Critical Software Software Verification

Barbara Guttman

September 14, 2021

Executive Order 14028, Section 4 Critical Software

- (g) Within 45 days of the date of this order, the Secretary of Commerce, acting through the Director of NIST, in consultation with the Secretary of Defense acting through the Director of the NSA, the Secretary of Homeland Security acting through the Director of CISA, the Director of OMB, and the Director of National Intelligence, shall publish a definition of the term “critical software” for inclusion in the guidance issued pursuant to subsection (e) of this section. That definition shall reflect the level of privilege or access required to function, integration and dependencies with other software, direct access to networking and computing resources, performance of a function critical to trust, and potential for harm if compromised.

Strategy

- Given the complexity of the software marketplace and the potential impact of this program, we (NIST, CISA & OMB) took a phased approach.
- What this means is that we will start with a subset of all possible software that could be considered critical and then build on it
- Advantages:
  - Learn what works and what doesn’t and improve as we go along
  - Allow time for coordination with other programs

What does this look like?

- So, this means we will have both a definition AND an initial phase
- We also have a table that lists the categories of software
- Lastly, we have some FAQs to answer what we think will be common questions.

Definition

- We decided on a definition that will be static across the phases.
- Here is the definition:

  ***EO-critical software* is defined as any software that has, or has direct software dependencies upon, one or more components with at least one of these attributes:**

  - **is designed to run with elevated privilege or manage privileges;**
  - **has direct or privileged access to networking or computing resources;**
  - **is designed to control access to data or operational technology;**
  - **performs a function critical to trust; or,**
  - **operates outside of normal trust boundaries with privileged access.**

Phases

- Initial phase:
- standalone, on-premises software that has security-critical functions or poses similar significant potentials for harm if compromised.
- Subsequent phases may address other categories of software such as those that control access to data; cloud-based and hybrid software; software development tools such as development tools, testing software, code repository systems, integration software, packaging software, and deployment software; software components in boot-level firmware; and software components in operational technology (OT).

Initial List of Categories of EO-Critical Software

- ICAM
- Operating systems, hypervisors, container environments
- Web browsers
- Endpoint security
- Network control
- Network protection
- Network monitoring and configuration
- Operational monitoring and analysis
- Remote scanning
- Remote access and configuration management
- Backup/recovery and remote storage

Operating systems, hypervisors, container environments	Software that establishes or manages access and control of hardware resources (bare metal or virtualized/ containerized) and provides common services such as access control, memory management, and runtime execution environments to software applications and/or interactive users	• Operating systems for servers, desktops, and mobile devices • Hypervisors and container runtime systems that support virtualized execution of operating systems and similar environments	• Highly privileged software with direct access and control of underlying hardware resources and that provides the most basic and critical trust and security functions
Network control	Software that implements protocols, algorithms, and functions to configure, control, monitor, and secure the flow of data across a network	• Routing protocols • DNS resolvers and servers • Software-defined network control protocols • Virtual private network (VPN) software • Host configuration protocols	• Privileged access to critical network control functions • Often subverted by malware as the first step in more sophisticated attacks to exfiltrate data

Identity, credential, and access management (ICAM)	**Software that centrally identifies, authenticates, manages access rights for, or enforces access decisions for organizational users, systems, and devices**	• **Identity management systems** • **Identity provider and federation services** • **Certificate issuers** • **Access brokers** • **Privileged access management software** • **Public key infrastructure**	• **Foundational for ensuring that only authorized users, systems, and devices can obtain access to sensitive information and functions**
Network monitoring and configuration	Network-based monitoring and management software with the ability to change the state of, or with installed agents or special privileges on, a wide range of systems	• Network management systems • Network configuration management tools • Network traffic monitoring systems	• Capable of monitoring and/or configuring enterprise IT systems using elevated privileges and/or remote installed agents

Web browsers	**Software that processes content delivered by web servers over a network, and is often used as the user interface to device and service configuration functions**	• **Standalone and embedded browsers**
Remote access and configuration management	Software for remote system administration and configuration of endpoints or remote control of other systems	• Policy management • Update/patch management • Application configuration management systems • Remote access/ sharing software • Asset discovery and inventory systems • Mobile device management systems

Remote scanning	**Software that determines the state of endpoints on a network by performing network scanning of exposed services**	• **Vulnerability detection and management software**	• **Typically has privileged access to network services and collects sensitive information about the vulnerabilities of other systems**
Endpoint security	Software installed on an endpoint, usually with elevated privileges which enable or contribute to the secure operation of the endpoint or enable the detailed collection of information about the endpoint	• Full disk encryption • Password managers • Software that searches for, removes, or quarantines malicious software • Software that reports the security state of the endpoint (vulnerabilities and configurations) • Software that collects detailed information about the state of the firmware, operating system, applications, user and service accounts, and runtime environment	• Has privileged access to data, security information, and services to enable deep inspection of both user and system data • Provides functions critical to trust

