3. Use of critical software

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

> **NIST Request:** *Guidelines outlining security measures that shall be applied to the federal government's use of critical software*, including but not limited to, least privilege, network segmentation, and proper configuration. See E.O. Section 4(l).

Below are comments that apply to securely using software, including open source software (OSS), in critical systems generally:

1. *Shift left*. The government sometimes tries to take insecure software and magically make it "secure by configuration." This typically fails. You have to build security in, if you want it secure. Selecting software that is secure, or working with OSS projects to improve their security, is more effective than trying to configure security into an insecure program.
2. *Limit privilege*. Both programs and people should be given limited privilege in production environments.
3. *Real network segmentation*. Segment networks, where important, so that attackers *cannot* break through. Many so-called "air gaps" aren't real air gaps. A "virtual" air gap is just another way to say "not an air gap." For example, in 2011, RSA's seed warehouse was broken into, enabling attackers to subvert its SecurID tokens. This break-in occurred because the network was not fully segmented. A network with a technological air gap, yet can't work if another network is subverted, is not truly a network with an air gap.
4. *Assume network subversion*. Mere presence on a network is no guarantee that a component is authorized. Where practical, assume the network is compromised and implement a Zero Trust Architecture (at least where mere presence on a network does not imply authorization).
5. *Default configuration should be secure where practical*. Even if the software is designed so it *could* be secure, if that's not the default configuration, someone will eventually fail to apply the configuration (or revert it), and the less-common configuration is probably less tested. Instead, the government should ask suppliers to deliver *security by default*. In the case of open source software (OSS), the government should work with the OSS projects to help them improve the OSS so it has a secure configuration by default (which may require work). The NSA notably did this with the Linux kernel, developing a feature known as Security-Enhanced Linux (or SELinux) in 2000 which has become a default feature in many commercially supported Linux distributions, and today it is widely utilized in production systems.

6. *Automate secure configuration and detect misconfiguration*. If something must be configured security (because it can't be default), that should be maximally automated and misconfiguration (say from an update) should be automatically detected.
7. *Consider the whole stack*. Security and integrity must be present from boot including firmware, ROM, etc. all the way up the stack through to applications.
8. *Generate SBOMs*. Software Bill of Material (SBOM) information should be automatically created as a byproduct of development and building, so it can be made available to others for risk assessment (both pre- and post deployment).

Note that the use of 5G for Critical Software including Network Segmentation and Slicing is directly part of LF Networking's initiative called 5G Superblueprints, covering Open RAN (Radio Access Networks) to Edge/IOT to Core OSS including projects like ONAP and Magma. Best Practices are an integral part of its Security Subcommittee for all its projects.