Comcast Submission to NIST's Call for Papers on Consumer Software Labeling

Noopur Davis | Chief Information Security and Product Privacy Officer

Software consumer labels are intended to allow non-expert end-users to differentiate products based on security capabilities. Well-designed labels should avoid being based on static snapshots or just-in-time artifacts of security assessments, as attacker capabilities are always evolving, and new vulnerabilities are discovered daily. Labeling regimes must therefore be voluntary and flexible, in order to facilitate their ready adaptation and configuration to the broad variety of business models, software products, development processes, applications, risk profiles, and platforms in the marketplace today. A voluntary labeling scheme also must avoid homogenization of security and creation of systemic risk. While security capabilities can be defined for a given software product itself, any such description would not necessarily capture the product's operation on any third-party platform or in combination with any other software. Security proofs for composable systems, for example, are a known hard problem in research. Based on these considerations, **the *process* that builds and releases software is arguably more important than any point-in-time security attribute of the software itself**.

Developers recognize that this build/release process is centrally important to security and as a result have developed approaches such as the Secure Software Development Life Cycle, Secure Development Lifecycle, or DevSecOps (collectively, SDL). These processes may differ based on the organization, its products, and its risk profile. But they almost always address security in the entire **software development process** – architecture, design, coding, testing, deployment, operation, and maintenance. A focus on SDL, rather than a point in time artifact, would help address the modern software development process, in which software is continuously released (or updated), often multiple times a day, as part of the continuous integration/continuous deployment (CI/CD) pipeline in cloud-based environments. The SDL process also covers the different kinds of software itself, including Software-As-A-Service (SaaS), Apps, embedded components in operational technology (OT) and Internet of Things (IoT), in addition to traditional on-premises enterprise software. Furthermore, vendors can verify the internal artifacts generated from their SDL processes without potentially disclosing proprietary information.

An effective SDL approach creates a culture of security, wherein security is not just the responsibility of the security team but also that of every developer, tester, product owner, infrastructure owner, and project manager. It goes beyond a focus on security tools and practices (e.g., static code analysis) to building a community (e.g., a Security Guild), artisanship (e.g., secure coding), governance (i.e., policies, standards, and technical controls), and communication (i.e., awareness and messaging).

One impediment to a security culture is the added friction or the additional costs of building in security. **Thus, SDL processes should aim to reduce the impact on software development teams.** For example, such processes should integrate security tools as resources into CI/CD pipelines. They should integrate Software Composition Analysis tools to identity vulnerabilities in open-source components, and code analysis tools to find vulnerabilities in code being written. SDL processes may also employ a "policy-as-code" approach, where use of an Application

Programming Interface (API) or a library or micro-service by a development team automatically implements a policy or standard. Availability of reusable secure design patterns can also reduce the overhead for secure implementation. Finally, team agreements to find-and-fix high-priority code vulnerabilities, prior to merging code, may reduce the cost of patching later.

Even the best and most security-minded organization cannot produce perfect code. Thus, **a core piece of a comprehensive and effective SDL should address vulnerability lifecycle management**. This includes system hardening, patching, updating, vulnerability scanning, a responsible vulnerability disclosure or a bug bounty program for third party researchers, red teaming, and more. Vulnerabilities may also be used to continuously re-assess and improve the SDL program.

As NIST examines "formal and informal processes and practices used to secure the software development process," resultant recommendations should focus on and leverage the key properties of an SDL identified above. Software vendors who produce code using an SDL with these properties may wish to engage in market differentiation. While no established or eco-system-wide accepted mechanisms to do so exist, any proposals must address scalability and sustainability. For example, NIST's call also includes "technical criteria needed to support validation." **As CI/CD pipelines can result in software releases multiple times a day, automated self-attestation may be the only option**.

In contrast, **centralized validation *schemes* may not be sustainable as they may not evolve to address a changing threat landscape**. Previous centralized approaches to security certification were criticized for security requirements that were too difficult to understand, too expensive to implement, too narrow to provide adequate security protection, or too broad to be practical. They may also be prohibitive for smaller vendors, which can reduce software diversity, and thereby further centralize cyber risk.

If done correctly, conformance assessments may be useful to eventually inform consumer software labels. Traditionally **there are three different types of labels**, each with distinct benefits. First is a binary seal of approval, e.g., a USDA Organic label. Second is a graded label that indicates increasing or decreasing levels of risk, e.g., Energy Star labels. Finally, there are descriptive labels, e.g., nutrition labels. Binary labels may offer the highest usability but communicate the least amount of information. Descriptive labels provide the most information but may be less usable for low literacy consumers. **It is unlikely that any single approach will apply across the ecosystem**. Given the lack of examples in this domain, it may be prudent to prioritize further research on consumer software labels.

Finally, to "incentivize participation by consumer software developers**," NIST must begin by acknowledging the importance of SDL process over any point-in-time attribute of the software itself.** A robust SDL is grounded in a culture of security, reduces developer impact, and supports risk-based vulnerability discovery and remediation. Corresponding validation and conformance assessments may rely on voluntary, risk-based, self-assessments to allow for greater diversity of vendor participation, scalability, and evolution in response to new threats. Labels themselves must address user experience and at this time require further research.