National Aeronautics and
Space Administration
Mary W. Jackson NASA Headquarters
Washington, DC 20546-0001

MEMORANDUM FOR NASA CONTRACTOR AND GRANT/COOPERATIVE AGREEMENT SOFTWARE SUPPLIER COMMUNITY

SUBJECT: Supplier Documentation Requirements for Software Producers Offering Third-Party Software to NASA for Purchase and/or Use

References:

(a) Office of Management and Budget (OMB) Memorandum M-22-18 "Enhancing the Security of the Software Supply Chain through Software Development Practices" dated September 14, 2022, M-22-18 (whitehouse.gov)

(b) OMB Memorandum M-21-30 " Protecting Critical Software Through Enhanced Software Measures" dated August 10, 2021, M-21-30 (whitehouse.gov)

The following is notification and communication of NASA's process as required in references (a) and (b). Evidence of documentation is not required to be provided to NASA at this time. To ensure the integrity of its systems, and in compliance with requirements defined in Executive Order (EO) 14028, Improving the Nation's Cybersecurity, and references (a) and (b), NASA requires all software producers to comply with the above requirements in order to offer software to NASA for purchase and/or use. Producers must implement and attest to conformity with secure software development practices as defined by the National Institute of Standards and Technology (NIST). As NASA further defines its processes for collecting these documents from its vendors, additional requirements may be implemented and communicated.

SECURITY OF SOFTWARE DEVELOPMENT REQUIREMENTS

1. **SCOPE.** As defined in the Software Security Guidance Under Executive Order (EO) 14028 Section 4e (nist.gov), these requirements apply to all software acquired and/or used by NASA, which includes firmware, operating systems, applications, and application services (e.g., cloud-based software, open-source software, as well as products containing software). This also includes software renewals and major version changes.

2. **ENSURE ALL SOFTWARE COMPLIES WITH SECURE SOFTWARE DEVELOPMENT PRACTICES RECOMMENDED BY NIST.** Federal agencies, including NASA, "must only use software provided by software producers who can attest to complying with the Government-specified secure software development practices, as described in the NIST Guidance." This guidance references two documents, i.e., The NIST Secure Software Development Framework (SSDF), SP-800-218 and the NIST Software Supply Chain Security Guidance.

3. **COMPLETE THE SELF-ATTESTATION CHECKLIST.** Software producers must implement and attest to conformity with secure software development practices by completing and submitting the self-attestation checklist. This required checklist, derived from NIST SP 800-218, Table 1: The Secure Software Development Framework (SSDF) Version 1.1, is attached to this memo (see Enclosure) and must be accompanied by a statement of conformity on company/entity letterhead. Submissions of non-public information shall be sent to Agency-ICT-SCRM@nasa.onmicrosoft.com with a copy to the cognizant Contracting Officer (CO) in accordance with the applicable deadline.

4. **A THIRD PARTY ASSESSMENT MAY BE ACCEPTED IN LIEU OF THE SELF-ATTESTATION CHECKLIST.** A third-party assessment from a certified FedRAMP Third-Party Assessor Organization (3PAO) or other NASA-approved third party may be accepted if an entity is unable to complete the self-attestation checklist. This information is posted on the FedRAMP Marketplace.

5. **INABILITY TO ATTEST.** If a software producer cannot attest to one or more practices from NIST Guidance and the self-attestation form, all risks must be documented as well as the practices the producer has in place to mitigate those risks. Those documents, in addition to a Plan of Action & Milestones (POA&M) must be submitted to the cognizant NASA Contracting Officer (CO). The CO will coordinate with NASA's Office of the Chief Information Officer (OCIO) to make a risk-based determination on NASA's use of the software.

6. **NASA MAY REQUIRE ADDITIONAL DOCUMENTATION.** Based on the criticality of the software, NASA may require a Software Bill of Materials (SBOM). SBOMs must be generated in one of the data formats defined in the National Telecommunications and Information (NTIA) report "The Minimum Elements for a Software Bill of Materials (SBOM)" or successor guidance as published by the Cybersecurity and Infrastructure Security Agency (CISA).

The FAR Council has opened a proposed rule (FAR Case 2023-002, Supply Chain Software Security) to implement section 4(n) of Executive Order EO 14028. This rule will also focus on requirements outlined in OMB M-22-18. Status of FAR Cases are located at rpt_OpenFARCasesWebReports (osd.mil). Once the rule is finalized, relevant NASA acquisition policy may be updated to further implement the FAR rule and OMB guidance.

I appreciate your cooperation and continued commitment to the NASA mission. Please direct any further questions regarding this letter to your cognizant NASA CO.

