SYNOPSYS®

WHITE PAPER

Build Security Into Your SDLC With Coverity

Table of contents

DevOps and security go hand in hand

Creating software applications has become an essential part of almost every enterprise. But as organizations create a plethora of new applications, new security threats are emerging, and that presents a growing challenge for enterprises that need to secure their applications against potential attacks. More than half of the vulnerabilities found in applications result from coding mistakes that could have been prevented. So development organizations are creating security programs and policies to find security vulnerabilities in their applications earlier in the DevOps life cycle and minimize the cost of finding and fixing them.

DevSecOps is the next step in the evolution of DevOps: it integrates security into DevOps workflows as an essential component. Embedding security practices in this way helps organizations ensure that they can detect security vulnerabilities early, assess their risks, and take informed actions to fix them. It is no longer economically feasible to employ security testing only at the production stage. Instead, security tools must aid in the software development process, fit seamlessly into the developer's workflow and CI/CD pipeline, and not slow down the software development life cycle (SDLC).

Essential elements of a successful DevSecOps strategy

- **Developer-centric approach.** Organizations must get developers' buy-in on using security testing solutions in their workflow. For example, a static analysis tool must support the programming languages, frameworks, and platforms developers use. It must discover defects accurately, and it must integrate into the developer's workflow, especially IDEs and SCMs.
- **Integration into existing pipelines.** To automate application security solutions in a high-velocity, agile development pipeline, organizations must use tools capable of integrating into the pipeline at many stages, from the developer's desktop, to the build server, to QA testing.
- **Monitoring and managing.** Organizations should use project management dashboards with reporting capabilities to monitor and manage application security during the entire SDLC.
- **Risk assessment and prioritization.** Teams need trusted risk analysis based on security testing to make informed decisions about application risk and prioritize fixes. It's vital to have tools that support various security standards, such as PCI DSS and OWASP Top 10.

These security practices have "shifted left" toward developers, who are becoming more involved in making decisions about security testing tools. Recent community research by an application security tools vendor indicated that mature DevOps teams integrate automated application security testing tools almost twice as often as immature DevOps teams, and that developers are happier with more-evolved security practices in the DevOps process.

Static application security testing (SAST) tools allow organizations to perform white box testing (i.e., look at the application from the inside out), which enables testers as well as developers to find potential flaws in the code by looking at the code using automated scanning tools. In comparison, dynamic black box testing relies on testers' knowledge of an application's capabilities to be able to model every aspect of the application behavior. SAST solutions, on the other hand, can be deployed earlier in the SDLC and therefore can provide useful insights to the developers about security issues even before they commit the code, and complement other security tools in the pipeline by reducing the scope of further testing. Thus, making SAST solutions part of the DevOps process right from the developer's desktop has proven more effective in shipping secure applications faster.

A note on terms: "SAST" and "static (code) analysis" are often used interchangeably. They both refer to the analysis and scanning of program code without executing it in order to find weaknesses and security vulnerabilities.

How Coverity fits into your SDLC

Figure 1 shows a typical CI/CD pipeline. The workflow starts with developers writing code in their IDEs and committing it to the repository. SAST integration at this stage helps prevent security vulnerabilities from entering the code. It also helps eliminate the inefficiencies and inaccuracies of manual code review by identifying vulnerabilities before the code is merged into a repository. IDE plugins and incremental analysis methods make this process even faster, allowing a SAST tool to analyze code as it's being written.

At the coding stage, a SAST solution should identify true-positive issues so that developers don't waste time looking at false positives. This is also when developers are most familiar with their code and can quickly fix issues identified by SAST. By contrast, later in the SDLC, fixes take longer, require more resources to retest for QA, and thus are more expensive for the organization.

It is no longer economically feasible to employ security testing only at the production stage. Instead, security tools must aid in the software development process, fit seamlessly into the developer's workflow and CI/CD pipeline, and not slow down the software development life cycle.

Teams can also add SAST as a gate to their CI/CD pipeline so it can fail the build or create issue tickets automatically if it finds quality and security issues or policy violations in the code. This way, pull requests or product releases can be blocked if certain types of issues are present in the code. SAST scanners can also perform enhanced security compliance tests later in the CI/CD pipeline.

A cautionary note is that SAST scanners are notorious for not being "developer-friendly" because of high false-positive rates. It is important that SAST tools find ***real*** security issues and not waste developers' time dealing with the false positives.

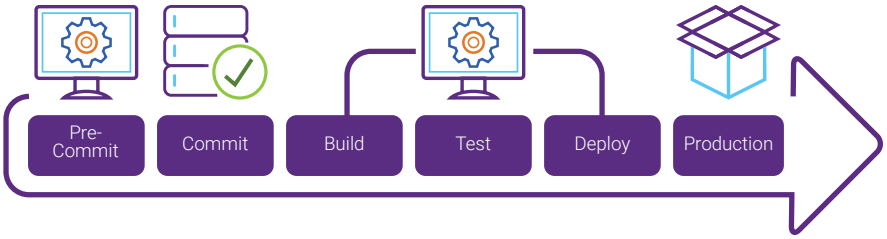


Figure 1. A typical SDLC pipeline.

Coverity®, provided by Synopsys, is a state-of-the-art SAST solution that gives developers and development managers exactly what they need. Developers get agility, ease of use, and accuracy so they can create secure, high-quality applications. Organizations get scalability, issue management, and risk analysis capabilities, along with compliance to industry standards. Coverity integrates seamlessly into the developer's workflow and the organization's CI/CD pipeline.

