Building Security In Using Continuous Integration

Thomas Stiehm and Gene Gotimer
Coveros

Building security into software is harder than it should be. This article explores a way to align application security practices with other software development best practices in order to make building security in easier to manage and more cost effective. In particular, this article looks at combining continuous integration (CI) with security testing and secure static code analysis.

Application development and security practices are often at odds. Application development is concerned with creating software quickly with the most features possible in the minimum amount of time. Application security is concerned with finding and removing security vulnerabilities and releasing software when critical security risks have been mitigated.

Many project stakeholders see application security practices as an increase in scope that adversely impacts the software delivery schedule. In order to build secure applications, it is important to align application development and security practices. Our analysis has found that one of the best ways to do that is to integrate secure code analysis and security testing into CI. CI is a software development practice where members of a team integrate their work frequently, verifying each integration by an automated build/test process to detect integration errors as quickly as possible [1]. Using CI, the time and effort to build security into the development process can be minimized, making teams more likely to include security practices in their software development process and thereby reducing the risk of a successful attack.

Immediate Notification

CI ensures that ongoing changes to the source code do not break the intent or design of the software. If a change does break the software, that break is identified immediately and can be fixed with a minimal cost and impact to the project's schedule.

CI started with the notion that each CI cycle should make a *clean build* from an up-to-date checkout from the source code repository, and that a set of unit tests should be run against the clean build as a regression against the changes in the code base. If the build and unit tests pass, then the recent checked-in changes did not break the software. If the build or unit tests fail, then the changes broke the software, and the CI server immediately notifies the team that the software is broken.

Secure Development

CI is now the foundation for serving many crucial software development process tasks. Originally focused on compiling and unit testing, CI practices have grown and evolved over time. They now include expanded practices, such as functional testing and code analysis to evaluate the health of a project. By integrating security testing and secure code analysis, CI can be further leveraged to include secure development practices while minimizing the amount of extra effort required to get the benefits of secure development. Since it is tied to CI, security testing and secure code review begins when a project begins and runs continuously throughout project development. With CI, security vulnerabilities testing becomes part of the regression test bed, executed automatically with each successive build on the CI platform.

Changing Testing Economics

Technology advances have changed the economics of testing, allowing more aggressive approaches to testing than historically possible. Using CI for build, test, and analysis automation has increased the depth and breadth of tests while also making them faster and less expensive. By making it cheap and easy to perform tests, teams are encouraged to test more and test sooner in the development cycle, reducing the cost of fixing bugs. It has also made it easier for managers and non-technical stakeholders to understand project progress and health.

For many projects, testing has gone from a process that slowed deployment down to one that provides true quality assurance, in turn helping stakeholders have more confidence in their projects. By reducing the cost of many quality control aspects of application development, teams have been able to use those controls more often and more effectively and have accelerated development while improving quality. This same change can be applied to security practices integrated into CI. The reduced cost of gathering security vulnerability data will encourage teams to collect the data more often and sooner in their development cycle, reducing the cost to fix issues such as cross-site scripting and SQL injection.

Building Security Testing Into CI

In order to integrate static code analysis and security testing into CI, a few key pieces of software are needed. Organizations should select a specific application for each of the software categories (as shown in Table 1), bearing in mind that some products might fit into more than one category. For example, Microsoft's Team Foundation Server (TFS) [2], fits into the CI Server, Source Code Repository, and Issue Tracking categories.

Choosing the Right Tools

It is possible to build a completely open source CI process that includes static code analysis and security testing. SecureCI is one example of a CI product that is made of all open source tools and can be downloaded and used free of charge [3]. There are also many good commercial products that, in some cases, provide more value than their open source alternatives, such as the Web crawling capability of AppScan and WebInspect.

There are many tools to choose from, ranging from unsupported to company-supported open source software to commercial software that is developed and maintained completely by the company that created it. In other words, an organization can purchase support contracts to commercial software—products that are then developed and maintained completely by the company that created them. There are even some commercial software products that are free to use but come with some restrictions, such as a limited number of users. There is no correct or incorrect tool as long as the acquired software works together and can meet organizational objectives.

