Cybersecurity Labeling Programs for Consumers of IoT Devices and Software

David A. Wheeler <dwheeler@linuxfoundation.org> (et al.)
Director of Open Source Supply Chain Security, The Linux Foundation (*willing to speak*)

This 2-page paper responds to NIST's call regarding Cybersecurity Labeling Programs for Consumers: Internet of Things (IoT) Devices and Software, due 2021-08-17, for "suggestions and feedback on challenges and practical approaches to initiating cybersecurity labeling efforts for Internet of Things (IoT) devices and consumer software." The Linux Foundation's previous position papers for NIST (about the executive order on cybersecurity) on best practices (#2) and testing (#4) provide relevant information on this and should be considered. Our other position papers on criteria for critical software (#1), the use of critical software (#3), and integrity chains (#5) also have relevant information. See <https://www.nist.gov/itl/executive-order-improving-nations-cybersecurity/enhancing-software-supply-chain-security> for all of them. Here are a few additional comments as requested by NIST.

First, we believe it's vitally important to avoid negative impacts on developing or using open source software (OSS). The average software application contains 528 OSS components that constitute 70% of its total code. However, while OSS is vital for automation, it's too easy to create "requirements" that cannot be met in the real world. Many OSS projects allow anyone to participate and use the software at no cost, and there is often no revenue stream. Thus, requiring that every OSS project pay hundreds of thousands of dollars for an independent lab's audit ignores the reality that while some OSS projects can handle these costs, many others cannot. In addition, source code is a form of speech (per *Bernstein v. United States*), so any prior restraint of speech may be considered unconstitutional as well as being unhelpful. Instead, NIST must ensure that its recommendations are pragmatic and actually help consumers.

Regarding "formal and informal processes and practices used to secure the software development process," NIST should strive to reuse and build on existing work, including broadly adopted international standards and norms. The OpenSSF's CII Best Practices badge establishes criteria for security best practices in OSS, with about 4,000 participating projects and 600 passing projects. It uses self-attestation combined with automated checking and public posting of results, and has three achievement levels. The OpenSSF Security Scorecards project measures OSS security practices in an entirely automated way, scoring various practices. The draft OpenSSF SLSA framework guides engineers through gradually improving the security of their software. It can take years to achieve the ideal security state, therefore intermediate milestones are important. AllStar can enforce some policies on GitHub. Multi-organization project governance should be encouraged as it reduces risks, but that's not always achievable. The OpenSSF Best Practices Working Group (WG) actively works to identify and promulgate best practices, while ISO/IEC 5230 (OpenChain) provides a standardized approach to inbound, internal, and outbound compliance processes for OSS. Good practices include the use of multiple vulnerability detection tools in a continuous integration (CI) pipeline to rapidly detect vulnerabilities long before deployment.

Regarding "technical criteria needed to support validation of consumer software security assertions that reflect a baseline level of secure practices," see our previous position papers. Some additional criteria might be appropriate for consumer household internet-connectable (IoT) products sold to consumers at a cost (not free) where at least some non-trivial number (say 1,000) are sold. Added criteria may be justified as the money exchanged can be used to implement added requirements, such purchasers typically lack technical sophistication, and security impacts could be widespread. Such criteria must consider and not interfere with current & potential "right to repair" regulations. These criteria must also be simple, low cost, and clearly justified. NIST IR 8259A doesn't get there. In particular:

1. **Training in secure software development**: At least one lead developer of the IoT software must be trained in how to develop secure software. Today most software developers are neither formally nor informally trained, with obvious results. The OpenSSF (an LF foundation) offers free training courses on how to develop secure software, with an optional certificate available to demonstrate knowledge.
2. **Mandatory secure updates for vulnerabilities**: Externally-exploitable security vulnerabilities (including via product subcomponents) must be freely fixed in a timely way for at least 5 years after final sale *or* the customer must be offered a full refund. By default these fixes must be automatically provided and installed over the Internet if there is an Internet connection, and its cryptographic signature must be verified before it's automatically accepted. It must also be possible for a customer to delay or disable these updates if the customer actively chooses to do so. This fixes problems once realized and provides some financial incentive to make secure devices. Escrows or bonds could be used if an organization might be "disbanded" and recreated per product to evade this requirement.
3. **No default network passwords**: Devices sold to consumers must not have default passwords for network-accessible services that are determinable by others (e.g., because they're shared between devices). Manufacturers could, e.g., require users to set a password on first use.
4. **Use secure communication**: Where practical, these devices must use encrypted or at least cryptographically authenticated external communication protocols such as TLS and SSH. DNS can remain an exception in the short term, though that exception should eventually be removed as secure DNS mechanisms become commonplace.
5. **Randomness**: Test & provide evidence that any device's secure random number generation mechanism passes statistical tests for secure randomness if such a mechanism is used. This is a huge problem in IoT devices; at least 35 billion IoT devices would fail to pass such tests as of 2021. See You're Doing IoT RNG (presentation) by Dan Petro and Allan Cecil, DEF CON 29 (2021).
6. **Hardware interlocks**: Where economically practical, require hardware interlocks to enforce vital safety properties. It's easy to make subtle mistakes in software (e.g., see information on the Therac-25).
7. **SBOM in 5 years**: After a 5-year transition period, require that a software bill of materials (SBOM) be publicly and electronically available in an internationally standardized format, e.g., SPDX.

Regarding "how different conformity assessment approaches (e.g., vendor attestation, third-party conformity assessment) can be employed in consumer software labeling efforts," proposed approaches must be *affordable* and *timely*. The National Information Assurance Partnership (NIAP) implementation of the Common Criteria uses third-party assessments, but software rarely goes through this process because its costs and delays often exceed what is commercially viable. It is often better to use less-expensive measures, such as vendor attestation and automation, and then use other mechanisms (such as publication of attestations) to counter their weaknesses. Centralization often risks squelching innovation. There are millions of OSS projects; no one can pay for third-party assessment of them all. Also, third-party *conformity* assessment has much less value compared to third-party security audits. The LF *does* pay for third-party security audits of some vital OSS, but in those cases we don't do conformity assessments. We want secure software, not security theater. A good third-party security audit focuses on "is the software adequately secure" not "does it conform to some necessarily incomplete checklist." Checklists are helpful for experts, but they do not create experts.

Labeling programs should focus on helping consumers measure & manage risk (the likelihood & impact of issues), not eliminating all risk. Simple & clear measures *can* be helpful. In addition, ignoring OSS ignores reality. We noted earlier that modern applications are mostly OSS components; one simple measure might be "percent of incorporated OSS components that have achieved at least a passing CII Best Practices Badge." We hope these comments help.