With the Code Sight™ IDE plugin, Coverity identifies critical quality defects and security vulnerabilities right at the developer's desktop, as code is being written, even before unit testing. Furthermore, integrating Coverity into the CI/CD pipeline, using either native plugins or simple scripts, helps developers and development managers find and fix vulnerabilities early in the SDLC. Coverity's compliance and vulnerability reporting provide a high-level picture of the risks associated with applications and help security managers make informed decisions regarding application deployment.

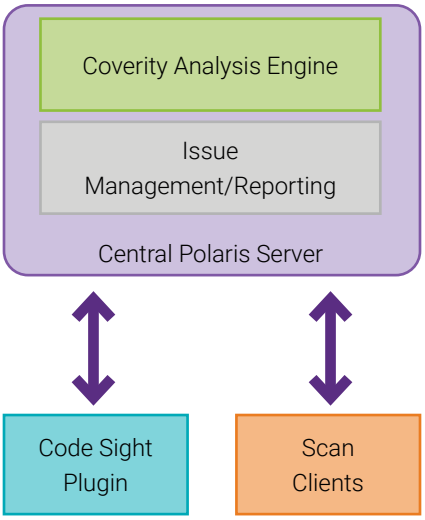


Figure 2. Simplified architecture of Coverity deployment on Synopsys' cloud-based Polaris Platform™.

There are three key components of the Coverity architecture: the developer interface (the Code Sight IDE plugin), the scan clients, and the central server that hosts the analysis engine and the server for issue management. A scan client is where the scan is triggered from, for example, a CI plugin or a desktop scanner. A simplified architecture diagram is shown in Figure 2. For enterprise web and mobile application development, some users will want to deploy Coverity using Synopsys' cloud-based Polaris Software Integrity Platform™. In Polaris, Coverity components are deployed in the Google Cloud infrastructure, with each instance uniquely issued to a customer and isolated in a containerized environment. For organizations not looking to deploy Coverity in the cloud, an on-premises deployment option that includes the Coverity Connect portal is also available. The same Coverity analysis engine is used for both on-premises and cloud-based deployments. Developers work with Coverity through the Code Sight IDE plugin, and all Code Sight IDE plugins are available for developers for both cloud and on-premises deployments.

How does Coverity work?

Static analysis involves a variety of techniques to anticipate outcomes without actually running the application. Similarly, Coverity uses patented analysis techniques that provide deep and accurate analysis. These techniques evolved from an understanding of real-world code and design patterns and include highly accurate dataflow, control flow, and semantic analysis, among others.[1] Using these

techniques to perform full-path and control analysis, Coverity can accurately identify code that would result in security and reliability issues and recommend remediation steps. Coverity's full-path interprocedural analysis uses a vast library of quality and security checkers to examine potential execution paths and outcomes. This analysis technique can detect issues spanning multiple files, enabling comprehensive analysis of complex code bases. Additionally, one of the greatest strengths of Coverity analysis is that it not only identifies pattern-based issues but also has a validation component that collects evidence for the presence of a defect. This greatly improves accuracy and minimizes false positives.

Critical issues that Coverity can identify include buffer overflows, SQL injection, resource leaks, and cross-site scripting. To ensure software quality, Coverity identifies potentially overlooked compilation warnings, difficult-to-detect concurrency issues, and potential resource leaks and race conditions for some languages. Additionally, Coverity can help identify policy violations based on supported standards or custom rulesets defined by users. Better security compliance from developers helps security executives improve operational efficiencies and release velocity, which translates into increased ROI for application security solutions.

Coverity capabilities

Language and security coverage

Coverity supports 21 languages (Figure 3) and more than 70 frameworks, including common systems and desktop application languages (e.g., C, C++, C#, Java), and web and mobile application languages (e.g., JavaScript, Swift, Kotlin, Go, .NET, Ruby, VB.NET, Node.js). Language versions are updated often so Coverity stays current.

Java is used for both desktop and web and mobile applications, so a SAST solution must be able to identify common and high-risk vulnerabilities in Java applications before deployment. For example, a SAST tool should be able to identify the security weaknesses in the OWASP Top 10, a list of the most critical weakness types in web application development, and the associated common weakness enumerations (CWEs). Coverity provides broad, deep support for identifying these weaknesses. And Coverity supports all except A9 from the OWASP Top 10 categories,[2] with multiple checkers for each category. For example, for category A5 (Broken Access Control), Coverity uses almost 100 sophisticated, unique checkers to identify various issues in Java applications. Using the CWEs associated with Coverity checkers, organizations can also build their own standards and taxonomies.

Coverity also supports up-and-coming languages such as Swift, Go, and Kotlin. Go has become a developer favorite because it's fast, easy to use, excellent in concurrency, and easy to maintain, and it runs with garbage collection, so memory not in use can be reclaimed. Stack Overflow's 2019 Developer Survey puts Kotlin among the top five most-loved languages, with Swift coming in sixth. As adoption of these languages in the enterprise software ecosystem proceeds quickly, application security testing solutions, particularly SAST solutions, must support them. That's why Synopsys ensures that Coverity can support these languages and their newest versions to help developers and their managers develop and ship secure, high-quality applications.

Figure 3. Languages supported by Coverity 2020.06.