Karla Jackson

Digitally signed by Karla Jackson
Date: 2023.01.18 14:20:18 -05'00'

Karla Smith Jackson
Assistant Administrator for Procurement

cc:
Jeffrey Seaton, Chief Information Officer
Marvin Horne, Deputy Assistant Administrator for Procurement
Julia Wise, Director, Procurement Management and Policy Division
Andre Sheppard, Acting Director, Procurement Strategic Operations Division
Geoffrey Sage, Director, Enterprise Services and Analysis Division
OP Procurement Officers (PO)/Deputy POs

Enclosure:

1. Checklist for Documenting Secure Software Development Activities in Support of EO 14028 Section 4e

Enclosure

Checklist for Documenting Secure Software Development Activities in Support of EO 14028 Section 4e

Practices	Tasks	Summary of Activities (including risk-based and mitigation actions in implementing the secure software development practices and tasks)	EO 14028 Subsections
Prepare the Organization (PO)			
Define Security Requirements for Software Development (PO.1): Ensure that security requirements for software development are known at all times so that they can be taken into account throughout the SDLC, and duplication of effort can be minimized because the requirements information can be collected once and shared. This includes requirements from internal sources (e.g., the organization's policies, business objectives, and risk management strategy) and external sources (e.g., applicable laws and regulations).	**PO.1.1**: Identify and document all security requirements for the organization's software development infrastructures and processes and maintain the requirements over time.		4e(ix)
	PO.1.2: Identify and document all security requirements for organization-developed software to meet and maintain the requirements over time.		4e(ix)
	PO.1.3: Communicate requirements to all third parties who will provide commercial software components to the organization for reuse by the organization's own software. [Formerly PW.3.1]		4e(vi) 4e(ix)
Implement Roles and Responsibilities (PO.2): Ensure that everyone inside and outside of the organization involved in the SDLC is prepared to perform their SDLC-related roles and responsibilities throughout the SDLC.	**PO.2.1**: Create new roles and alter responsibilities for existing roles as needed to encompass all parts of the SDLC. Periodically review and maintain the defined roles and responsibilities, updating them as needed.		4e(ix)
	PO.2.2: Provide role-based training for all personnel with responsibilities that contribute to secure development. Periodically review personnel proficiency and role-based training, and update the training as needed.		4e(ix)
	PO.2.3: Obtain upper management or authorizing official commitment to secure development and convey that commitment to all with development-related roles and responsibilities.		4e(ix)
Implement Supporting Toolchains (PO.3): Use automation to reduce human effort and improve the accuracy, reproducibility, usability, and comprehensiveness of security practices throughout the SDLC, as well as provide a way to document and demonstrate the use of these practices. Toolchains and tools may be used at different levels of the organization, such as organization-wide or project-specific, and may address a particular part of the	**PO.3.1**: Specify which tools or tool types must or should be included in each toolchain to mitigate identified risks, as well as how the toolchain components are to be integrated with each other.		4e(iii) 4e(ix)
	PO.3.2: Follow recommended security practices to deploy, operate, and maintain tools and toolchains.		4e(i)(F) 4e(ii) 4e(iii) 4e(v)

Practices	Tasks	Summary of Activities (including risk-based and mitigation actions in implementing the secure software development practices and tasks)	EO 14028 Subsections
SDLC, like a build pipeline.			4e(vi) 4e(ix)
	PO.3.3: Configure tools to generate artifacts of their support of secure software development practices as defined by the organization.		4e(i)(F) 4e(ii) 4e(v) 4e(ix)
Define and Use Criteria for Software Security Checks (PO.4): Help ensure that the software resulting from the SDLC meets the organization's expectations by defining and using criteria for checking the software's security during development.	**PO.4.1**: Define criteria for software security checks and track throughout the SDLC.		4e(iv) 4e(v) 4e(ix)
	PO.4.2: Implement processes, mechanisms, etc. to gather and safeguard the necessary information in support of the criteria.		4e(iv) 4e(v) 4e(ix)
Implement and Maintain Secure Environments for Software Development (PO.5): Ensure that all components of the environments for software development are strongly protected from internal and external threats to prevent compromises of the environments or the software being developed or maintained within them. Examples of environments for software development include development, build, test, and distribution environments.	**PO.5.1**: Separate and protect each environment involved in software development.		4e(i)(A) 4e(i)(B) 4e(i)(C) 4e(i)(D) 4e(i)(F) 4e(ii) 4e(iii) 4e(v) 4e(vi) 4e(ix)
	PO.5.2: Secure and harden development endpoints (i.e., endpoints for software designers, developers, testers, builders, etc.) to perform development-related tasks using a risk-based approach.		4e(i)(C) 4e(i)(E) 4e(i)(F) 4e(ii) 4e(iii) 4e(v) 4e(vi) 4e(ix)