Tool Integration

In general, the CI server will be the integration hub and orchestrator for a CI process. CI servers come with integration application programming interfaces (APIs) for many of the tools an organization puts in its toolset so that working with those tools will be easy. If there isn't an API for one of the selected tools, CI servers can be extended using scripting languages and compiled code plug-ins. This flexibility means that an organization can add any tools that can be scripted or programmatically controlled into their CI process, making it possible to add security practices to an existing CI process without having to reinvent that process. For example, with Hudson, in order to use PMD [4] or FindBugs [5] for static code analysis, all that is needed is creation of a build job that uses an Ant [6] build to run the tool. You then point the tool's Hudson [7] plug-in to the XML reports created during the Ant build. The plug-in picks up the reports and parses them for display using HTML and generated graphics.

The CI server market is full of both open source and commercial products. Many of the open source CI servers have commercial support contracts available. Because of the mature CI server market, there is no advantage in using either open source or commercial products. The exception to this statement is when an organization is considering development ecosystem suites like Microsoft's TFS and Visual Studio suite.

Integrating security testing tools requires a little more work. The application under test needs to be deployed and running because security testing tools work by interacting with the application, analyzing the requests and responses from the point of view of a Web browser. This means that the CI server will have to deploy to an application server, start it, and then kick off the security testing tool. While it seems like a lot of work, there are a number of good examples on the Internet to help get started. The commercial security testing tools can be configured to just launch and crawl an application.

Creating Multiple Complementary Builds to Support Specific Needs

One of the first things organizations will notice about using static code analysis and automated security testing tools is that build times—the time it takes to go through a CI process—will increase significantly. Because of this increase, it is common for projects to use multiple CI jobs. The different jobs are set to run on different intervals and for different reasons. For instance, most teams have a quick job that runs within 10 minutes of a check-in and produces a result within 10 minutes. This quick job consists of a clean build and only executes unit tests. This quick build tells the developers that the new code works and that all of the unit tests pass (i.e., the code hasn't introduced defects into previously working code). Many teams will also use longer running tests (a couple of hours), compiling the project and running the unit tests, while also executing other kinds of tests (e.g., database tests and automated functional tests). These tests take longer to complete and have a longer feedback cycle. By executing multiple jobs, an organization can provide the team with feedback as quickly as possible for a given type of feedback. Finally, many projects have jobs running from once a day to once a week that perform static analysis or security testing. This allows the processes to run at their own pace without slowing down other test processes.

The selection of a source code repository is generally based on what already exists in an organization's environment. From the overall goal of building security into applications, the choice of source repository makes little difference—it just needs to work with the CI server.

Utilizing Both Commercial and Open Source Tools

There is a healthy marketplace for issue tracking applications having both open source and commercial products.

Commercial issue tracking software has an advantage over open source in terms of the reporting and integration options. Commercial applications tend to have a better and more customizable reporting system and tend to integrate more with (usually commercial) software that might be used in an overall development process. That said, many projects and organizations don't need or use the extra capability of commercial issue tracking software. For small project teams or small companies, open source issue tracking software works just fine. For large enterprises with multiple related development projects, a commercial issue tracking application may offer needed features that can justify the cost of acquiring the software.

Open source unit testing tools are usually frameworks that provide a core unit testing capability. These tools require a developer to write and maintain unit tests. If the developer follows the conventions of the framework, running unit tests is very simple and easy and many CI servers can read and report on the results. The commercial products in the space build on top of the open source tools by adding the capability to generate unit tests. The commercial tools will scan the source code and determine how to exercise the code paths (in the source code) in order to get 100 percent test coverage and possibly add negative unit tests.