Frameworks and compilers

Frameworks provide reusable software environments, as well as some generic built-in functionalities and flows, that allow developers to build and maintain software applications more easily. For a comprehensive SAST tool like Coverity, a deep understanding of framework semantics is essential to maintain a low false-positive rate.

Coverity supports 25 frameworks for Java and 27 client- and server-side frameworks for JavaScript. In addition, Coverity supports several other JavaScript frameworks, including cloud API frameworks for cloud-native JavaScript, and frameworks for other languages, such as C# and Python. It also supports popular Spring and Struts frameworks for Java, and even up-and-coming ones such as Passport for JavaScript. With this expansive framework support, Coverity can provide more accurate and more comprehensive security vulnerability identification in web applications.

For compiled languages, Coverity wraps around the native build to enable build capture analysis. Different compilers that build applications optimize program flows in different ways. For some low-level languages with highly complex build environments, such as C/C++, a SAST tool must parse the make files and build system to understand and capture code accurately. To understand the application flow for compiled languages and compiler operations and optimizations, Coverity supports about 50 compilers for compiled languages. The Coverity datasheet has an overview of supported compilers and frameworks, and the Coverity Installation and Deployment Guide provided to Coverity customers via the Synopsys community page has a detailed list of supported versions.

Integrations

As mentioned earlier, SAST tools should not slow down DevSecOps and should integrate seamlessly into the CI/CD pipeline. Every organization has its own set of processes and tools to manage DevOps, and a comprehensive set of APIs enables teams to easily integrate Coverity and Polaris into the DevOps process flow.

As shown in Figure 1, the DevOps process flow starts at the developer's desktop. Code Sight integrates seamlessly into the IDE to improve productivity and eliminate security and quality issues as developers are writing code.

Organizations can also integrate and configure Coverity to scan automatically as the CI server is building the code. This way, Coverity can be used as a quality gate for the project and fail the build if the project violates certain policies. It can also be configured to notify the developers responsible for such policy violations, and even automatically create issue tickets through integrations with bug-tracking systems such as Jira and Bugzilla.

Figure 4 shows Coverity's SDLC integration capabilities via the Polaris platform.[3] There are similar integrations available for on-premises deployment as well. Synopsys professional services and field teams can also help users create custom integrations—for example, pull request integrations with Git and other tools.

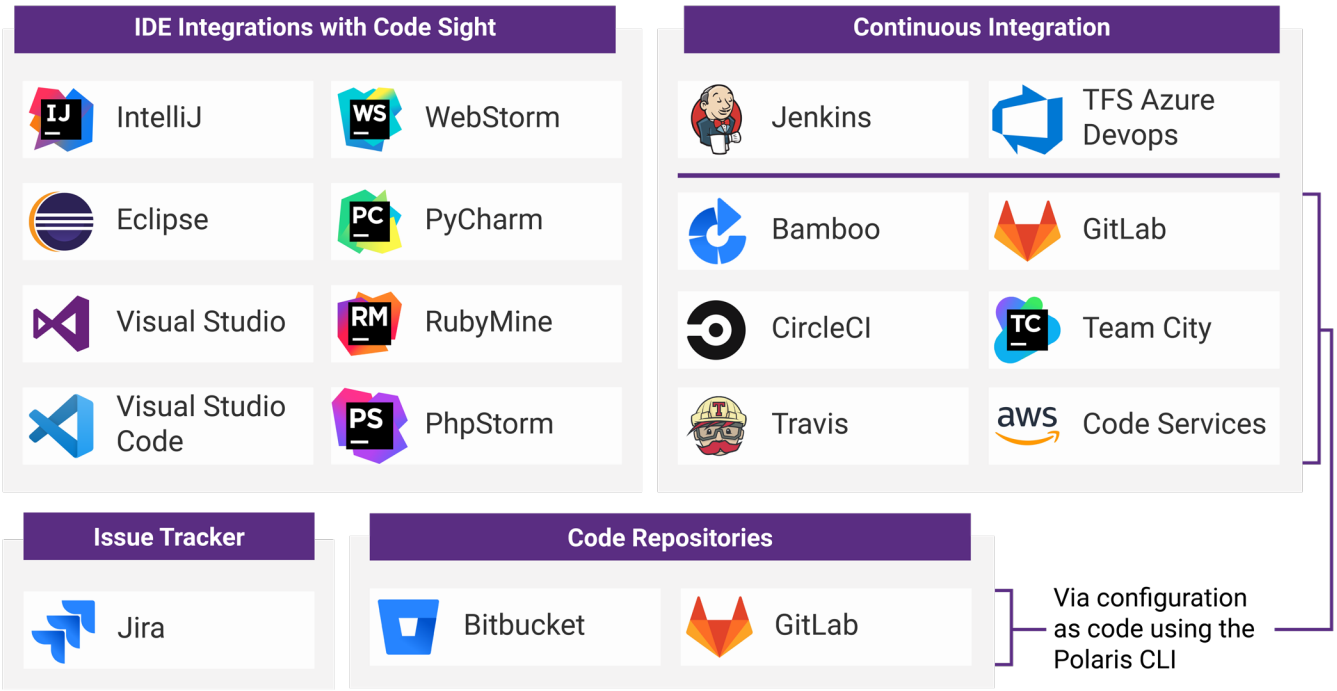


Figure 4. Coverity SDLC integration capabilities via the Polaris platform.