Practices	Tasks	Summary of Activities (including risk-based and mitigation actions in implementing the secure software development practices and tasks)	EO 14028 Subsections
Protect Software (PS)			
Protect All Forms of Code from Unauthorized Access and Tampering (PS.1): Help prevent unauthorized changes to code, both inadvertent and intentional, which could circumvent or negate the intended security characteristics of the software. For code that is not intended to be publicly accessible, this helps prevent theft of the software and may make it more difficult or time-consuming for attackers to find vulnerabilities in the software.	**PS.1.1**: Store all forms of code – including source code, executable code, and configuration-as-code – based on the principle of least privilege so that only authorized personnel, tools, services, etc. have access.		4e(iii) 4e(iv) 4e(ix)
Provide a Mechanism for Verifying Software Release Integrity (PS.2): Help software acquirers ensure that the software they acquire is legitimate and has not been tampered with.	**PS.2.1**: Make software integrity verification information available to software acquirers.		4e(iii) 4e(ix) 4e(x)
Archive and Protect Each Software Release (PS.3): Preserve software releases in order to help identify, analyze, and eliminate vulnerabilities discovered in the software after release.	**PS.3.1**: Securely archive the necessary files and supporting data (e.g., integrity verification information, provenance data) to be retained for each software release.		4e(iii) 4e(vi) 4e(ix) 4e(x)
	PS.3.2: Collect, safeguard, maintain, and share provenance data for all components of each software release (e.g., in a software bill of materials [SBOM]).		4e(vi) 4e(vii) 4e(ix) 4e(x)
Produce Well-Secured Software (PW)			
Design Software to Meet Security Requirements and Mitigate Security Risks (PW.1): Identify and evaluate the security requirements for the software; determine what security risks the software is likely to face during operation and how the software's design and architecture should mitigate those risks; and justify any cases where risk-based analysis indicates that security requirements should be relaxed or waived. Addressing security requirements and risks during software design (secure by design) is key for improving software security and also helps improve development efficiency.	**PW.1.1**: Use forms of risk modeling – such as threat modeling, attack modeling, or attack surface mapping – to help assess the security risk for the software.		4e(ix)
	PW.1.2: Track and maintain the software's security requirements, risks, and design decisions.		4e(v) 4e(ix)
	PW.1.3: Where appropriate, build in support for using standardized security features and services (e.g., enabling software to integrate with existing log management, identity management, access control, and vulnerability management systems) instead of creating proprietary implementations of security features and services. [Formerly PW.4.3]		4e(ix)

Practices	Tasks	Summary of Activities (including risk-based and mitigation actions in implementing the secure software development practices and tasks)	EO 14028 Subsections
Review the Software Design to Verify Compliance with Security Requirements and Risk Information (PW.2): Help ensure that the software will meet the security requirements and satisfactorily address the identified risk information.	**PW.2.1**: Have 1) a qualified person (or people) who were not involved with the design and/or 2) automated processes instantiated in the toolchain review the software design to confirm and enforce that it meets all of the security requirements and satisfactorily addresses the identified risk information.		4e(iv) 4e(v) 4e(ix)
***Verify Third-Party Software Complies with Security Requirements (PW.3)**: Moved to PW.4*	***PW.3.1**: Moved to PO.1.3*		
	***PW.3.2**: Moved to PW.4.4*		
Reuse Existing, Well-Secured Software When Feasible Instead of Duplicating Functionality (PW.4): Lower the costs of software development, expedite software development, and decrease the likelihood of introducing additional security vulnerabilities into the software by reusing software modules and services that have already had their security posture checked. This is particularly important for software that implements security functionality, such as cryptographic modules and protocols.	**PW.4.1**: Acquire and maintain well-secured software components (e.g., software libraries, modules, middleware, frameworks) from commercial, open-source, and other third-party developers for use by the organization's software.		4e(iii) 4e(vi) 4e(ix) 4e(x)
	PW.4.2: Create and maintain well-secured software components in-house following SDLC processes to meet common internal software development needs that cannot be better met by third-party software components.		4e(ix)
	***PW.4.3**: Moved to PW.1.3*		
	PW.4.4: Verify that acquired commercial, open-source, and all other third-party software components comply with the requirements, as defined by the organization, throughout their life cycles.		4e(iii) 4e(iv) 4e(vi) 4e(ix) 4e(x)
	***PW.4.5**: Moved to PW.4.1 and PW.4.4*		
Create Source Code by Adhering to Secure Coding Practices (PW.5): Decrease the number of security vulnerabilities in the software and reduce costs by minimizing vulnerabilities introduced during source code creation that meet or exceed organization-defined vulnerability severity criteria.	**PW.5.1**: Follow all secure coding practices that are appropriate to the development languages and environment to meet the organization's requirements.		4e(iv) 4e(ix)
	***PW.5.2**: Moved to PW.5.1 as example*		
Configure the Compilation, Interpreter, and Build Processes to Improve Executable Security (PW.6): Decrease the number of security vulnerabilities in the software and reduce costs by eliminating vulnerabilities before testing occurs.	**PW.6.1**: Use compiler, interpreter, and build tools that offer features to improve executable security.		4e(iv) 4e(ix)
	PW.6.2: Determine which compiler, interpreter, and build tool features should be used and how each should be configured, then implement and use the approved		4e(iv) 4e(ix)

