unknown state of security risk until the last moment.** An important part of the SANS institute report is respondents noted a lack of visibility into the state of security of an application. If management does not understand or recognize the need to address security risks, the team isn't given the direction to work on it. Without visibility, software teams are stuck perpetually fire fighting security issues during development. Approaching security in this mode is not going to help improve the process nor accelerate deliverables.

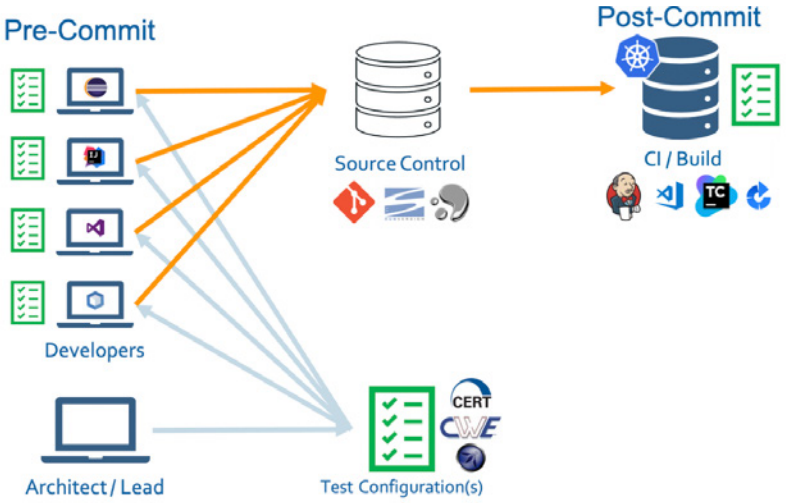
Cross the Security Skills Gap

One of the first places to start in bridging the gap in a software team's security knowledge is right at the beginning of the code writing process with SAST/static analysis. The typical workflow in a CI/CD process consists of the following key elements:

- Developers writing code in their local IDE.
- Developers submitting this code into source control.

Central to the CI/CD pipeline is the CI build process that is powered by Jenkins, for example. There are also the team leads and architects who define the work items, monitor the process and create the policies used by the group.

Setting and implanting the security policy is a key starting point for security by design. Luckily, there are industry standards such as SEI CERT, CWE Top 25 or OWASP Top 10 available to help guide this policy. By ensuring that code is created in compliance with these standards prevents software weaknesses that are the root cause of security vulnerabilities.



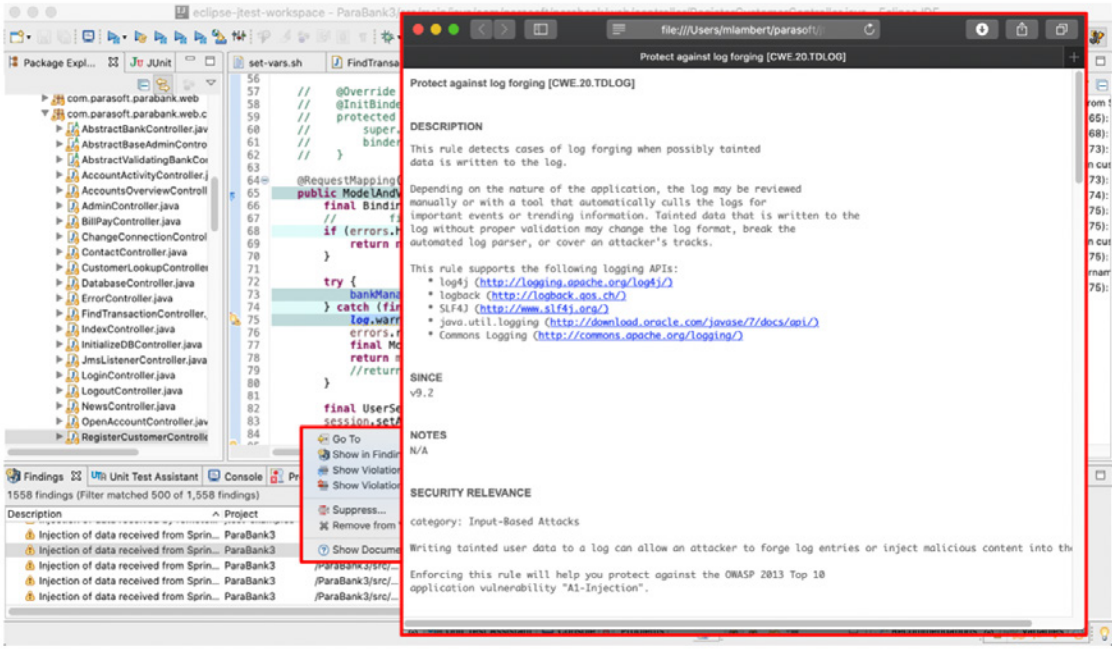
With the Parasoft solution, security policies are edited, stored, and replicated to the team with series of prebuilt test configurations that have industry guidelines enabled. There are also severities associated with each checker, allowing further refinement. New checkers can be created, and existing ones can be extended and customized. There is a framework available for teams to create their standards or design patterns.