Vulnerability identification

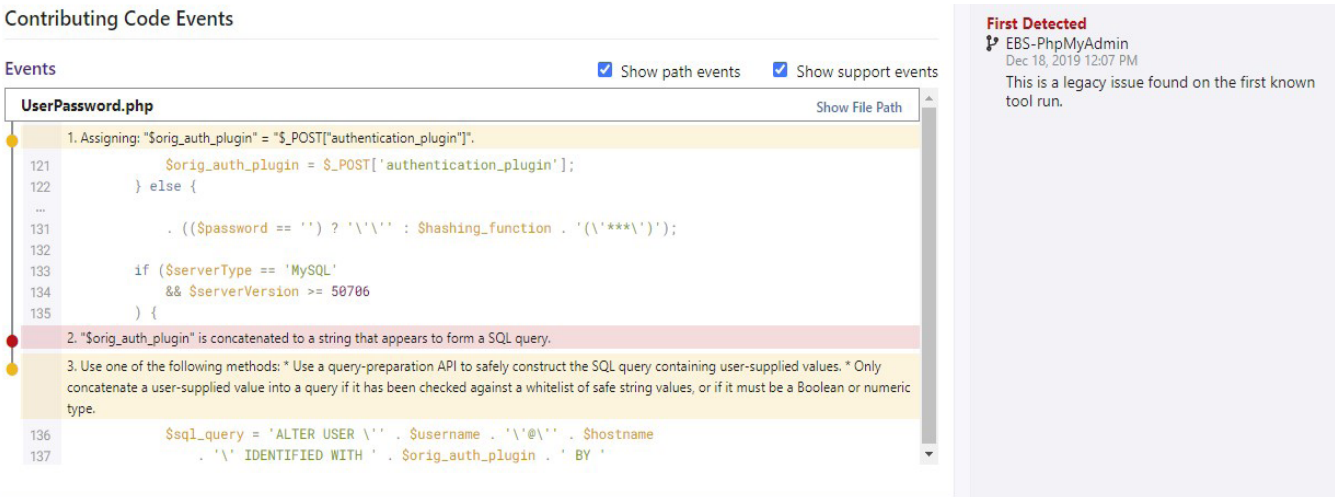
Some of the critical security vulnerabilities that Coverity can identify include:

- API usage errors
- SQL injection
- Cross-site scripting
- Insecure data handling
- Hard-coded credentials
- Buffer overflows
- Cross-site resource forgery
- Memory corruptions
- Path manipulation
- Security misconfigurations
- Concurrent data access violation

Example 1: SQL injection

Let's look at SQL injection, one of the most vulnerable security issues in web applications (see this list of examples). A SQL injection attack occurs when an attacker inserts tainted user data into a SQL statement that has a set of safety requirements or obligations. Figure 5 shows a snippet of a PhpMyAdmin application program block containing a possible SQL injection vulnerability. The 'orig_auth_plugin' variable is assigned a user-supplied value that might return an untrusted data and concatenates to form a SQL query in an unsafe manner. If the application doesn't correctly sanitize this insertion, the tainted characters could change the statement to something unintended by the developer. Executing the statement could affect the confidentiality, integrity, and availability of the database system and enable the attacker to expose information that should be restricted.

Figure 5. PHP code snippet highlighting a possible SQL injection issue in a PhpMyAdmin application.



Coverity identifies critical issues such as SQL injection in the application and offers remediation guidance. It also provides context-aware sanitizer libraries that understand the context related to unsafe data outputs as well as the appropriate sanitization methods and relevant technologies in use in the code. Thus, Coverity can produce actionable remediation guidance based on the information it knows about the program and help developers fix the issue quickly and accurately. For Java and C#, Coverity also offers an open source sanitizer library that includes sanitizers for difficult contexts not usually available in other libraries.[4]

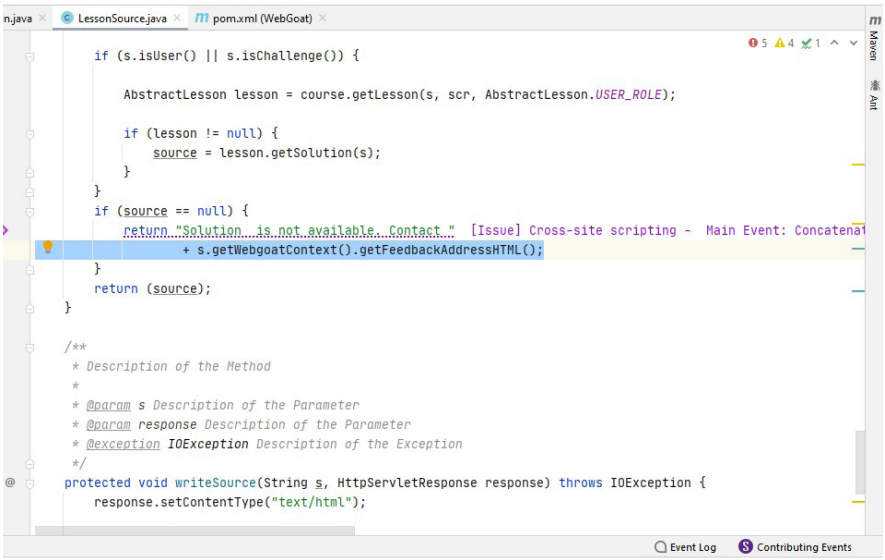
Example 2: Cross-site scripting

Identifying cross-site scripting (XSS) issues and providing context-aware remediation advice are also Coverity's strengths. According to the OWASP Top 10 website, XSS is the second-most-prevalent issue in web applications. The vulnerability allows an attacker to inject malicious, executable script code into an application or website. Without proper data sanitization, the script code could be executed by the browser of an unsuspecting user visiting the website. Potential consequences of such XSS attacks include an attacker stealing the user's keystrokes or cookie information, or redirecting the user to a malicious website.

