How Now Software Security?

Gary McGraw, Ph.D.
When we started working on software security ten years ago, we couldn’t convince even our best customers of its importance, much less their network security people and developers. Things have come a long way since then. Today, everyone seems to agree that we need to do something to address the security problem at the software level, and a number of companies are even starting to do something about it. Cigital continues to lead the way.

It’s still early days for software security, though, and it’s a very good time to assess the state of the problem, how far we’ve come to address it, and how far we have to go. In general, we are very optimistic about the state the industry is in, especially considering the progress that leading software producers are making.

**Why Software Security?**

By almost any measure, it’s clear that the software security problem is growing. The Trinity of Trouble—connectedness, complexity, and extensibility—the three major factors that make the problem a continuously evolving challenge all do their part to keep things interesting. CERT reports that the serious software vulnerability problem continues to grow, with an increase since 2000 of over 500%. In 2005, there were 5,690 software security vulnerabilities that resulted in widespread security attacks (see Figure 1).

![Software Vulnerabilities](image)

*Figure 1: Vulnerability data from CERT.*
What’s critical to understand is that not all of these problems happen at the “application level” as some security vendors wish you to believe. Software security problems show up in router code, in operating system code, in cell phone code, and even in code for public key cryptosystems. Web-based applications suffer from serious software security problems too, but declaring that the problem applies only to Web-based software running over HTTP is naïve at best, and dangerous at worst.

If software plays an essential role in your business, you need to concern yourself with software security.

**Best Practices**

When *Building Secure Software* was published six years ago, much of its focus was involved with describing the problem. This was important philosophical ground breaking, and we have been pleased with the result, but the time has come to stop wringing our hands about the problem and start doing something about it.

The good news is that there is something we can do about the problem today. The new book *Software Security* (see Figure 2) describes in detail seven specific best practices that can be easily adopted by any software producing organization no matter what software development process it currently uses. We call these seven best practices the “software security touchpoints.” By focusing security attention on a set of common software artifacts like source code, architecture, and requirements documentation, we avoid religious warfare over which approach to building software is best and get down to the business of building better software.

If you build software today, you’re ready to adopt the touchpoints.

The seven touchpoints, presented in the order that they should be applied are:

1. Code review with a static analysis tool
2. Architectural risk analysis
3. Penetration testing
4. Security testing
5. Abuse case development
6. Security requirements
7. Software security operations

Adopting all seven touchpoints may be too much to swallow for some organizations, especially all in one fell swoop. Cigital’s recommendation is to start with the first two—code review and architectural risk analysis—and go from there. Whatever you do, though, don’t fall into the trap of doing either of the first two touchpoints without the other. It is important to pay attention to both kinds of software defects that lead to security problems—*bugs* at the implementation level and *flaws* at the architectural level.
If you find yourself only applying the third touchpoint by hiring “reformed hackers” to hack your software, you may do OK diagnosing the problem, but you’re unlikely to know what to do about it. It’s better to review your code and take a close look at your software architecture, then fix what you find.

One last thing about the touchpoints: notice that the touchpoints are not security features or mechanisms like cryptography, authentication, access control, and confidentiality. Instead, they are lightweight software analysis and construction activities that should be carried out during development. Software security is not security software.

**Tools**

There are two kinds of basic tools for software security that are widely available: security testing tools (which we refer to as badness-ometers) and source code analysis tools.

Security testing tools for software, such as the ones made by SPI-Dynamics and Watchguard, have a few unfortunate problems, but they have an important role to play in the evolution of software security. Though they only test Web-based application software, application security testing tools can help alert an organization of trouble in the software sector. That is, if canned black box tests find problems in your applications, you know you’re in very deep trouble indeed! Of course, if the same tests find no security problems at all, all that you know is that those particular tests didn’t find any problems...you do not know that you’re secure. That’s why we refer to such tools as badness-ometers (as opposed to security-ometers)—they can return results only in the range from “deep trouble” to “who knows?” See Figure 3.

All that said, we believe that everybody should stock up on badness-ometer tools, and use them on a daily basis. It turns out that it’s good to know if you’re in deep software security trouble. Such knowledge can help motivate an initiative to do better.

Source code analysis tools, such as the one produced by Fortify Software, are more important to making forward progress solving the software security problem than those tools that only diagnose the problem. Static analysis tools that help developers and software security analysts find and remove common software security bugs from their code are essential. If you’re not using a source code security scanner tool today, we believe you are taking an unacceptable business risk.

The first touchpoint in Software Security is based on properly adopting these tools in all development groups. Of course you can’t simply throw a bag of tools over the wall to the developers and expect the software security problem to disappear magically. These tools, however, have built in knowledge and analysis power often otherwise lacking in most software development shops, so they do have an important role to play.
Software Security Initiatives

Simply put, we can’t tool our way out of this problem. Instead, we must adapt the way we build software by integrating security touchpoints deeply into the software development lifecycle (SDLC).

Probably the best known large-scale software security initiative in a large enterprise is Microsoft’s Trustworthy Computing Initiative. That’s because Microsoft has put some serious muscle behind the program and is proud to trumpet the result. Michael Howard, Steve Lipner, and the others at the heart of the initiative deserve high praise for their work. The interesting thing is that many other large enterprises are diligently working on corporate-wide software security initiatives involving thousands of developers (though they’re not touting them in PR campaigns).

Cigital has been fortunate to be directly involved in five such programs in organizations ranging from large finance houses and investment banks to producers of consumer goods and hotel chains. It turns out that a number of large enterprises that you might not associate with software understand the kinds of business risks that software brings to bear, and they are working hard to manage them.

These large initiatives all have several things in common: a framework that is sponsored and supported at the executive level, a large-scale awareness and training program for developers, creation and maintenance of a portal resource that includes code samples and other guidelines for development, and the instantiation and adoption of the touchpoints inside development groups themselves.

Software Security Now

In our experience at Cigital, organizations go through several phases of maturity when they address software security. First-stage companies still need to get a handle on the software pile and its associated risks. In some cases, these companies have appointed a “lone wolf”—a person who has been made responsible for software security and is just getting started. This person quickly becomes overwhelmed. In the second stage of maturity, companies build an internal group that often ends up functioning as a fire department. The fire department carries out spot code reviews, puts out raging software security fires, and generally acts as a “go-to” group in the company. The fire department is extremely useful, but does more to react to the problem than to proactively address it. Third-stage companies execute against a framework for enterprise best practice adoption, moving software security into the SDLC and taking pressure off the fire department. We have customers in every stage and know what it takes to move from one level to the next.

No matter what size your organization is, from a handful of developers to tens of thousands spread over four continents, the time has come to spearhead a software security initiative. Computer security depends on it.
About Cigital

Cigital is a leading consulting firm specializing in software security and quality. Cigital enables clients to deliver reliable software faster—gaining insight and control over software processes, products, and policies. For repeatable delivery and deployment of secure and reliable software on time and under budget, come to Cigital. Software Confidence. Achieved.

Founded in 1992, Cigital is headquartered in Northern Virginia with offices in Boston and Los Angeles. For additional information about Cigital and its services, please contact Cigital at 800.824.0022 or at http://www.cigital.com.

About the Author


Acknowledgements

A version of this white paper appeared in the June 1, 2006 edition of SD Times, pp. 32-33.