It's at this point that software organizations can engage their security team early in the process to define what standards are to be used. They also a play a role in auditing the results to make sure that the standards are applied. Feedback on the policies is used to refine configurations as needed (and propagated out to the team automatically).

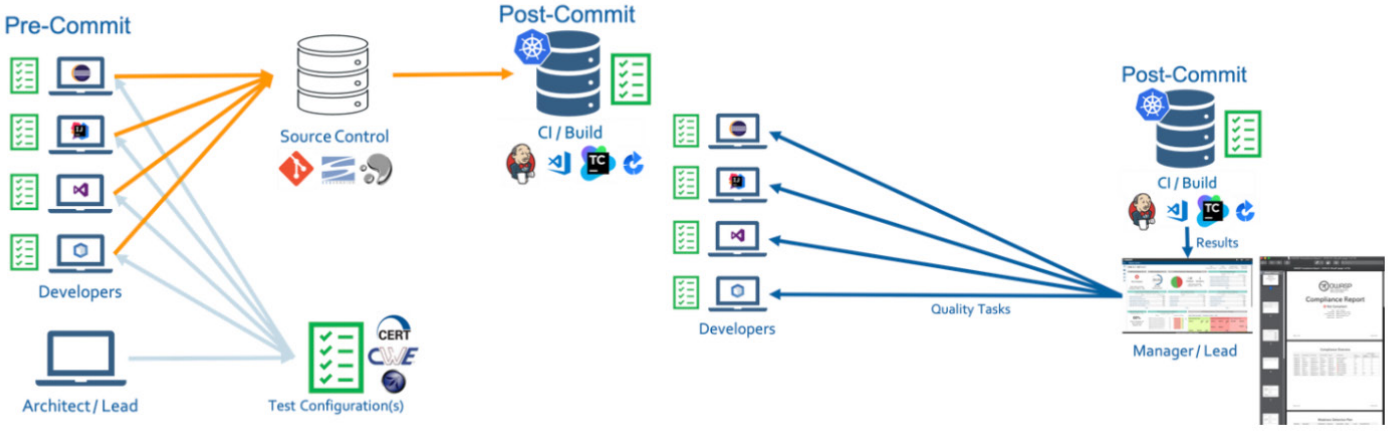
The ability to leverage industry standards with automatic checkers right in the developer's IDE is a quick way to enable the whole team. A critical aspect of bridging the security knowledge gap is how security warnings are handled in the IDE. All checkers have documentation associated with them, which includes examples of a vulnerability, an example of how to repair it, as well as links to relevant hands-on training material—effectively placing developers at the front line for security.

Shift-Left Security to the Developer's Desktop

Security policies need to be part of the everyday workflow for developers so that it becomes habit and part of the expected workload. As part of this, test configurations need to be standardized—each developer works from a common configuration—across the team. These configurations are automatically fed directly to each developer's IDE where they execute analysis locally. Static analysis and test automation need to be supported at the IDE level. Command line availability is important too.



Parasoft tools support a wide range of IDEs including Eclipse, IntelliJ, Visual Studio, and Visual Studio Code. Consider the following workflow example:



It's important for test automation to support both the developer and build/CI use case. This enables the entire team and multiple teams within the organization to leverage the technologies in different ways. Testing is done on the developer's desktop and during a secondary process that runs post-commit. These automated test suites are part of a CI/CD pipeline. The purpose of this process is to gate a "trust-but-verify" process to make sure that nothing slips through.

Shifting left security means developers do the early stage security work in their IDE to catch as many software weaknesses as possible before code enters source control. The CI/CD process is augmented to run a sanity check to capture anything that's potentially missed, but also do a holistic analysis of the entire code base. This approach covers more than the area that the developer was working and catches negative impacts on the rest of the application.

Monitor the State of Security in Realtime

Test automation and SAST generate lots of interesting data derived from the CI/CD process. This data needs analysis and aggregation to provide a useful high-level view, usually in terms of visual dashboards. The dashboards provided by Parasoft tools can be customized depending on the management task at hand. For example, the OWASP dashboard shows the status of the project in terms of compliance with the Top 10 guidelines.