To prevent XSS, an application must perform context-aware parsing and sanitization of HTTP requests. As explained in this blog post, input validation close to the data source can help protect against the exploitation of this vulnerability, but it will not remove the defect. Context-aware sanitization of the input—specifically, understanding how the data is being used to nest and escape HTML in the right order—near the data sink is necessary to remove XSS vulnerabilities.

Figure 6. An XSS vulnerability example in a WebGoat-Legacy application, before remediation (top) and after remediation by escaping the HTML (bottom).

Coverity detects XSS vulnerabilities and offers libraries to help developers add context-aware sanitization. As shown at the top of Figure 6, the WebGoat-Legacy application shows



The unsanitized 's.WebgoatContext().getFeedbackAddressHTML()' results into an XSS defect.

The Escape library is imported in the source file.

The Escape library imported from Coverity is used to escape the HTML and remedy the defect.

an XSS vulnerability due to the unsanitized '*s.getWebgoatContext().getFeedbackAddressHTML()*'. To eliminate this defect, the application must sanitize the input at the location where it's used (the sink). As shown at the bottom part of Figure 6, a developer can use Coverity's Escape library (imported, as shown in the middle part of Figure 6) to escape the HTML and remedy the defect.

Coverity components

There are three main components of the Coverity ecosystem:

- Developer interface: IDE plugin
- Analysis: Coverity Analysis engine
- Issue management and reporting interface: Polaris or Coverity Connect dashboard

IDE plugin

Developers interface with Coverity through an IDE plugin. Coverity provides IDE plugins for many of the common IDEs that developers use, such as Eclipse, and also for less-common IDEs, such as WindRiver Workbench. Code Sight is our latest IDE plugin for Coverity and other SIG tools, such as Black Duck® (Figure 7). Code Sight identifies security issues using incremental scanning techniques while developers write code, but it doesn't require them to switch tools or manually invoke a scan. By keeping developers in their familiar IDE environment, Code Sight improves productivity over nonintegrated scanners. It also provides remediation advice and context-aware training through its eLearning integration.

Integration. Code Sight integrates with most commonly used IDEs, including IntelliJ IDEA, Visual Studio, and Eclipse. When Code Sight is installed, it detects Synopsys tools based on your product subscriptions, and downloads the relevant engines in a matter of seconds. No further configuration is needed.

For SAST analysis, Code Sight downloads the Coverity Analysis engine, which runs a high-fidelity file-level analysis behind the scenes every time a file is opened or saved. Developers can specify the scope of results shown, from one file to all open files. For each issue found, Code Sight provides a dataflow trace, including the line numbers of the main event and supporting events, to help developers understand the scope of the issue and find all problem areas in the code.

Code Sight automatically syncs with the Coverity server, whether deployed on Polaris or on-premises, so the baseline for incremental analysis is synced automatically from the central server to the developer's desktop.

