2. Best Practices

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

> **NIST request:** *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.* This list of standards shall include criteria and required information for attestation of conformity by developers and suppliers. See EO Section 4(e)(i, ii, ix, and x).

Here are some places to look for best practices in the development of software:

1. The OpenSSF's CII Best Practices badge project specifically identifies best practices for open source software (OSS), focusing on security and including criteria to evaluate the security practices of developers and suppliers (it has over 3,800 participating projects). You can see the full set of criteria for all three badge levels (passing, silver, and gold).
2. LF is also working with SLSA (currently in development) as additional guidance focused on further addressing supply chain issues.
3. Best practices are only useful if developers understand them, yet most software developers have never received education or training in developing secure software. The LF has developed and released its Secure Software Development Fundamentals set of courses available on edX to anyone at no cost, and is available in markdown format. It also includes guidelines.
4. The OpenSSF Best Practices Working Group (WG) actively works to identify and promulgate security best practices and tools for open source developers and maintainers. We also provide a number of specific standards, tools, and best practices, as discussed below.
5. Encourage the use of memory safe languages. 70% of Microsoft's and Chrome's vulnerabilities are due to memory safety problems. A key best practice is to write software that is accessible from a network interface in a memory safe programming language. The Linux Foundation and the Internet Security Research Group (ISRG) who manages Let's Encrypt have developed a proposal and work plan to modernize network accessible, critical open source software to re-implement network interfaces in memory-safe programming languages, such as Rust. E.g., see Rustls improvements.
6. Our cloud computing industry collaborating within CNCF has also produced a guide for supporting software supply chain best practices for cloud systems and applications.
7. Reproducible builds are the primary way to give confidence that a built package matches its claimed source code. Otherwise, attacks such as those on the build system of SolarWinds' Orion will continue undetected. It is good to harden build systems against attack, but it is unwise to assume that attacks can never succeed. The documentation from reproducible-builds.org can be helpful.

8. Encryption can help counter many attacks. The EO 3(d) requires agencies to adopt "encryption for data at rest and in transit." Encryption in transit is implemented on the web using the TLS ("https://") protocol, and Let's Encrypt is the world's largest certificate authority for TLS certificates.
9. NIST may also want to look at the LF Confidential Computing Consortium is dedicated to defining and accelerating the adoption of confidential computing, that is protecting data *in use* (not just at rest and in transit) by performing computation in a hardware-based Trusted Execution Environment.
10. The Defense Industrial Board (DIB) Software Acquisition and Practices (SWAP) Study discusses developing software; its main report page 10 suggests some best practices.

We urge "*fixing upstream*" in open source software (OSS). Governments and contractors may receive OSS, fix and improve it locally in their environment, and be tempted to leave that improvement local to them even when others might benefit from sharing the improvements. However, that can lead to a "project fork," where the generally-used "upstream" version lacks the fix, and the local copy will typically be impractical to update and maintain to receive the other improvements from upstream. If the fixes are not sent upstream, the local fix will need to be re-engineered every time the local systems are updated to a newer version of the upstream OSS. Governments and contractors should instead attempt to "fix upstream" where practical. One example of "upstream first" is DARPA's initiative for Open Programmable Secure 5G that it has launched in collaboration with the LF.

When vulnerabilities *are* found, it's important to have and apply best practices in reporting them so that they can be fixed. The Forum of Incident Response and Security Teams (FIRST) has a number of useful documents about this, though one challenge is that they often focus on large organizations, and can currently be challenging to apply to small businesses or OSS projects. The OpenSSF Vulnerability Disclosures Working Group is working to help "mature and advocate well-managed vulnerability reporting and communication" for OSS.

One problem is that the US' Vulnerabilities Equities Process (VEP) sometimes fails to disclose vulnerabilities to the suppliers who can fix them. We hope the US will work more cooperatively with commercial organizations, including OSS projects, to share more vulnerability information. Every vulnerability that the US fails to disclose is a vulnerability that can be found and exploited by attackers, creating a "ticking time bomb" of the US' own making.