freedom-to-tinker.com

Software Security: The Badness-ometer

Gary McGraw

4-5 minutes

Here is another excerpt from my new book, Software Security: Building Security In.

Application Security Tools: Good or Bad?

Application security testing products are being sold as a solution to the problem of insecure software. Unfortunately, these first-generation solutions are not all they are cracked up to be. They may help us diagnose, describe, and demonstrate the problem, but they do little to help us fix it.

Today's application security products treat software applications as "black boxes" that are prone to misbehave and must be probed and prodded to prevent security disaster. Unfortunately, this approach is too simple.

Software testing requires planning and should be based on software requirements and the architecture of the code under test. You can't "test quality in" by painstakingly finding and removing bugs once the code is done. The same goes for security; running a handful of canned tests that "simulate

malicious hackers" by sending malformed input streams to a program will not work. Real attackers don't simply "fuzz" a program with input to find problems. Attackers take software apart, determine how it works, and make it misbehave by doing what users are not supposed to do. The essence of the disconnect is that black box testing approaches, including application security testing tools, only scratch the surface of software in an outsideâ†'in fashion instead of digging into the guts of software and securing things from the inside.

Badness-ometers

That said, application security testing tools can tell you something about security—namely, that you're in very deep trouble. That is, if your software fails any of the canned tests, you have some serious security work to do. The tools can help uncover known issues. But if you pass all the tests with flying colors, you know nothing more than that you passed a handful of tests with flying colors.

Put in more basic terms, application security testing tools are "badness-ometers," as shown in the figure above. They provide a reading in a range from "deep trouble" to "who knows," but they do not provide a reading into the "security" range at all. Most vulnerabilities that exist in the architecture and the code are beyond the reach of simple canned tests, so passing all the tests is not that reassuring. (Of course, knowing you're in deep trouble can be helpful!)

The other major weakness with application security testing tools

is that they focus only on input to an application provided over port 80. Understanding and testing a complex program by relying only on the protocol it uses to communicate provides a shallow analysis. Though many attacks do arrive via HTTP, this is only one category of security problem. First of all, input arrives to modern applications in many forms other than HTTP: consider SSL, environment variables, outside libraries, distributed components that communicate using other protocols, and so on. Beyond program input, software security must consider architectural soundness, data security, access control, software environment, and any number of other aspects, all of which are dependent on the application itself. There is no set of prefab tests that will probe every possible application in a meaningful way.

The only good use for application security tools is testing commercial off-the-shelf software. Simple dynamic checks set a reasonably low bar to hold vendors to. If software that is delivered to you fails to pass simple tests, you can either reject it out of hand or take steps to monitor its behavior.

In the final analysis, application security testing tools do provide a modicum of value. Organizations that are just beginning to think through software security issues can use them as badness-ometers to help determine how much trouble they are in. Results can alert all the interested parties to the presence of the problem and motivate some mitigation activity. However, you won't get anything more than a rudimentary analysis with these tools. Fixing the problems they expose requires building better software to begin with—whether you created the software or not.