We have worked on projects that have used both developer-written unit tests and tool-generated unit tests. When dealing with security features, it is important to write comprehensive unit tests that exercise both the positive and negative paths

Table 1: *Tools for Different Software Categories*

Software Category	Description	Open Source Tools	Commercial Tools
CI Server	Server software that monitors the source code repository and runs the build when changes to the repository are detected.	Hudson, CruiseControl	TeamCity, Bamboo, TFS
Source Code Repository	Software that keeps source code, maintaining versions of the source files and file groups (i.e., labels and tags).	Concurrent Versions Systems, Subversion	Polytron Version Control System, Clearview, TFS
Issue Tracking	Software that is used to manage software issues and report their status.	Trac and Sonar, Bugzilla	Quality Center, TFS
Unit Testing	Tools or frameworks used by developers to test at a source code level.	JUnit, NUnit	JTest
Functional Testing	Tools, frameworks, or applications used to test the functionality of software.	Selenium, Watir	Quick Test Professional, SilkTest
Security Testing	Tools, frameworks, or applications used to test the security aspects of software (i.e., penetration testing tools).	RatProxy, WebScarab	AppScan, WebInspect
Static Code Analysis	Tools, frameworks, or applications used to inspect either source code or compiled files for known issues.	FindBugs, PMD, CheckStyle	Fortify Source Code Analyzer

Software Defense Application

Application security is a priority for DoD applications, given the always on, global nature of the DoD mission. At the same time, it is important to make application security work within the development practices in common use among DoD development teams. By integrating application security practices with CI, we can address both the security needs of the applications as well as the efficiency and cost-effective requirements of the development teams.

through the code. Many security defects come from security features that don't have a proper *fail-safe* method for when there are program or data processing errors. Negative testing—or the testing for failure paths in the source code—addresses the correctness of failure states and can show that a specific security feature can fail-safely when something unexpected happens. If there is a large legacy code base that doesn't have unit tests, using a tool to generate unit tests is a good way to add tests quickly. Keep in mind, however, that the project team still needs to review all of those tests, change some, remove others, and write new tests covering situations that the tools couldn't.

Security Testing Tools

Security testing is an excellent way to discover many of the security vulnerabilities in an application. These tools are run against an application, usually in a testing environment, in its production configuration. One goal of this testing is to determine if the application has defects, making it vulnerable to outside attack while in production; another is to see if the application will fail safely when attacked. Failing safe has different meanings to different applications, with the basic guideline that an application should not give users unauthorized information or allow them to take unauthorized actions.

In order to automate many open source security-testing tools, they will need to be used in conjunction with functional test tools. For example Ratproxy [8], a popular open source security-testing tool, can't (unlike popular commercial security testing tools) crawl a Web application. This means that people using Ratproxy need to use another tool to crawl the Web application while Ratproxy is running. Another alternative is to have the testers use Ratproxy while they are conducting manual functional tests.

Another area where commercial security testing tools have an advantage is in reporting. Commercial security testing tools, like AppScan [9], have a customizable reporting capability that gives users numerous ways to report their findings (in order to conform to organizational standards or highlight different aspects of the findings for different audiences). The reporting capability also does a good job of explaining the findings—that is, how they can be exploited and remediated. In contrast, Ratproxy has only one report format that provides little detailed information about an issue beyond its name and where it was found. An analyst using Ratproxy has to figure out what the issue means and how to remediate it. Finding out this information isn't hard, but it can be time-consuming.

Static Code Analysis Tools

Static code analysis tools examine the code base for many problems including security and code style issues, potential code defects, and race conditions. These tools are more akin to automated code review than to unit and integration tests. The difference between commercial and static code analysis tools mirror the differences found in security testing tools: Commercial tools have better reporting capabilities and provide more information on the nature of the findings and suggested remediation. Another area where some commercial tools have added value is in the depth of analysis. Fortify Software's Source Code Analyzer [10] creates an entire model of the application under analysis, examining the data and control flow of the software to determine if there is a larger-context problem. Most open source static code analysis tools only look for problems in a local context (i.e., the immediate line of code, code block, or file).

Static code analysis can either be performed on the source code of an application or on the compiled binaries. For example, FindBugs runs against compiled Java class files. It can find bad practices, null pointer dereferences, static use of non-thread safe code, and security issues. PMD runs against Java source files. It can find dead code, performance issues, style issues, and potentially dangerous code practices. While there is overlap with FindBugs, using both is not considered redundant.

Data Analysis and Evaluation

Once the vulnerability data has been collected from both security testing and code analysis, it has to be analyzed and evaluated. The goal of this step is to decide if the findings are truly a problem and what to do about it. This step includes determining the priority and severity of the security issues and putting them into an issue tracking system. In order for application security practices to positively affect the project, issues uncovered in testing and analysis need to be tracked and fixed. By integrating security practices into CI, security issues are discovered and dealt with more quickly, in turn preventing many security risks from entering production and minimizing the possibility of exploitation.