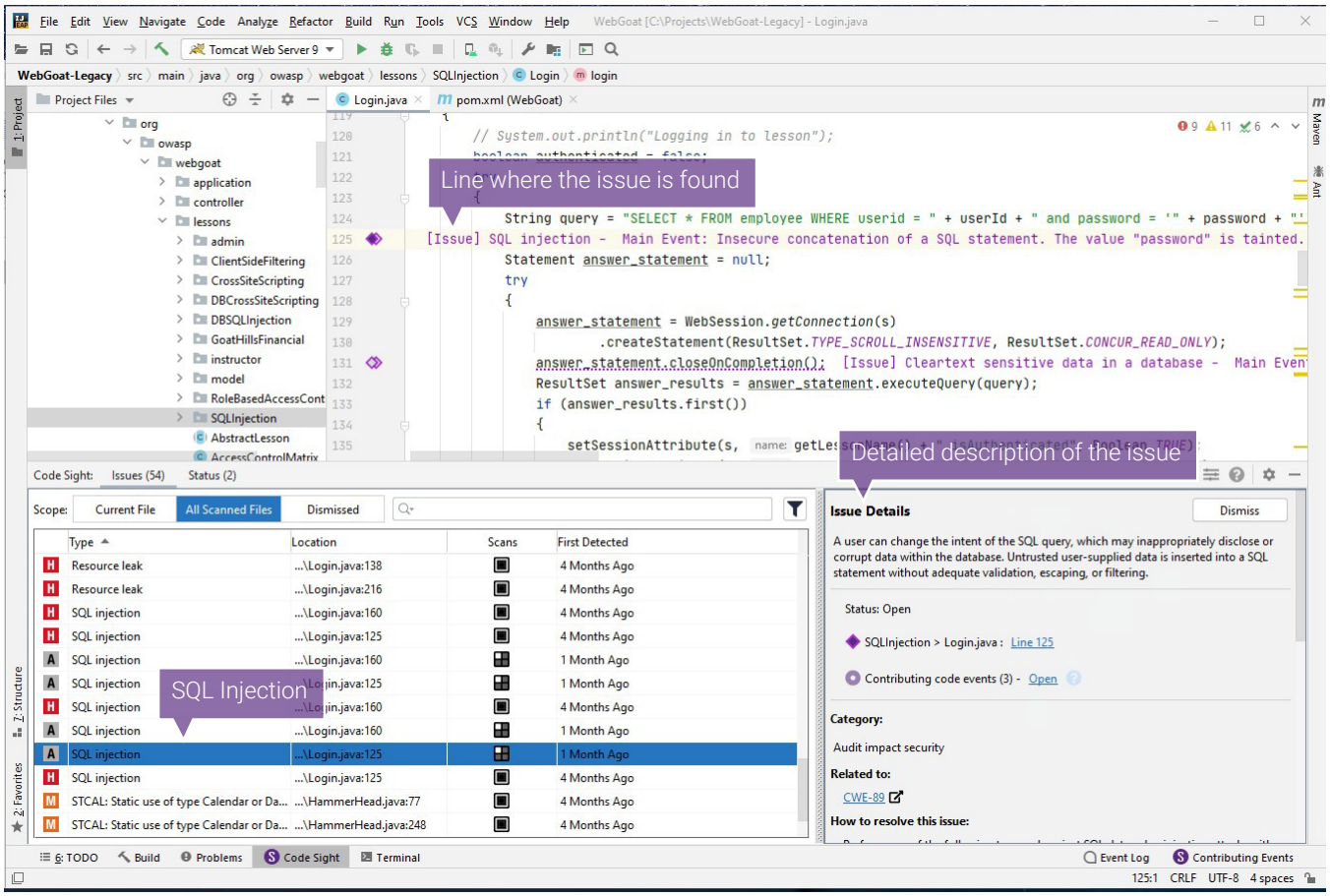
Remediation guidance. Code Sight integrates with relevant eLearning courses to help developers get training related to security issues and best practices right when they need it. When developers view an issue in Code Sight, they'll get links to relevant eLearning courses that help them understand the context of the vulnerability and take steps to develop more secure coding practices. This capability helps organizations nurture a culture focused on weaving security into software design, development, and testing.

Coverity Analysis engine

The Coverity Analysis engine is the horsepower running behind the scenes to find security and quality defects in the code. Coverity Analysis supports both build capture and buildless capture of issues, depending on the language. This way we support easy integration into an in-band development pipeline (via build capture) and an out-of-band security pipeline (via buildless capture).

Build capture. For compiled languages, such as C/C++, Coverity supports build capture and wraps around a native build to understand native compiler operations and optimizations. This functionality helps the Coverity Analysis engine understand information about dependencies and build-related programs.

Figure 7. Viewing issues found by Coverity in the developer's IDE through the Code Sight plugin.



Buildless capture. Buildless capture analysis requires minimal user knowledge and no build. It's geared toward security teams that don't have access to development or build tools to quickly get analysis. Users only need to provide the location of the project on disk or in a Git repository. Buildless capture also allows users to limit the analysis to a specific set of files and exclude other files, such as test cases for library code, libraries the code doesn't specifically require, and so on.

Coverity supports buildless capture for C# and JavaScript projects. For Java projects, Coverity can perform analysis in either mode.

