Integrating Application Security into Software Development

Jeffery Payne
Coveros

Building secure applications is the next challenge in application security. Unfortunately, application security initiatives are usually the domain of security organizations that don't understand software development or that can't influence the software engineering practices of their software development groups. Due to this chasm, it's difficult to transition application security initiatives from identifying vulnerabilities after software has been produced to proactively mitigating vulnerabilities during the entire software development process.

Key Approaches and their Pitfalls

Many approaches attempt to cross this chasm. The following three key approaches have pitfalls that should be understood and mitigated to make them successful.

Approach #1: Integrate Security into the Software Process

In organizations that follow a defined software development methodology, security groups often attempt to incorporate security "touchpoints" into the software process and enforce their use as part of a security policy.[1] Touchpoints added to the software process are typically either new software assurance activities (such as Web application security testing) or suggestions for building more secure software (such as defensive programming guidelines). Several pitfalls exist for integrating security into a software process.

Lack of application security knowledge. Most software engineers today know that software applications have security vulnerabilities that they need to eradicate. Unfortunately, they often lack the knowledge and training to effectively build security into their software. Attempting to enforce a secure software development standard when the software development team hasn't been trained won't work. For an application security initiative to succeed, you need to incorporate the appropriate training.

Too much change too fast. For most organizations, rolling out a secure software development process is too much change occurring too fast. It's best to develop an incremental rollout plan that incorporates those security touchpoints that balance the need to be effective with ease of implementation. Code analysis and security testing are often effective first steps, as the barrier to entry is small and good automated tools are available. Such tools include Fortify 360 (www.fortify.com), AppScan (www.ibm.com), Ounce 6 (www.ouncelabs.com), HP WebInspect (www.hp.com), and Coverity 5 (www.coverity.com).

Time and schedule pressures. Expecting existing software projects to address security issues prior to their next software release is typically unrealistic. Existing budgets and schedules were defined before security was made a consideration, and most development teams won't have the bandwidth or time to incorporate security into their development process. If an existing project's next release is deemed the most appropriate place to begin adopting application security practices, consider augmenting project teams with software savvy security

 Published by the IEEE Computer Society 1520-9202/10/$26.00 © 2010 IEEE

engineers who can perform the appropriate activities alongside software development.

Approach #2: Training Software Teams

There's no doubt that a successful application security program must include appropriate education and training. This includes training not only software developers and testers but also business analysts, project managers, and business sponsors—anyone involved in project and product planning. The best training incorporates hands-on activities, because most people learn best by doing. When appropriate, you should augment classroom training with mentoring during software projects. Note the following pitfalls when training software teams on application security.

"Not my problem" mentality. While many software engineers will agree that software design and coding activities often introduce vulnerabilities, they don't believe they themselves make such mistakes. Training is often wasted on individuals who believe that what they're hearing doesn't apply to them. It's best to incorporate into each training course specific examples of vulnerabilities from the organization's own code. Instead of using canned examples and case studies, try repurposing the results of application security assessments to make the training more applicable. It's difficult to argue that application security isn't a problem when actual assessment results show otherwise.

Hiring the wrong trainers. Nothing turns software engineers off faster than listening to someone talk about software development who isn't as technical as they are. Many organizations make the mistake of trying to use security personnel or professional trainers to teach application security to software engineers. This often results in half the class leaving after the first break. It's more important that application security trainers have strong software skills than that they be professional trainers or security gurus. Regardless, trainers must be able to deliver training in an engaging manner and know their material.

Approach #3: Deploying Security Tools

Tools are great for speeding up processes or providing knowledge or content that enhances a person's capabilities. They're also great at performing monotonous tasks repeatedly, saving your staff time and energy better spent elsewhere.

Security teams and management often procure application security tools (such as secure code analysis or Web application security testing tools) and expect software engineers to integrate and use such tools as part of their daily activities. Besides time and schedule constraints with introducing new technologies into existing projects, there are other pitfalls when procuring security tools.

A fool with a tool is still a fool. The value from a software security tool comes from its use within a defined process that's understood and followed. Make sure when wielding tools that users have been appropriately trained and are following a process that will result in business value. Too often tools are thrust upon software engineers without adequate training. Remember, automating a poor process just gives you poor results faster.

Choosing the wrong tool for the job. Security organizations sometimes make the mistake of evaluating and selecting tools that don't work in existing development or production environments. Sometimes nuances in your software (the programming language, code libraries, and deployed environment) or IT infrastructure (firewalls, security policies, and data encryption) will rule out some tools. Before you commit to purchasing and deploying a tool, make sure you first evaluate it in your development environment, on your code base, and within your operational environments.

Our Approach

To address these challenges, we at Coveros have developed a simple approach for introducing application security into ongoing software development projects.

