Headquarters
Department of the Army
Washington, DC
15 September 2021

Department of the Army Pamphlet 25–2–5

Information Management: Army Cybersecurity

Software Assurance

By Order of the Secretary of the Army:

JAMES C. MCCONVILLE
General, United States Army
Chief of Staff

Official:

MARK F. AVERILL
Administrative Assistant to the Secretary of the Army

History. This publication is an administrative revision. The portions affected by this administrative revision are listed in the summary of change.

Applicability. This regulation applies to the Regular Army, the Army National Guard/Army National Guard of the United States, and the U.S. Army Reserve, unless otherwise stated.

Proponent and exception authority. The proponent of this regulation is the Deputy Chief of Staff, G–6. The proponent has the authority to approve exceptions or waivers to this regulation that are consistent with controlling law and regulations. The proponent may delegate this approval authority, in writing, to a division chief within the proponent agency or its direct reporting unit or field operating agency, in the grade of colonel or the civilian equivalent. Activities may request a waiver to this regulation by providing justification that includes a full analysis of the expected benefits and must include formal review by the activity's senior legal officer. All waiver requests will be endorsed by the commander or senior leader of the requesting activity and forwarded through their higher headquarters to the policy proponent. Refer to AR 25–30 for specific guidance.

Suggested improvements. Users are invited to send comments and suggested improvements on DA Form 2028 (Recommended Changes to Publications and Blank Forms) directly to the Chief Information Officer (SAIS–PRP), 107 Army Pentagon, Washington, DC 20310–0107 or Users are invited to send comments and suggested improvements on DA Form 2028 (Recommended Changes to Publications and Blank Forms) via email to usarmy.pentagon.hqda-dcs-g-6.mbx.publications-management@army.mil.

Distribution. This pamphlet is available in electronic media only and is intended for the Regular Army, the Army National Guard/Army National Guard of the United States, and the U.S. Army Reserve.

Contents (Listed by chapter and page number)

Chapter 1
Introduction, *page 1*

Chapter 2
Identify Software, to Establish a Baseline Confidence Level, *page 2*

Chapter 3
Perform Software Assurance, *page 4*

Chapter 4
Manage Software Assurance Risk throughout the Software Life Cycle, *page 7*

Appendixes

A. References, *page 11*

Table List

Table 3–1: Software Assurance Resources, *page 4*

Figure List

Figure 1–1: Procedure synopsis, *page 2*

Glossary of Terms

Summary of Change

Chapter 1
Introduction

1–1. Purpose
This pamphlet provides procedures and resources on how to implement software assurance as a component of Army readiness. It applies to all Army organizations and materiel using software.

1–2. References, forms, and explanation of abbreviations
See appendix A. The abbreviations, brevity codes, and acronyms (ABCAs) used in this electronic publication are defined when you hover over them. All ABCAs are listed in the ABCA database located at https://armypubs.army.mil/abca/.

1–3. Associated publications
Policy associated with this pamphlet is found in AR 25–2.

1–4. Records management (recordkeeping) requirements
The records management requirement for all record numbers, associated forms, and reports required by this publication are addressed in the Records Retention Schedule–Army (RRS–A). Detailed information for all related record numbers, forms, and reports are located in Army Records Information Management System (ARIMS)/RRS–A at https://www.arims.army.mil. If any record numbers, forms, and reports are not current, addressed, and/or published correctly in ARIMS/RRS–A, see DA Pam 25–403 for guidance.

1–5. Overview
a. The Army continues to increase its reliance on software to fight and win our Nation's wars. Software as both materiel and a product is governed by AR 25–1, AR 70–1, and AR 702–19.

b. The Army continues to increase the quantity and complexity of software used to fight, plan, staff, organize, train, equip, and lead. As the Army marches towards the future end state described in the Army Vision, leaders must direct attention toward the protection of operational security by assuring the software they depend upon is fit for use and purpose. The U.S. Government, the Department of Defense (DoD), the Army, the Defense Industrial Base, and academia make significant resources, services, and solutions available to Army leaders so they can assure the software used to support their mission is fit for use and purpose.

c. Recognizing the changing character of warfare, this DA Pam seeks to proactively improve the way the Army procures, develops, and sustains software by empowering leaders with awareness of and access to appropriate software assurance tools, resources, and organizations. Army leaders who employ these software assurance techniques enhance cybersecurity and improve readiness for those systems dependent on software.

d. Army leaders must continually assess software assurance levels as part of risk management. Software assurance is defined as the level of confidence that software functions as intended and is free of vulnerabilities, intentionally or unintentionally designed or inserted as part of the software, throughout the life cycle. The first step towards improving software assurance is achieving software awareness within a command or organization. Like all materiel, the Army requires visibility of software assets as a legal, mission, and business imperative. Commanders and leaders must insist on the same or enhanced visibility into their own software assets as a way to assess risk.