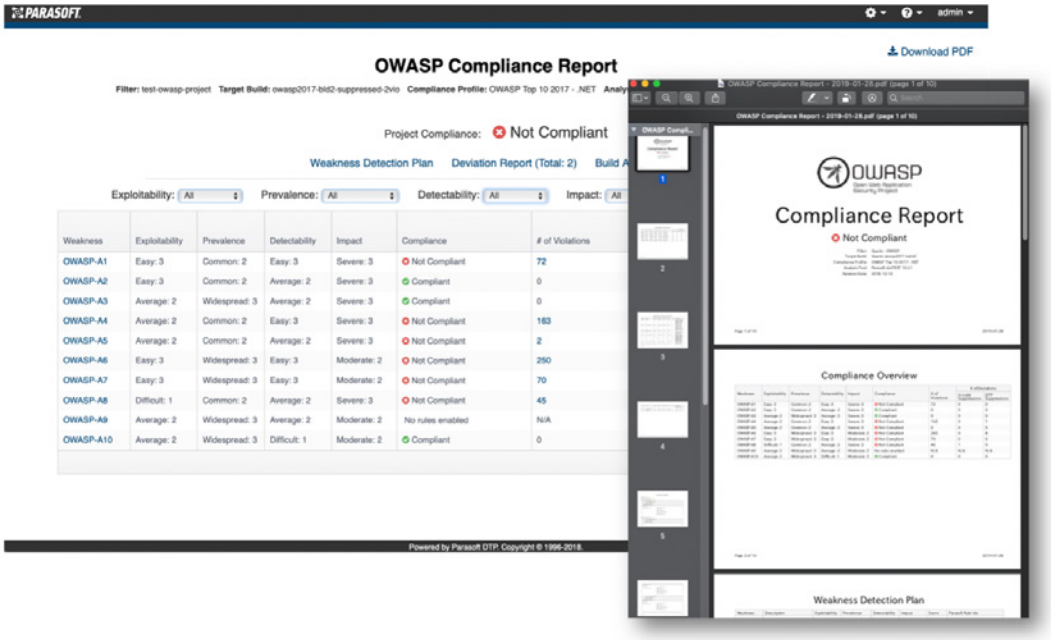
The dashboards are customizable depending on need, such as integrating data from security metrics with other testing practices. For example, the dashboard below may represent a team's live dashboard. This integrates metrics from work items, security data on the right-hand side, both CWE Top 25 and OWASP Top 10 compliance, static analysis violation data, and test results with coverage information. All of that, in one centralized location, is a centralized view of security and quality.

Realtime Compliance Reporting

If compliance to a standard is required by the end customer or a desired goal for a project, it's important that the test automation tools provide the necessary information to demonstrate compliance. Compliance is more than a final yes or no answer at the end of the project. In fact, deviations are inevitable, and the record of these deviations must be recorded with appropriate documentation.

Compliance reporting is something that needs monitoring on an ongoing basis to avoid a mad rush near the end of the project. Just as it's difficult to add security at the end of a project, it's difficult to make software comply to a standard at late stages of development. Automated compliance reporting via dashboards and full form reports are important. An example of an OWASP compliance report from Parasoft is shown below. These reports include support for compliance requirements such as a weakness detection plan, deviation reports, and so on.

As with any test or SAST result, standard deviations can turn into work items for developers. Centralized analysis as part of the CI/CD pipeline is available to all members of the team via web portals and dashboards but also in the developer's IDE. Developers avoid the need to scan for compliance locally. Deviations are imported from the central CI/CD process and new ones can be reported locally and uploaded to the repository. As such, previously discussed workflows apply.



SUMMARY

Security is top of mind for public sector organizations, but they struggle to integrate security into their existing development practices. It's important to think of security and quality together as they are two sides to the same coin. Software teams know that quality, like products meeting requirements while operating reliably, is important. It's difficult to build quality software without requirements or improve reliability at the end of development. It's hard to "test" security into an application. With quality and security considered together, software teams must build it in with the right practices, processes, and tools.

To build in security, it's necessary to analyze and test code earlier in the software lifecycle—shifted to the left. However, there are barriers, which include lack of security skills in the general developer population and lack of visibility into the current state of security of the product. These challenges are overcome using workflows that leverage test automation tools, which support both the developer's desktop and the centralized build process in a CI/CD pipeline.

Centralized data analysis drives the decision making to help focus developers on weaknesses that emerge in the software. The early use of SAST tools educates developers on which coding practices to avoid while preventing insecure code from even entering the code base.

Parasoft's solutions focus on speeding up software quality and security efforts by quickly identifying and fixing issues found during testing—from the start of the SDLC. By integrating continuous quality and security into the CI/CD pipeline, projects can be delivered on time and on budget—with less engineering hours lost waiting for results, searching for issues, and digging through logs.

CONTACT US

Build security into your software development process from the beginning.
Talk to one of our experts to get started today.

ABOUT PARASOFT

Parasoft helps organizations continuously deliver quality software with its market-proven, integrated suite of automated software testing tools. Supporting the embedded, enterprise, and IoT markets, Parasoft's technologies reduce the time, effort, and cost of delivering secure, reliable, and compliant software by integrating everything from deep code analysis and unit testing to web UI and API testing, plus service virtualization and complete code coverage, into the delivery pipeline. Bringing all this together, Parasoft's award winning reporting and analytics dashboard delivers a centralized view of quality enabling organizations to deliver with confidence and succeed in today's most strategic ecosystems and development initiatives—cybersecure, safety-critical, agile, DevOps, and continuous testing.