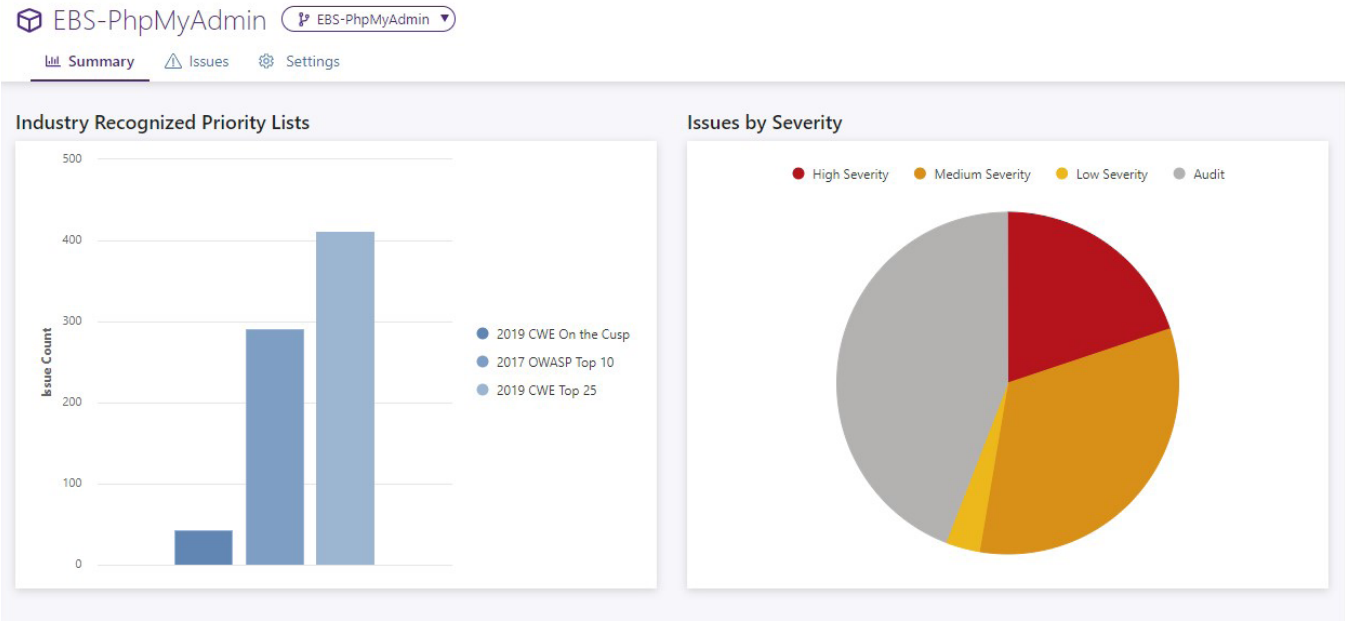
Dashboards for cloud and on-premises

Once Coverity Analysis finishes scanning the source code, users can view and manage the issues it found through the Polaris or Coverity Connect web portals, or the IDE plugins. As described earlier, Code Sight and other IDE plugins help developers fix issues as they code. The Polaris and Coverity Connect web portals let developers, development managers, and security managers view and manage the issues Coverity finds. Additionally, both web portals provide a range of high-level reporting capabilities, which can help users prioritize issues, assess risks associated with the applications, and present succinct findings to management executives.

Polaris is the central platform for the Synopsys application security portfolio. It enables development and security managers to triage and manage all their application security testing results on one platform, and it is the way Synopsys recommends that organizations deploy Coverity for cloud deployments. Figure 8 shows the Polaris dashboard and reporting platform.

For on-premises deployments, the Coverity Connect web interface is available to view and manage analysis results and generate reports via the Policy Manager™.

Figure 8. The Polaris dashboard for issue management.



Compliance reports. In addition to managing projects, development and security managers also need to generate compliance reports for a variety of industry standards (e.g., OWASP Top 10), assess risks, and prioritize attention to specific findings. The Polaris dashboard and the Coverity Connect web portal provide highly intuitive reporting capabilities so stakeholders can accomplish their tasks. These reports help organizations create priority lists based on their risk assessment and focus on the issues that matter the most for them. For example, financial service organizations need reports for PCI DSS compliance, whereas organizations developing web applications are more interested in risks relating to OWASP Top 10 categories. And both groups can use the Polaris dashboard or the Coverity Connect portal to generate reports and assess and manage risks.

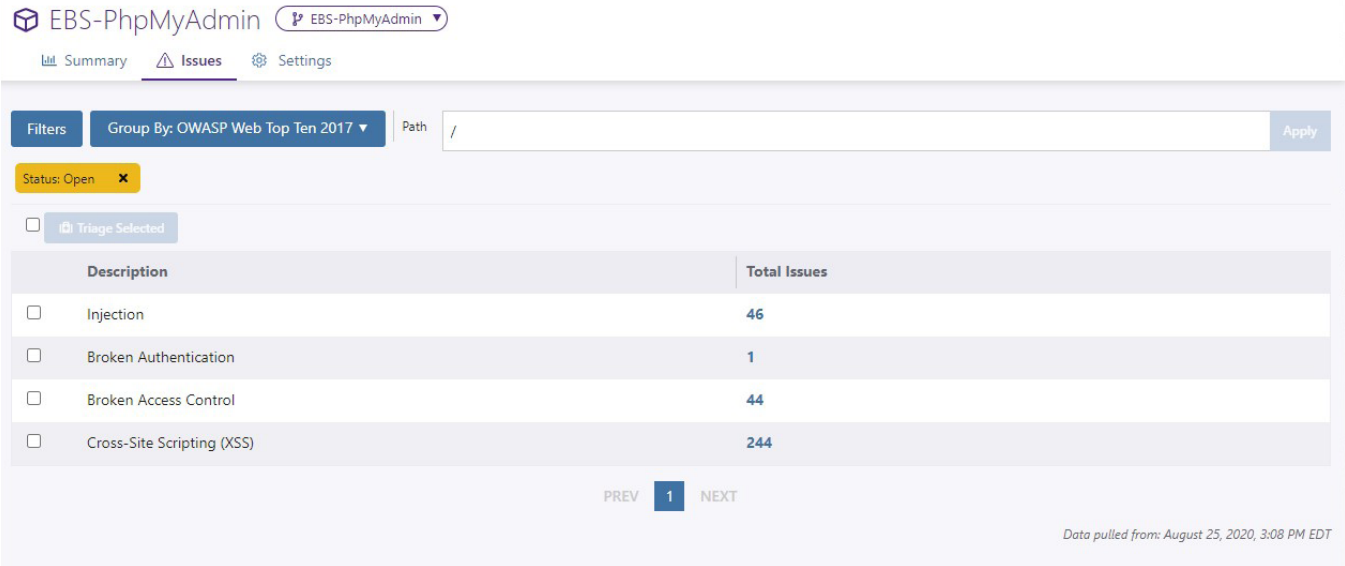
Trend reports. Each project has an issue trend report that is updated with each analysis, so managers can view a trend graph and track the progress made on issues. Security managers can use trend graphs to assess the organization's progress toward reducing security risks and demonstrate the ROI for Coverity.

Figure 9 shows how the Polaris dashboard can also filter issues by standard and generate reports for executives. Users can

manage all projects through this dashboard and view issue types and their risk scores.

Similarly, Figure 10 shows an example of OWASP Top 10 category issues displayed as a pie chart in the Coverity Connect Policy Manager.