e. In accordance with AR 25–2, the program manager (PM) or information system owner (ISO) will ensure (as part of supply chain risk management and through implementation of the Army's trusted systems and networks strategy) risks associated with software acquired for use in Army Information Technology (IT) are identified, evaluated, and managed. This is done prior to use and throughout the software life cycle. See Department of Defense Instruction (DoDI) 5200.44 and the system acquisition requirements of DoDI 5000.02 and DoDI 5000.75.

f. The Risk Management Framework (RMF) as defined in DoDI 8510.01, including reciprocity and inheritance, applies to software assurance.

1–6. Procedure synopsis
Procedure synopsis for performing software assurance (see fig 1–1).

Figure 1–1. Procedure synopsis

Chapter 2
Identify Software, to Establish a Baseline Confidence Level

The PM or ISO maintains software assurance information throughout the entire life cycle of the program or system. The procedure to identify software is to create a Software Products List (SPL), determine rights, and establish the baseline software assurance confidence level for each software product.

2–1. Create a Software Product List
Use the following procedure to create a list of software products for identifying software risk:

a. Identify software products. Record the following information for all software products:

(1) Manufacturer–Identify the company or organization who creates, owns, and licenses the software.

(2) Product Name–Name assigned to the software by the manufacturer.

(3) Version Number–Numbers assigned to the software by the manufacturer that identify a specific release.

(4) Release Date–Day, month, and year the software was released for use.

(5) Cryptographic Hash–Globally-unique value representing the output of a DoD approved cryptographic hash function. Software manufacturers may use different cryptographic functions to generate a hash for their software code or distributable binaries. Document the name of the software utility used to generate the cryptographic hash and how the utility was used. See cryptographic hash resources in table 3–1.

b. Determine software product type. The software product type may include one or more of the following:

(1) *Commercial off-the-shelf.* Software is commercial off-the-shelf (COTS) if it is a commercial item, sold in substantial quantities in the commercial marketplace, and is offered to the Government, without modification, in the same form which it is sold in the commercial marketplace.

(2) *Government off-the-shelf.* Software is Government off-the-shelf (GOTS) if it is developed exclusively for government use and the government typically owns the source code and data rights.

(3) *Open Source Software.* Open Source Software (OSS) is software for which the human-readable source code is available for use, study, re-use, modification, enhancement, and re-distribution by the users of that software.

c. Information used to obtain software. List information used to obtain the software in accordance with established procurement and acquisition procedures.

(1) Record the Army Portfolio Management Solution Army Information Technology Registry (AITR) Number for each software product in the SPL. The list may contain the same AITR number for more than one software product.

(2) Code.mil Registration. Identify all unclassified custom-developed source code created or paid for by the Army, regardless of data rights or open source status, created on or after August 2016, and provide information as prescribed by the guidelines and processes at https://code.mil/tell-us-about-your-

code.html. For more information, see Memorandum, DoD Chief Information Officer, subject: Custom-Developed Source Code Data Call, 12 October 2018. https://code.mil/assets/data-call-memo.pdf

2–2. Determine Software Usage, License, and Distribution Rights

Document usage rights, license type, and distribution methods for each item in the SPL:

a. Usage Rights.

(1) Document usage rights for the software. Documentation may include receipts, purchase orders, contract agreements, or software license agreements in accordance with the DoD Instruction for Accountability of Internal Use Software (see DoDI 5000.76).

b. License. Specify the software license type(s) in the SPL.

(1) *Open Source Software Licenses.* OSS licenses can be grouped into three main categories: Permissive, strongly protective, and weakly protective. Definitions for these and other software license terms are available at the DoD OSS Frequently Asked Questions in the references.

(2) *Proprietary Licenses.* These are also known as “closed source software” licenses obtained from a third-party for which a third-party retains intellectual property and other ownership rights such as copyright or patent. COTS licenses are typically proprietary.

(3) *Negotiated Rights.* Include any rights negotiated in accordance with the Defense Federal Acquisition Regulation Supplement such as Specially Negotiated License Rights for noncommercial computer software and noncommercial computer software documentation. Include any rights negotiated in a Customized Commercial License for commercial computer software or commercial computer software documentation. Review Army Directive 2018–26 prior to negotiations. Identify any rights in noncommercial computer software regarding Small Business Innovation Research. Case study example contract language is available in table 3–1.

c. Distribution. A description of the intended software distribution method in the SPL. Software can be distributed in many ways including direct download, administrative installation, and mobile application stores or channels, within the limits of the license agreement.

