CrossTalk would like to thank DHS for sponsoring this issue.

Success of mission and business functions should be the focus of verification and validation activities. Issues from sloppy manufacturing hygiene and insufficient test and diagnostics can enable exploitation and compromise of operational functionality. Enormous energies are put into assuring safety-critical functions address any source of taint, such as vulnerabilities, weaknesses, and malicious logic. With more functionality being delivered through cloud services the burden of security shifts to development with test being the last line of defense.

Industry has substantially invested in improving the quality of products and systems because of customer/user demand; contributing to a rise of automated software testing capabilities and test services. As the size and complexity of software and logic-bearing devices increase, test and diagnostic capabilities must mature to address the changing environment in which products and systems are deployed. Despite encouraging results with various quality improvement approaches, the software industry is still far from zero defects. Part of that is a cultural issue with developers often making risk decisions for which they are not held accountable, such as disabling compiler warnings and selecting unpatched components from libraries. Testing is further complicated because software-based systems often have additional features, interfaces, and functionality that use third party libraries, general purpose applications, and a multiplicity of features in system libraries and system calls. As witnessed by the myriad of patches needing to be addressed due to residual exploitable vulnerabilities, weaknesses, and malware, software-based systems are inherently susceptible to attack and manipulation. To address the difference between what is conceived and what is delivered, testers need to think about how software-based systems are actually integrated and deployed. If libraries are incorporated and deployed by a compiler, or configuration choices undermine design choices, or someone exposes a weakness, then testers need to factor in means for detecting these before deployment; not after an application or system compromise. Often, more comprehensive test programs are needed. This requires improved security functionality and more rigorous review, testing and inspection. Test coverage for agile continuous testing, automated API testing, metamorphic testing (including runtime checking), fuzz testing, and other techniques and methods need to be part of test organizations' process improvement list of strategic considerations. Multiple techniques and tools, including static and dynamic analysis, should be used for software assurance. Test-driven development is a programmer practices that has been employed by a growing number of software development teams. Despite the fact that testing often accounts for at least 30-40% of total project costs, only limited attention has been given to testing in various software process improvement models, including the Capability Maturity Model Integration (CMMI). As a result, the testing community has developed and used its own improvement models, such as the Test Maturity Model integration (TMMi) for test process improvement that is positioned as being complementary to the CMMI to more comprehensively address those issues important to test managers, test engineers and quality professionals.

Unfortunately, partially because of the lack of adequate due-diligence and due-care in development and integration test activities, vulnerabilities are proliferating rapidly; thus stretching mission capabilities and resources. As we seek to discover and mitigate the root causes of these vulnerabilities, sharing the knowledge we have of them help to mitigate their impact. In order to keep pace with growing threats we must facilitate the automated exchange of information. With that objective the Department of Homeland Security (DHS) sponsors 'free for use' standardized means for sharing information. These include the Common Weakness Enumeration (CWE) that provides standardized means for identifying and mitigating architectural, design and coding flaws introduced during development and detectable in testing, along with the Common Attack Pattern Enumerations and Classification (CAPEC) that enables developers, testers and defenders to discern attack constructs, build software resilient to them, and determine the sufficiency of test regimes focused on checking security concerns. These open specifications for interoperable security automation enable secure, machine-to-machine communication of actionable indicators within and between organizations that want to share this information. These have been developed collaboratively between Federal Government and industry partners working toward information sharing mechanisms and solutions to reduce the risk of tainted components. These standardized means for sharing information are already being used, and they contribute to efforts that enable more stakeholders to secure their part of cyberspace.

CrossTalk again thanks DHS Office of Cybersecurity and Communications for co-sponsoring this issue focused on test and diagnostics. Along with DoD, NIST, and GSA, DHS co-sponsors the Software & Supply Chain Assurance (SSCA) Forum in which Federal, academic, and industry stakeholders address risks and mitigation methods. SSCA Forums are free and open to the public, and resources are available on the SSCA Community Resources and Information Clearinghouse, with many applicable to test and diagnostics -- see <https://buildsecurityin.us-cert.gov/swa/pocket_guide_series.html> for "Software Security Testing" and "Key Practices for Mitigating the Most Egregious Exploitable Software Weaknesses."

Justin T. Hill
Publisher, CrossTalk