Standards and Testing for Software Security

Position Paper for the NIST Workshop on Standards and Guidelines to Enhance Software Supply Chain Security

Steven B. Lipner, Executive Director, SAFECode

The Executive Order on Improving the Cybersecurity of the Federal Government is a major step in the U.S. Government's recognition of the challenges posed by software security and initiation of measures to meet those challenges. This position paper is submitted on behalf of SAFECode, a nonprofit global industry forum dedicated to the creation of scalable and effective software security programs. The paper responds specifically to the NIST calls for input on secure development lifecycle standards and on requirements for testing source code (areas 2 and 4 of the workshop announcement).

Secure Development Lifecycle Standards

Secure software development processes date back to the early 2000s when several software vendors, including SAFECode members, began to integrate specific secure development requirements into their software development processes. At a high level, most secure development processes share common mandatory elements including:

- The use of threat modeling as a design analysis technique.
- The imposition of coding standards appropriate to the organization's software development language(s).
- The use of code analysis (static analysis) tools to detect code-level security errors.
- The use of dynamic analysis testing to detect security errors in executable code.
- Strict rules for the review, acceptance, and management of imported software components.

While the elements listed are common across secure development processes, the processes must be very specific to be effective. For example, an effective process would not simply require "run the xxxx static analysis tool," but rather "run the xxxx static analysis tool and fix all errors from the following list '4532 6029 6053 6057…'" Different development organizations will make their own choices of tools and must-fix errors based on their choice of programming language, their experience with security errors, and considerations of cost-effectiveness and integration. There is no single standard that is both sufficiently specific to be effective and sufficiently general to encompass all "critical software" as the Executive Order defines the term.

Thus, it will be necessary for NIST to define a high-level secure development lifecycle process standard that is general enough to apply across development organizations and technologies, and for development organizations to create their own specific process standards that adhere to the requirements of the high-level standard.

Fortunately, the NIST Secure Software Development Framework (SSDF) provides an excellent basis for the creation of organizations' process standards. The SSDF reflects industry experience creating secure development processes including input from SAFECode, BSA – The Software Alliance, and OWASP. It has

undergone public review and was released in April 2020. We recommend that the SSDF be adopted without modification as the Secure Development Lifecycle standard required by the Executive Order.[1]

Individual development organizations must create their own specific development processes based on their choices of programming languages, technology, and tools. Any such process can be mapped to the practices identified by the SSDF. Development organizations' correct implementations of their specific processes will be reflected by specific entries in their development workflow (bug tracking) systems. The SSDF includes a requirement that the development organization conduct a root cause analysis of any new vulnerability and update the secure development process as appropriate: the revision history of the development process provides proof of compliance with this very important requirement.

Requirements for Testing Source Code

The SSDF incorporates a practice for implementation of a supporting toolchain for assuring the security of software under development. In general, such a toolchain may include static analysis tools, ad hoc tools that search for specific errors, configuration scanning tools that search for vulnerable permissions, and dynamic testing tools (such as fuzzers) that search for errors in completed software components. The choices are numerous and, as discussed above, the specific tools chosen must depend on the programming language chosen for development and the operating system platform, interfaces, and potential threats facing the end product.

No single standard can specify which testing tools or techniques should be applied to every kind of critical software. Individual development organizations must make their own choices. Frequent reporting of vulnerabilities in released software will make it clear if a development organization has made insufficient or ineffective choices, and if the organization applies root cause analysis and continuous process improvement, as specified in the SSDF, over time it will improve its choices and use of tools and the security of its end product.

The possible use of penetration testing for software security is worth specific mention. Penetration testing can serve the purposes of (1) detecting prevalent errors that indicate that a development organization is using an ineffective secure development process or not actually implementing one at all and (2) detecting subtle and sophisticated security vulnerabilities.

The use of penetration testing like that of other tools, should be defined by the secure development process. It is important that penetration testing (and "bug bounty" programs) be used for one of these two purposes listed, and not as a substitute for the use of automated tools.

Summary

The SSDF provides an excellent starting point for NIST's implementation of its responsibilities under the Executive Order and for development organizations' efforts to create secure development processes. Development organizations should use automated tools appropriate to their technology and development language as a key component of the secure development processes that they create in response to the requirements of the Executive Order.

---

[1] Note that the SSDF includes a Practice "Protect All Forms of Code from Unauthorized Access and Tampering" which addresses software integrity and provenance rather than the "pure development" practices usually associated with a secure development lifecycle process.