Instead of asking software development organizations to modify their processes, wield tools they're unfamiliar with, or attend training they don't yet concur is necessary, our approach adds application security analysis into the software development process as a side effect of existing development best practices. We've found that by doing so, organizations are more likely to become aware of application security issues and begin addressing them as part of their existing software process. Once this occurs, it's much easier to move the organization toward a secure software process, institute appropriate training, and deploy security tools for use going forward.

Our approach leverages a software development best practice called *Continuous Integration* as the entry point for application security into software development.

Continuous Integration

CI is the software development practice of frequently integrating

software during the development process.[2] It's an outgrowth of the *nightly build* concept, which software development teams have been doing for decades to assure that new or modified code is automatically compiled and tested on a nightly basis. CI extends the idea of night builds to let development teams automatically compile and test their application at other critical times during software development. Automated CI is often performed during

- *Code check-ins*—code checked into a source code control system can be automatically integrated and unit tested to assure its quality. CI done during code check-in typically doesn't test the application's entire feature set but quickly confirms that code enhancements compile and pass a set of unit tests.
- *Nightly builds*—each night, software is automatically compiled and a full battery of regression tests are run to ensure the entire code base integrates and operates properly. Nightly builds also often automatically execute code analysis to ensure quality and compliance.
- *Weekly builds*—for tests that take too long to execute on a nightly basis, weekly builds are often established to compile and test software more fully.

To manage an automated CI process, CI servers have emerged. CI servers let development teams define when CI activities are performed and the amount of testing and code analysis that will be done. You can configure CI servers to not only automatically compile and test software applications but also to populate project and quality dashboards with CI results and also notify the appropriate individuals when the CI process fails.

Integrating Application Security into CI

Because many software organizations already use CI to automatically perform code analysis and testing as part of their software process, CI is a natural integration point for introducing application security analysis into software development. You can easily integrate secure code analysis and application security testing into existing code analysis and regression testing frameworks to make application security vulnerabilities visible during development.

You can integrate secure code analysis tools into your CI process easily by modifying your build scripts to perform secure code analysis during software compilation. All of the commercial and open source code analysis tools have instructions on how to do this with little effort. Since secure code analysis can take a significant amount of time to run, I recommend performing this analysis only on nightly and weekly builds. If you use a secure code analysis product that lets you control the fidelity of the analysis, you might be able to perform quick and simple scans during code check-ins as well, resulting in more frequent feedback on software vulnerabilities.

The best approach for integrating security testing tools into CI depends on how the security testing tool you select interacts with the application it's testing. Web application security testing tools often proactively crawl through an application looking for vulnerabilities, while other products analyze security while functional tests are executing. You can set up Web application security tools to spawn a process and crawl through an application during any testing phase. Tools that work alongside functional testing tools are often configured as a proxy that sits between an automated testing tool and the application being tested, looking for vulnerabilities as existing tests run.

Results from secure code analysis and security testing can be displayed within any standard project or quality management dashboard alongside the results of your traditional analysis. By integrating security results into traditional code quality and functional testing reports, application security becomes just another aspect of software quality. In addition, this integration makes it difficult for the software development organization reviewing and responding to defects identified during CI to ignore application security issues, since they appear as just another defect to correct before release.

SecureCI

To ease the process of incorporating application security analysis into CI and its associated software development process, Coveros developed SecureCI—an open source CI package that includes secure code analysis and application security testing. We integrated into one downloadable package security tools along with best-of-breed open source tools for source code control, control of the CI process, build management, automated testing, code analysis, and project dash-boarding. Besides integrating secure code analysis and application security testing into SecureCI, we also enhanced a quality management dashboard to display the results of security analysis when run as part of CI.

Figure 1 shows the quality dashboard that's populated when using SecureCI. The dashboard generates standard measures of code quality and compliance along with application security testing results (see the "ratproxy issues" box) while performing CI

