2. Initial list of secure software development lifecycle standards, best practices, and other guidelines acceptable for the development of software for purchase by the federal government

Submitted by:
Jeff Williams, Veteran Application Security Industry Leader
jeff.williams@contrastsecurity.com
https://www.linkedin.com/in/planetlevel/
410-707-1487

We write to urge NIST to consider modern software development practices, such as Agile and DevOps when creating a list of secure software development standards for software to be purchased by the federal government.

We encourage NIST to consider the following ideas while coming up with this list.

1. **Focus on cost**. NIST must seek out real evidence about both cost and benefit of every activity, practice, and requirement into account. Many application security practices and requirements are duplicative, wasteful, and unnecessary. For many software security standards, the cost of fully implementing them can be many times the cost of developing the software. Even for many of the most common practices there is little or no proof that they deliver value. Actually, there is often considerable evidence that they do not deliver value. We recommend highly focused standards and practices that are highly cost-effective and specifically target eliminating the most serious risks to critical systems.

2. **Focus on outcomes**. NIST should not attempt to specify specific application security requirements or specific lifecycle activities. Standards that will work for all the different types of software development organizations and all the different types of software do not and will not exist. Instead, NIST should focus on the outcomes proving secure software and leave the implementation to software organizations. It does not matter what practices are used to create these outcomes are achieved and software development organizations are likely to innovate in this space if they are given the opportunity. The most important outcome is a strong assurance argument demonstrating that each threat has a strong defense strategy, that each strategy is implemented with standard defenses, that each defense is correct and effective, that defenses are protected and monitored in production, and that evidence is continuously gathered to support and communicate supporting this chain of trust.

3. **Require visibility**. NIST should require organizations to disclose information about the security of the software they produce. Without this disclosure, buyers, consumers, and users cannot make informed decisions about security. This is a classic market failure that disincentivizes cybersecurity in the software market. A software security label or data sheet should include information about the software itself (defenses, components, configuration), the tools and technologies used to build the software (IDE, pipeline, QA), security tools and processes (threat modeling, security architecture, AST, pentests, code review), and the people involved in building the software.

4. **Focus on direct measurement**. NIST should heavily prioritize direct measurement of application security. To determine whether software is secure, you can either directly measure the software itself or you can measure attributes the organization that built the software, such as their security processes, tools used, training programs, security culture, etc… Indirect measures have repeatedly been found to have little to no correlation with the security of the code that created. Despite the use of traditional security practices, the average number of serious vulnerabilities per application has remained stubbornly constant at about 35 over the past 20 years.

5. **Evolve existing standards**. NIST should create new software security standards designed for real world modern software development as existing standards are dated and ineffective. There are myriad software and application security standards available. We have been deeply involved with several of them, starting with the Rainbow Series, Common Criteria, and Systems Security Engineering CMM and, more recently, the OWASP Top Ten, Software Assurance Maturity Model, OWASP ASVS, and PCI Software Security Standard. While logical and well intentioned, there is vanishingly little evidence that demonstrates that existing standards have been effective in reducing vulnerabilities or generating assurance.

6. **Learn from DevOps**. NIST should follow the path blazed by DevOps pioneers to improve quality and predictability. Software projects were traditionally slow, late to market, full of bugs, and had massive cost overruns. DevOps has made dramatic improvements in all these areas by focusing on encouraging the flow of work into production by breaking work into small pieces, creating tight feedback loops, and fostering a culture of innovation and learning. Software security has not learned any of these lessons. Work is generally massive tasks that span all risks – like defining security requirements, building a security architecture, or testing application security. Feedback loops are slow and often driven by PDF reports. And security culture is often siloed, authority driven, and rife with conflict. While many are attempting "DevSecOps," many will fail as very few are fundamentally changing the nature of security work. NIST can take a leadership role and drive the entire software industry into a new era of security, velocity, and assurance.