by Gary McGraw

Is Penetration Testing a Good Idea?

Penetration testing, like television, is best experienced in moderation. It should be approached with a healthy degree of skepticism and carried out under adult supervision. In a standard-issue penetration test, one or two problems are unearthed and then exploited for all they're worth. This isn't the path to a thorough or complete security exam, however. In fact, penetration testing shares much philosophically with the "penetrate and patch" approach to security, which has had a detrimental impact on solid security engineering. Finding a couple of holes and fixing them makes everybody feel good, but may not necessarily result in a system that's any more secure.

Modern penetration testing has been broadened to include application penetration testing. Application penetration tests have an outside-in focus on software defects such as buffer overflow or cross-site scripting vulnerabilities. Once an application is finished, its owners subject it to penetration testing as part of the final acceptance regimen. These days, security consultants typically perform assessments like this in a "time-boxed" manner, expending only a small and predefined allotment of time and resources on the effort. The problem with this is that it almost always represents a too-little, too-late attempt to tack security on at the end of the development cycle.

As I've expressed in previous columns, software security is an emergent property of the system, and attaining it involves applying a series of best practices throughout the software development lifecycle. A late-lifecycle penetration test uncovers problems at a point when both time and budget constraints severely reduce the options available for remediation. More often than not, fixing things at this stage is prohibitively expensive.

JUST SAY NO TO MITNICK

In the commercial world, most penetration testing is carried out by external consulting firms usually claiming to be made up of "reformed" black-hat hackers. The good news is that these ex-hackers certainly know what they're doing in terms of breaking into target systems. The bad news is that it's not exactly clear what it means to be reformed. Too often, the claim of being reformed is a self-applied label. Who's to say that these reformed hackers will report every interesting result that may lead to a compromise, especially when one or two holes usually seem to satisfy most customers?

Penetration testing requires uncompromised and uncompromisable ethics—not the sort demonstrated by petty criminals. I'm not a fan of employing criminals or treating malicious hackers like rock stars, even if there *is* some evidence of improved behavior. Criminals should be treated as criminals. Period.

PENETRATION TESTING DONE RIGHT

One real advantage of penetration testing is that it probes security posture in a real-world environment (as opposed to in a lab). Because of this, different levels of security defenses can be probed at once in a realistic setting. For example, although an application may include security weaknesses that are exploitable, it may be protected by a firewall in a given network configuration. A penetration test provides a holistic view of the situation that takes network defenses into account.

Who's to say that these reformed hackers will report every interesting result that may lead to a compromise?

Penetration tests are best devised with some knowledge of the target system and its architecture. Ideally, a complete risk analysis will be available to the analysts and can be used to target the most risk-prone areas of the system. This kind of risk-driven penetration testing is far superior to a more "random" approach. Real malicious hackers not only carry out a risk-driven approach, but they also use advanced analysis tools such as decompilers, disassemblers, debuggers, coverage instruments, and fault injection engines. White-hat penetration testers must do the same.

Gary McGraw is CTO of Cigital, a software quality management consultancy. He is co-author of *Exploiting Software* (Addison-Wesley, 2004), *Building Secure Software* (Addison-Wesley, 2001), and *Java Security* (Wiley, 1996). Reach him at gem@cigital.com.