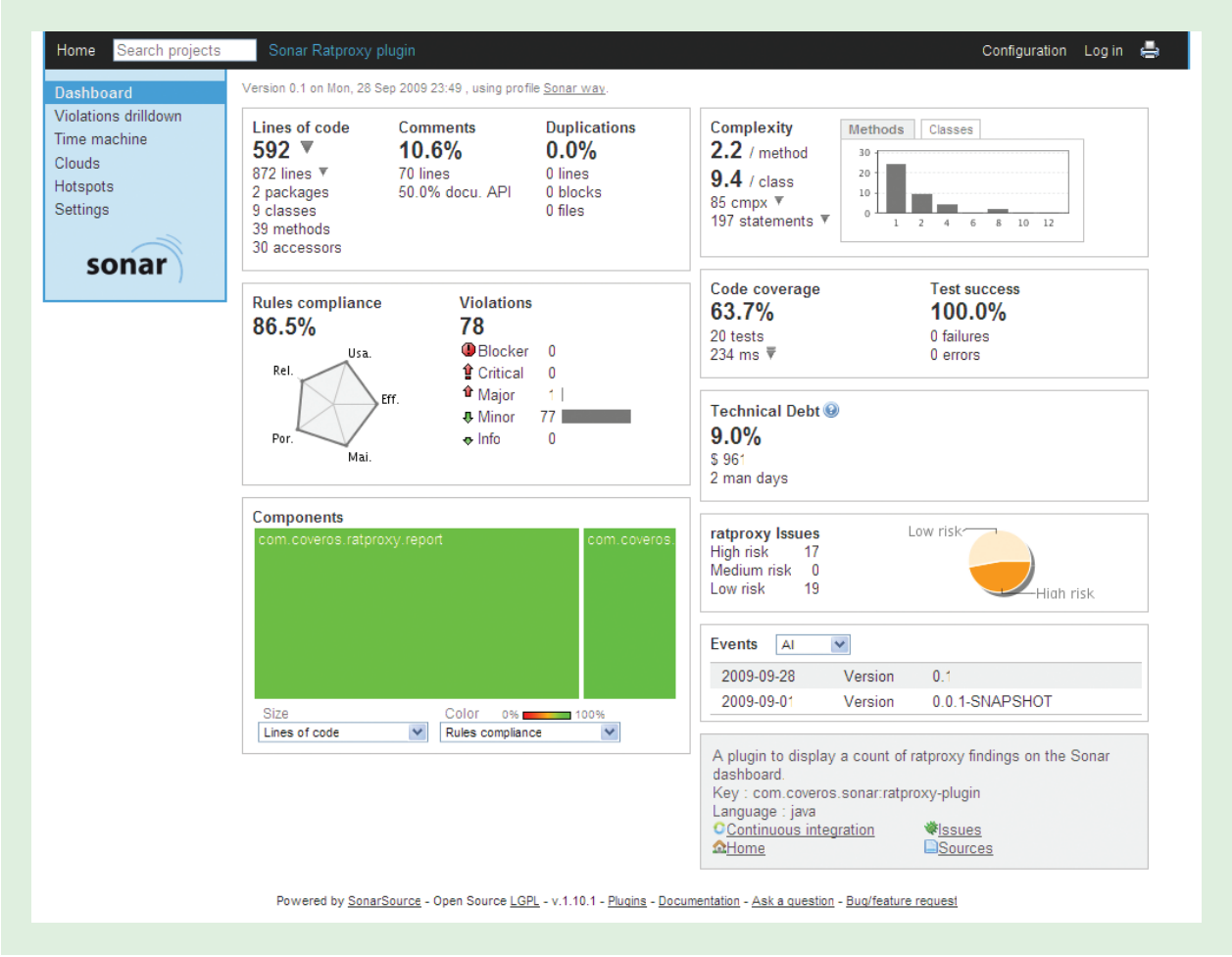

Figure 1. The Sonar Quality Dashboard for SecureCI. It displays integrated software vulnerability information.

activities on an application. You can view results for any individual build, or you can graph them over time to review quality and security trends during the development process.

Open source tools that we integrated into SecureCI include

- Subversion—for source code control,
- Hudson—a continuous integration server,
- Sonar—a quality management dashboard,
- Trac—for tracking defects,
- Maven and Ant—for build management,
- Ratproxy—for application security testing,
- Junit and Selenium—for unit and functional testing,
- PMD and Findbugs—for static code analysis (for both quality and security), and
- Cobertura—for code coverage.

Note that SecureCI currently supports only Java environments. A version for .NET is in the planning stage. (You can download SecureCI at www.coveros.com.)

Establishing application security initiatives within software development organizations is a critical challenge that must be addressed to build secure software applications. While process improvement, training, and security tools all play a part in any application security initiative, organizations often need a simpler starting point. By leveraging existing CI practices, you can seamlessly integrate secure code analysis and security testing into the software development process, thereby providing a vehicle for making development teams aware of the application security vulnerabilities they introduce. We have found that this awareness often opens the door for better integrating secure development processes, tools, and training into software development organizations.

References

1. G. McGraw, *Software Security: Building Security In*, Addison-Wesley Professional, 2006.
2. P. Duvall, S. Matyas, and A. Glover, *Continuous Integration: Improving Software Quality and Reducing Risk*, Addison-Wesley Professional, 2007.

Jeffery Payne *is CEO of Coveros, a software firm that specializes in secure software development. His research interests include code analysis, continuous integration, automated testing, agile development methods, and secure software development. Payne received an MS in computer science from The College of William and Mary. Contact him at jeff.payne@coveros.com.*