Building application security practices into a project can be done with minimum impact to a project's budget, schedule, and resources. Integrating security testing and secure code analysis into CI is the first step to building security into software. These practices help build security awareness by showing developers how and why their code is vulnerable. They give testers the tools needed to find many security vulnerabilities, and project managers a way to demonstrate the results of security practices. While it would be instructive to provide quantitative analysis of the benefits of integration CI and security, the practice is still new enough that no major studies have been conducted.

Application security practices are hard for many teams to adopt because they don't have the time, budget, or resources needed. CI can help change the economics of security testing and analysis by giving project teams tools that can be deployed in their environment. These solutions can enable a team to quickly go from not considering security to a solid initial step in finding and proactively fixing security vulnerabilities.◆

References

1. Duvall, Paul, and Steve Matyas. *Continuous Integration: Improving Software Quality and Reducing Risk*. New York: Addison-Wesley, 2007.
2. Microsoft. "Team Foundation Server Home." *Team Foundation Server Home*. 2009 <http://msdn.microsoft.com/en-us/teamsystem/dd408382.aspx>.
3. Coveros. *Coveros: Research & Insights – Free Secure CI Download*. Fairfax, VA: Coveros, 2009.
4. InfoEther. "PMD" *SourceForge.net*. 2009 <http://pmd.sourceforge.net>.
5. The University of Maryland. "FindBugs – Find Bugs in Java Programs." *SourceForge.net*. 21 Aug. 2009 <http://findbugs.sourceforge.net>.
6. The Apache Ant Project. "Apache Ant – Welcome." *Apache Ant*. 15 Oct. 2009

About the Authors

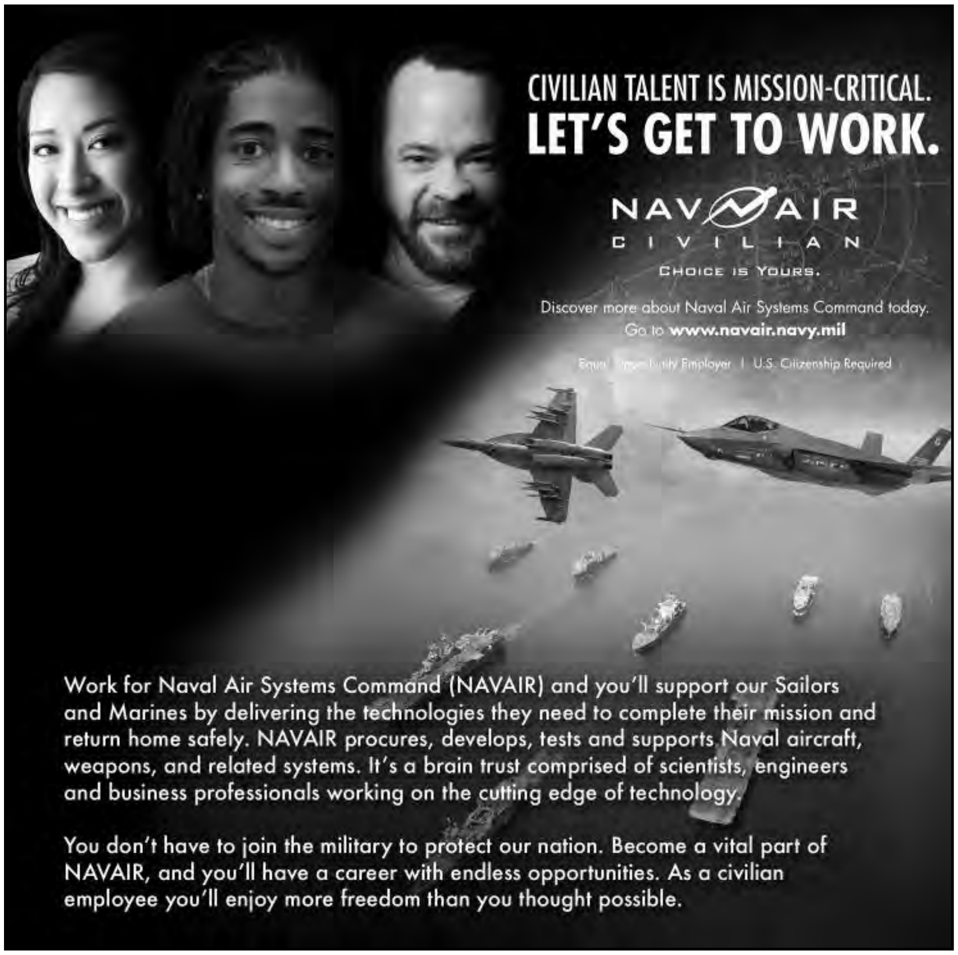
Thomas Stiehm has been developing applications and managing the software development teams for 16 years. As Chief Technical Officer of Coveros, he is responsible for the oversight of all technical projects and integrating application security practices into software development projects.