2–3. Establish the Baseline Software Assurance Confidence Level for Each Software Product

Establish the baseline software assurance confidence level for each software product in the SPL based on existing available information. Once established, the level is updated in the evaluation steps listed in chapter 3. The baseline software assurance confidence level is one of four possible values: “unknown”, “low confidence”, “moderate confidence”, or “high confidence”. All software products in the SPL begin with a baseline software assurance confidence level “unknown”. Complete the following steps for each software product in the SPL:

a. Review known weaknesses, vulnerabilities, and Security Technical Implementation Guides (STIG) findings.

(1) Obtain the source code files.

(2) Obtain documentation of existing source code static analysis report and software composition report if available from the original equipment manufacturer, procurement office, distribution vendor, or GOTS product owner.

(3) Review source code static analysis report for known weaknesses, known vulnerabilities, and STIG findings.

b. Identify intended functions by reviewing the requirements or specifications the software must meet to support the mission objective and end users.

(1) Review functional requirements documents, needs statements, and statements of objectives to identify if software functions are defined.

(2) Review unit tests, function tests, and acceptance tests to verify the scope of software functionality testing.

(3) Review software source documents containing an explanation of how the software was obtained.

c. Using the results from a. and b. above, determine and record the baseline confidence level for all products in the SPL.

(1) Software confidence level is "high" if the software is free from critical or high-level weaknesses/ vulnerabilities/STIG findings, functions as intended, and source code is available.

(2) Software confidence level is "moderate" when only one of the following conditions is true: the software contains critical or high-level weaknesses/vulnerabilities/STIG findings, does not function as intended, or the source code is unavailable.

(3) Software confidence level is "low" when two or more of the following conditions are true: the software contains critical or high-level weaknesses/vulnerabilities/STIG findings, does not function as intended, or the source code is unavailable.

2–4. Maintain software confidence levels

The PM or ISO maintain information gathered in paragraphs 2–1 through 2–3 throughout the entire life cycle of the program or system in accordance with established configuration management policies and procedures. Chapter 3 describes further detailed evaluation steps to increase software assurance. Chapter 4 describes when to apply the processes in chapter 3 throughout the software life cycle.

Chapter 3
Perform Software Assurance

The PM or ISO is accountable to perform software assurance. This task may be assigned to a responsible party such as a software development team, third-party vendor, or software assurance service provider. Refer to table 3–1 for additional resources. This occurs whenever the PM or ISO has a need to determine the software assurance confidence level for one or more software products. This procedure may recur frequently for a given program, project, product, or system, to support the objective level of mission assurance and protection. The output of this evaluation is a Software Assurance Risk Register (SARR). Every entry in the SARR must be evaluated and include a likelihood of exploitation and a severity of the consequence of exploitation, in addition to other information described in the sections below.

3–1. Analyze Software Assurance Risk

The output of this procedure is an updated SARR. For each product in the SPL:

a. Identify and analyze software assurance risks based on weaknesses, vulnerabilities, attack patterns, and STIGs. Perform static analysis (source code, binary, or bytecode) and record other detection methods used, such as:

(1) Architecture or design review.
(2) Dynamic analysis with automated or manual results interpretation.
(3) Software composition analysis.
(4) Black box, gray box, white box.

b. Choose a reference list of common software security weaknesses (see table 3–1). Use selected detection method(s) to analyze the software product and determine if it contains any of the listed weaknesses. For each identified weakness create a new entry in the SARR. Identify the point(s) in the software life cycle at which introduction of the identified weakness may occur. Describe the possible consequences associated with the weakness and the application security area that is violated.

Table 3–1
Software Assurance Resources

Reference list	Web site	Description
DoD Joint Federated Assurance Center (JFAC)	Perform a web search for “JFAC”	JFAC member organizations and their technical service providers interact with program offices and other interested parties to provide software and hardware assurance expertise and support, to include vulnerability assessment, detection, analysis, and remediation services, information about emerging threats and capabilities, software and hardware assessment tools and services, and best practices.
DoD Software Assurance Tools and Providers	https://jfac.navy.mil/JFAC/software/providers/usa	List of JFAC technical service providers.