Figure 9. Filtering issues in OWASP Top 10 categories for risk assessment, reporting, and prioritization.



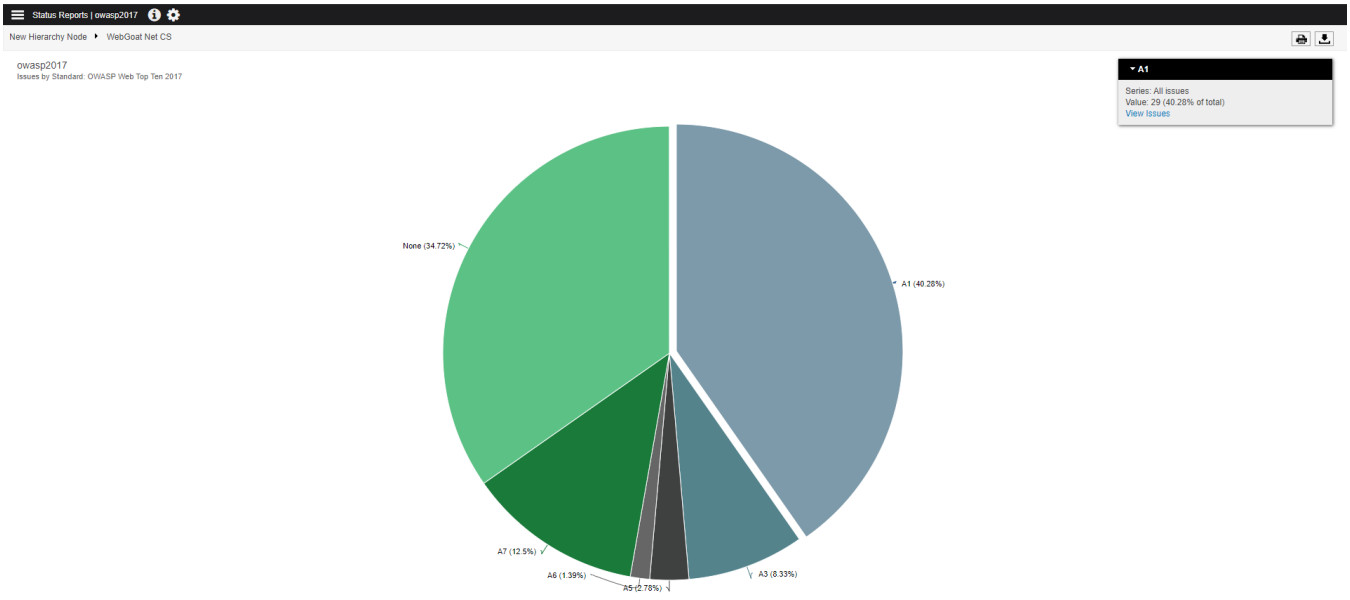


Figure 10. OWASP dashboard in Coverity Connect.

Summary

Coverity provides developers with a comprehensive SAST solution that offers the agility, ease of use, and accuracy they need to write high-quality, secure code without leaving their development environment. With the Code Sight IDE plugin, Coverity seamlessly integrates into the developer's environment, aids the software development process, and helps developers sign off on code changes efficiently. High-fidelity incremental analysis in Code Sight and real-time results help developers stay agile in CI/CD workflows without compromising the accuracy of their code.

Coverity's support for a wide range of languages, frameworks, and compilers, as well as its integration with other stages of the CI/CD pipeline, enables users to onboard a wide variety of projects and employ Coverity as a quality gate. Its high accuracy, low false-positive rate, and fast analysis help development and security teams save time and resources and accelerate software development.

Deploying Coverity using Synopsys' cloud-based Polaris platform allows enterprises to employ a SAST solution with the scalability of a public cloud environment. The Polaris dashboard and reporting capabilities give users a highly intuitive interface to manage projects, assess associated risks, and generate issue reports for executives. Development and security managers can use status and trend information to measure the impact of changes without disrupting the development cycle. Similar dashboarding and reporting capabilities are also available via the Coverity Connect portal for on-premises deployment.

To learn more about Coverity or request a demo, please visit the Coverity SAST webpage.

Endnotes

1. For more information, see "A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World" [https://cacm.acm.org/magazines/2010/2/69354-a-few-billion-lines-of-code-later/fulltext].
2. The OWASP Top 10 category A9 (Using Components With Known Vulnerabilities) is covered by Black Duck, Synopsys' software composition analysis tool.
3. For Coverity's native, out-of-the-box integrations, please refer to Coverity datasheet.
4. Sanitizer libraries can be found at https://github.com/coverity.

SYNOPSYS®

The Synopsys difference

Synopsys helps development teams build secure, high-quality software, minimizing risks while maximizing speed and productivity. Synopsys, a recognized leader in application security, provides static analysis, software composition analysis, and dynamic analysis solutions that enable teams to quickly find and fix vulnerabilities and defects in proprietary code, open source components, and application behavior. With a combination of industry-leading tools, services, and expertise, only Synopsys helps organizations optimize security and quality in DevSecOps and throughout the software development life cycle.

For more information, go to www.synopsys.com/software.

Synopsys, Inc.
185 Berry Street, Suite 6500
San Francisco, CA 94107 USA

Contact us:
U.S. Sales: 800.873.8193
International Sales: +1 415.321.5237
Email: sig-info@synopsys.com