Backup/recovery and remote storage	**Software deployed to create copies and transfer data stored on endpoints or other networked devices**	• **Backup service systems** • **Recovery managers** • **Network-attached storage (NAS) and storage area network (SAN) software**	• **Privileged access to user and system data** • **Essential for performing response and recovery functions after a cyber incident (e.g., ransomware)**
Operational monitoring and analysis	Software deployed to report operational status and security information about remote systems and the software used to process, analyze, and respond to that information	• Security information and event management (SIEM) systems	• Software agents typically widely deployed with elevated privilege on remote systems • Analysis systems critical to incident detection and response and to forensic root cause analysis of security events • Often targeted by malware trying to deactivate or evade it

Network protection	Products that prevent malicious network traffic from entering or leaving a network segment or system boundary	• Firewalls, intrusion detection/ avoidance systems • Network-based policy enforcement points • Application firewalls and inspection systems	• Provides a function critical to trust, often with elevated privileges

Executive Order 14028, Section 4
Software Verification

“Section 4(r) Within 60 days of the date of this order, the Secretary of Commerce acting through the Director of NIST, in consultation with the Secretary of Defense acting through the Director of the NSA, shall publish guidelines recommending minimum standards for vendors’ testing of their software source code, including identifying recommended types of manual or automated testing (such as code review tools, static and dynamic analysis, software composition tools, and penetration testing).”

Software Verification

- Applicable to all software development
  - In house
  - Off the shelf
- Stand alone section of the EO

Approach

- Minimum standards recommended for verification by software vendors or developers.
- No single verification standard can encompass all types of software testing, be specific and prescriptive, and present efficient and effective testing.
- Expands on NIST’s Secure Software Development Framework (SSDF) practices
- Full document at the NIST EO Website - Guidelines on Minimum Standards for Developer Verification of Software

Structure

- 11 recommended minimums (+ fixing bugs!)
- Background and supplemental information about each technique
  - References for each technique
- Beyond software verification
  - Software development
  - Installation and operation
  - Additional software assurance techniques

Technique Class	Technique	Description & Reference to Recommended Minimums Document
Threat modeling	**Threat modeling helps identify key or potentially overlooked testing targets.**	Section 2.1. Threat modeling methods create an abstraction of the system, profiles of potential attackers and their goals and methods, and a catalog of potential threats. Threat modeling can identify design-level security issues and help focus verification.
Automated testing	**As testing is automated, it can be repeated often, for instance upon every commit or before an issue is retired.**	Section 2.2. Automated testing can run tests consistently, check results accurately, and minimize the need for human effort and expertise. Automated testing can be integrated into the existing workflow or issue tracking system.

Technique Class	Technique	Description & Reference to Recommended Minimums Document
Code-based (static) analysis	**Use a code scanner to look for top bugs.**	Section 2.3. Static analysis tools can check code for many kinds of vulnerabilities and for compliance with the organization’s coding standards. For multi-threaded or parallel processing software, use a scanner capable of detecting race conditions.
	Review for hardcoded secrets.	Section 2.4. Heuristic tools can be somewhat effective checking for hardcoded passwords and private encryption keys since functions or services taking these as parameters have specific interfaces.

Dynamic analysis (i.e., run the program on test cases)	**Run with built-in checks and protections.**	Section 2.5. Programming languages, both compiled and interpreted, provide many built-in checks and protections.
	Create "black box" test cases.	Section 2.6. "Black box" tests can address functional specifications or requirements, negative tests (invalid inputs and testing what the software should not do), denial of service and overload attempts, input boundary analysis, and input combinations.
	Create code-based structural test cases.	Section 2.7. Code-based, or structural, test cases are based on the implementation, that is, the specifics of the code. Code-based test cases may also come from coverage metrics.
	Use test cases created to catch previous bugs.	Section 2.8. Test cases which have been created to specifically show the presence (and later, the absence) of a bug can be used to identify issues in the absence of more general "first principles" assurance approaches for detecting bugs.
	Run a fuzzer.	Section 2.9. Fuzzers can try an immense number of inputs with minimal human supervision. The tools can be programmed with inputs that often reveal bugs, such as very long or empty inputs and special characters.
	If the software might be connected to the Internet, run a web app scanner.	Section 2.10. If there is a network interface, use a dynamic security testing tool (e.g., web application scanner) to detect vulnerabilities.

Technique Class	**Technique**	**Description & Reference to Recommended Minimums Document**
Check included software	**Use similar techniques to gain assurance that included libraries, packages, services, etc. are no less secure than your code.**	Section 2.11. Use the verification techniques recommended in this section to gain assurance that included code is at least as secure as code developed locally. The components of your software must be continually monitored against databases of known vulnerabilities; a new vulnerability in existing code may be reported at any time.
Fix bugs	**Fix critical bugs that are uncovered.**	Correct critical bugs as soon as possible and make process improvements necessary to prevent such bugs in the future, or to at least catch them earlier in the development process.

Questions?