Agitar™ TECHNOLOGIES

Early and Often: Avoiding Security Flaws with Continuous Integration with High Code Coverage

Security vulnerabilities are caused by flaws in code that are exploitable and are not caught before software is released. There are tools available to try and find such vulnerabilities after they have been coded, but these tools are often used after software changes are migrated to the later integration/testing phases of development. Software flaws would be much easier to find early, and less likely to occur at all, if the software is being properly analyzed and tested in a continuous integration environment with tests providing a high level of code coverage. Such a process complements and enhances the value of static analysis tools that scan code for known security flaws.

A continuous integration process with high code coverage will:

- Help reduce security flaws from being added in unexpected ways, because negative impacts of code enhancements/changes are more likely to be found

- Help keep security flaw remediation from breaking other code, because negative impacts of code fixes to surrounding code are more likely to be found

The value of such a continuous integration process can be further enhanced if the process includes feedback from static analysis, including code rules and complexity metrics.

AgitarOne provides these capabilities for Java components of an application. **AgitarOne JUnit Generator** automates the generation of JUnit tests to a very high level of unit test code coverage. When executed after code changes, the generated tests indicate potential negative impacts of such changes. **AgitarOne Agitator** identifies code behavior and potential problems, and is particularly useful during new development and major enhancements to help the developer determine whether actual code behavior is as intended. Both AgitarOne components integrate easily with continuous integration environments, including the popular Cruise Control and Hudson solutions. Both AgitarOne products also come with Code Rules and Dashboard reporting capabilities to further enhance the continuous integration process.

In this paper, we will discuss the advantages of a continuous integration process with a high level of code coverage to facilitate the development of safe and secure code. We will consider:

- Why CONTINUOUS Integration of Each Code Change?

- Why HIGH Code Coverage During Unit Testing?

- Why Behavioral Analysis of NEW CODE As You Write It?

- Why Static Analysis (Code Rules & Metrics) WITH Continuous Integration?

- AgitarOne & Continuous Integration, for Security Flaw Avoidance

Why CONTINUOUS Integration of Each Code Change?

We all have great aspirations with our projects that *this* time, we are going to do it right. We will have high code coverage and no software will be released with failing tests or open bugs. In reality, the deadline often rules the release. Manually running tests is doomed to failure because there is frequently no enforced requirement that they be run, and when they are finally run there are so many failures that it is hard to track down the causes and bring the tests back to a passing state. If you have a build environment that runs all of your tests every time you check in code however, it is hard to ignore when tests fail. This causes quick turnaround time on failure resolution. Continuous feedback from tests also raises awareness of responsibility with developers. You cannot be confident in the security of your code if you don't actually know what it is doing, so testing is incomplete without continuous integration.

AgitarOne provides a continuous integration environment pre-packaged and provides a wizard to easily create builds that continuously run all of your tests … and continuously run other tool analysis features if you wish. Setting up your continuous integration environment is as easy as installing the AgitarOne Server and an Eclipse Client on the server machine, then running the Team Dashboard Wizard to configure a build (including build triggers, version control, email targets, combined projects and trend chart range). **In a short time you will have a fully configured continuous integration server feeding testing metrics to your entire team each time new code is checked in**.

Why HIGH Code Coverage During Unit Testing?

Lack of thorough code coverage during unit testing, combined with highly complex code, means that you will more likely fail to identify a new flaw in the development phase, and possibly miss the defect in the QA phase as well. This translates to high cost and a large time investment each time this happens. Lost time means less time for testing and less time for writing code at a high quality level. Low quality code means higher security risk. Having a wide array of tests that test positive code paths as well as exception conditions raises confidence and frees up time for your developers, and therefore allows closer scrutiny of the code being written.

AgitarOne provides the highest code coverage of any tool out of the box (over 80%!) and all of our generated tests are returned in a passing state. Our leading competitors only generate about half that, and many of their tests will be returned in a failing state, requiring manual manipulation to make them pass. This set of characterization tests gives you a safety net that catches regressions immediately when code is changed, lowering your risk of unintended behavior.

Automatically generated unit tests from AgitarOne are more thorough than hand-written tests alone. We recommend that you use both hand-written and tool-generated tests. AgitarOne combines coverage from both types of tests within continuous integration and the metrics dashboard.

Why Behavorial Analysis of NEW CODE As You Write It?

When writing new code, it is very easy to code in behavior that is different than what you intended. **Unintentional behavior is a huge risk and the leading cause of security vulnerabilities**. AgitarOne is the only tool available that provides a visual interface to write tests for your code as you write it, and allows you to understand what the test is actually doing. AgitarOne Agitator exposes your code's possible errant behavior such as unintended exceptions, unreachable code, common coding errors which are commonly introduced by copying and pasting, 'for' loop iteration problems, 'and vs. or', off by one, etc. Catching these problems early ensures that your code behaves as intended. With AgitarOne Agitator, creating tests based off of that behavior is as easy as clicking a check box.

Why Static Analysis (Code Rules & Metrics) WITH Continuous Integration?

AgitarOne includes code rules that use static code analysis. Enforcing code rules gives you the power to standardize your coding practices. The more standardized your code is, the easier it is for everyone to comprehend, which means more time for analyzing and less time deciphering. In order to find possible bugs and security problems and to create an enforceable expectation of code quality for your entire organization, no project should be without code rules.

AgitarOne is able to identify your high risk classes using our risk metric (low coverage and high complexity mean high risk), so you know specifically which classes need attention in testing and/or simplification.

AgitarOne code rules also identify common problematic coding mistakes that are either non-standard or likely to cause instability in the future. Examples include using 'equals ()' when comparing objects and not naming non-constructor methods the same name as the class.

In addition, the AgitarOne code rules component includes an API for you to extend the rule checks. This powerful feature enables you to create your own custom checks to detect coding structures and syntax usage that may introduce security vulnerabilities. Identifying and remediating such coding practices early on is an important factor in secure software development.

AgitarOne & Continuous Integration, for Security Flaw Avoidance

AgitarOne complements the static analysis tools that scan code for known security flaws. As with software quality, software security is best achieved by combining multiple techniques for defense. One should start by delivering software classes and methods that are validated to be free of flaws that could be exploited to allow security problems.

Summary

As noted earlier, both AgitarOne JUnit Generator and AgitarOne Agitator include built-in support for continuous integration and testing.

Combining JUnit tests created by AgitarOne JUnit Generator and assertions created using AgitarOne Agitator gives you a far more thorough set of regression tests than would be possible with hand-written tests alone. It is important to run these tests frequently to reduce the time that passes between a change and a regression caused by that change. Continuous integration using AgitarOne rebuilds the modules, runs the regression suite, and reports back on the status. This immediate feedback allows a developer to identify whether any change has introduced a regression, so it can be fixed immediately.

Using AgitarOne to catch unintended behavior early saves time and money, and helps ensure that the code you are writing is correct and safe. If you are not confident in your code and are concerned that a critical security breach could happen, AgitarOne can help your development team code better and code safer.