Practices	Tasks	Summary of Activities (including risk-based and mitigation actions in implementing the secure software development practices and tasks)	EO 14028 Subsections
	configurations.		
Review and/or Analyze Human-Readable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements (PW.7): Help identify vulnerabilities so that they can be corrected before the software is released to prevent exploitation. Using automated methods lowers the effort and resources needed to detect vulnerabilities. Human-readable code includes source code, scripts, and any other form of code that an organization deems human-readable.	**PW.7.1**: Determine whether code *review* (a person looks directly at the code to find issues) and/or code *analysis* (tools are used to find issues in code, either in a fully automated way or in conjunction with a person) should be used, as defined by the organization.		4e(iv) 4e(ix)
	PW.7.2: Perform the code review and/or code analysis based on the organization's secure coding standards, and record and triage all discovered issues and recommended remediations in the development team's workflow or issue tracking system.		4e(iv) 4e(v) 4e(ix)
Test Executable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements (PW.8): Help identify vulnerabilities so that they can be corrected before the software is released in order to prevent exploitation. Using automated methods lowers the effort and resources needed to detect vulnerabilities and improves traceability and repeatability. Executable code includes binaries, directly executed bytecode and source code, and any other form of code that an organization deems executable.	**PW.8.1**: Determine whether executable code testing should be performed to find vulnerabilities not identified by previous reviews, analysis, or testing and, if so, which types of testing should be used.		4e(ix)
	PW.8.2: Scope the testing, design the tests, perform the testing, and document the results, including recording and triaging all discovered issues and recommended remediations in the development team's workflow or issue tracking system.		4e(iv) 4e(v) 4e(ix)
Configure Software to Have Secure Settings by Default (PW.9): Help improve the security of the software at the time of installation to reduce the likelihood of the software being deployed with weak security settings, putting it at greater risk of compromise.	**PW.9.1**: Define a secure baseline by determining how to configure each setting that has an effect on security or a security-related setting so that the default settings are secure and do not weaken the security functions provided by the platform, network infrastructure, or services.		4e(iv) 4e(ix)
	PW.9.2: Implement the default settings (or groups of default settings, if applicable), and document each setting for software administrators.		4e(iv) 4e(ix)
Respond to Vulnerabilities (RV)			
Identify and Confirm Vulnerabilities on an Ongoing Basis (RV.1): Help ensure that vulnerabilities are identified more quickly so that they can be remediated more quickly in accordance with risk, reducing the window of opportunity for attackers.	**RV.1.1**: Gather information from software acquirers, users, and public sources on potential vulnerabilities in the software and third-party components that the software uses and investigate all credible reports.		4e(iv) 4e(vi) 4e(viii) 4e(ix)
	RV.1.2: Review, analyze, and/or test the software's code to identify or confirm the presence of previously undetected		4e(iv) 4e(vi)

Practices	Tasks	Summary of Activities (including risk-based and mitigation actions in implementing the secure software development practices and tasks)	EO 14028 Subsections
	vulnerabilities.		4e(viii) 4e(ix)
	RV.1.3: Have a policy that addresses vulnerability disclosure and remediation, and implement the roles, responsibilities, and processes needed to support that policy.		4e(viii) 4e(ix)
Assess, Prioritize, and Remediate Vulnerabilities (RV.2): Help ensure that vulnerabilities are remediated in accordance with risk to reduce the window of opportunity for attackers.	**RV.2.1**: Analyze each vulnerability to gather sufficient information about risk to plan its remediation or other risk response.		4e(iv) 4e(viii) 4e(ix)
	RV.2.2: Plan and implement risk responses for vulnerabilities.		4e(iv) 4e(v) 4e(viii) 4e(ix)
Analyze Vulnerabilities to Identify Their Root Causes (RV.3): Help reduce the frequency of vulnerabilities in the future.	**RV.3.1**: Analyze identified vulnerabilities to determine their root causes.		4e(ix)
	RV.3.2: Analyze the root causes over time to identify patterns, such as a particular secure coding practice not being followed consistently.		4e(ix)
	RV.3.3: Review the software for similar vulnerabilities to eradicate a class of vulnerabilities, and proactively fix them rather than waiting for external reports.		4e(iv) 4e(viii) 4e(ix)
	RV.3.4: Review the SDLC process and update it if appropriate to prevent (or reduce the likelihood of) the root cause recurring in updates to the software or in new software that is created.		4e(ix)