Coveros
4000 Legato RD
STE 1100
Fairfax, VA 22033
Phone: (703) 599-6243
E-mail: tom.stiehm@coveros.com

Gene Gotimer has been building Web applications and working with security for the last 13 years. He specializes in CI and Agile Java development.

Coveros
4000 Legato RD
STE 1100
Fairfax, VA 22033
Phone: (703) 963-1620
E-mail: gene.gotimer@coveros.com

<http://ant.apache.org>.
7. Sun Microsystems. "Hudson CI". *Hudson: Extensible continuous integration server*. 2009 <http://hudson-ci.org>.
8. Google. "Ratproxy: passive web application security assessment tool." Ratproxy – Project Hosting on Google Code. 2009 <http://code.google.com/p/ratproxy>.
9. IBM. "Help ensure Web site security and compliance." *IBM Web Site Security and Compliance – Rational*. 2009 <http://www-01.ibm.com/software/rational/offerings/websecurity>.
10. Fortify Software, Inc. "Source Code Analyzer (SCA) in Development." 2009 <www.fortify.com/products/detect/in_development.jsp>.


CIVILIAN TALENT IS MISSION-CRITICAL.
LET'S GET TO WORK.
NAVAIR
CIVILIAN
CHOICE IS YOURS.
Discover more about Naval Air Systems Command today.
Go to www.navair.navy.mil
Equal Opportunity Employer | U.S. Citizenship Required

Work for Naval Air Systems Command (NAVAIR) and you'll support our Sailors and Marines by delivering the technologies they need to complete their mission and return home safely. NAVAIR procures, develops, tests and supports Naval aircraft, weapons, and related systems. It's a brain trust comprised of scientists, engineers and business professionals working on the cutting edge of technology.

You don't have to join the military to protect our nation. Become a vital part of NAVAIR, and you'll have a career with endless opportunities. As a civilian employee you'll enjoy more freedom than you thought possible.

CrossTalk
The Journal of Defense Software Engineering

Get Your Free Subscription

Fill out and send us this form.

517 SMXS/MXDEA
6022 Fir Ave
Bldg 1238
Hill AFB, UT 84056-5820
Fax: (801) 777-8069 DSN: 777-8069
Phone: (801) 775-5555 DSN: 775-5555

Or request online at www.stsc.hill.af.mil

Name:____________________

Rank/Grade:____________________

Position/Title:____________________

Organization:____________________

Address:____________________

____

Base/City:____________________

State:________Zip:__________

Phone:(____)________________

Electronic Copy Only? Yes No

E-mail:____________________

Check Box(es) To Request Back Issues:

May2008	❑	Lean Principles
Sept2008	❑	Application Security
Oct2008	❑	Fault-Tolerant Systems
Nov2008	❑	Interoperability
Dec2008	❑	Data and Data Mgmt.
Jan2009	❑	Eng. for Production
Feb2009	❑	SW and SYS Integration
Mar/Apr09	❑	Rein. Good Practices
May/June09	❑	Rapid & Reliable Dev.
July/Aug09	❑	Process Replication
Nov/Dec09	❑	21st Century Defense
Jan/Feb09	❑	CMMI: Process

To request back issues on topics not listed above, please contact <stsc.customerservice@hill.af.mil>.