Table 3–1
Software Assurance Resources—Continued

Reference list	Web site	Description
Incorporating Software Assurance into DoD Acquisition Contracts	https://jfac.navy.mil/JFAC/resources/contracts	This document suggests language that programs might tailor for use in Request for Proposal packages and contracts to provide the program office with insight into software development activities of its contractors and to provide assurance regarding the developed software's ability to meet mission needs.
DoD Security Technical Implementation Guides	https://cyber.mil/stigs/	STIGs that are the configuration standards and technical guidance to "lock down" software that might otherwise be vulnerable to a malicious computer attack.
Risk Management Framework (RMF) Knowledge Service (KS)	https://rmfks.osd.mil/	RMF KS is DoD's official site for enterprise RMF policy and implementation guidelines.
DoD Software Assurance Community of Practice	https://www.intelink.gov/go/4k3KvLb	Web site for members of the DoD Software Assurance community to collaborate and capture information.
DoD Cyber Security and Information Systems Information Analysis Center (CSIAC)	https://www.csiac.org/	CSIAC that provides DoD with a central point of access for Information Assurance (IA) and Cybersecurity to include emerging technologies in system vulnerabilities, Research and Development, models, and analysis to support the development and implementation of effective defense against information warfare attacks.
Design and Development Process for Assured Software–Department of Defense Software Assurance Community of Practice: Volume 1	https://www.csiac.org (Search for "Design and Development Process for Assured Software")	Describes different aspects of developing, deploying and training on how to build assured software.
Tools & Testing Techniques for Assured Software–Department of Defense Software Assurance Community of Practice: Volume 2	https://www.csiac.org (Search for "Tools & Testing Techniques for Assured Software")	Describes different aspects of software assurance competencies that can be used to improve software assurance functions and how to develop/deploy assured software throughout the life cycle acquisition process.
DoD Secure Coding Guidelines	https://intellipedia.intelink.gov/wiki/Secure_Coding_Guidelines	Secure Coding Guidelines, also called Software Assurance Implementation Guidelines and Secure Implementation Guidelines, are guidelines for writing software code during its implementation that prevent or reduce the risk of failures in software assurance.
Code.mil	https://www.code.mil	Code.mil supports collaboration within the developer community across the world on DoD open source projects.

Table 3–1
Software Assurance Resources—Continued

Reference list	Web site	Description
Army Cybersecurity Portal (One-Stop Shop)	https://www.milsuite.mil/wiki/Portal:Army_Cyber-security	This website catalogs cyber security-related directives, laws, regulations, and guidance applicable to the Army.
DoD Developer's Guidebook for Software Assurance	https://resources.sei.cmu.edu/asset_files/Spe-cialReport/2018_003_001_538761.pdf	Provides a bottom-up approach to tool selection, considering which activities and tools are normally appropriate at different stages of the development or product life cycle.
Program Manager's Guidebook for Software Assurance	https://resources.sei.cmu.edu/asset_files/Spe-cialReport/2018_003_001_538779.pdf	Aids the PM in understanding and addressing the software assurance responsibilities critical in defending software-intensive systems.
Common Weakness Enumeration (CWE)	https://cwe.mitre.org/	CWE is a community-developed catalog of common software security weaknesses, which can be used as a baseline for weakness identification, mitigation, and prevention efforts.
Software Assurance Reference Dataset (SARD)	https://samate.nist.gov/index.php/SARD.html	National Institute of Standards and Technology (NIST) SARD provides a dataset encompassing a wide variety of flaws, languages, platforms, and compilers.
National Vulnerability Database (NVD)	https://nvd.nist.gov/	The NVD is the U.S. government repository of standards-based vulnerability management data, represented using the Security Content Automation Protocol.
Common Attack Pattern Enumeration and Classification (CAPEC)	https://capec.mitre.org/	A comprehensive dictionary and classification taxonomy of known attacks that can be used by analysts, developers, testers, and educators to advance community understanding and enhance defenses.
National Institute of Standards and Technology (NIST) Software Quality Group	https://www.nist.gov/itl/ssd/software-quality-group	The Software Quality Group develops tools, methods, and related models for improving the processes for ensuring that software behaves correctly and for identifying software defects.
National Cybersecurity and Communications Integration Center (NCCIC)	https://www.us-cert.gov/	The NCCIC mission is to reduce the risk of systemic cybersecurity and communications challenges in its role as the nation's flagship cyber defense, incident response, and operational integration center.
Cyber.mil and Approved Cryptographic Hash	https://cyber.mil/	The DoD Cyber Exchange provides one-stop access to cyber information, policy, guidance and training for cyber professionals throughout the DoD and in the general public. These resources are provided to enable the user to comply with rules,

Table 3–1
Software Assurance Resources—Continued

Reference list	Web site	Description
		regulations, best practices, and federal laws. For approved cryptographic hash, search "approved hash" or https://cyber.mil/?s=approved+hash.

c. Choose a reference list of common software vulnerabilities (see table 3–1). Use selected detection method(s) to analyze the software product and determine if it contains any of the listed vulnerabilities. For each identified vulnerability, create a new entry in the SARR. Identify the attack vector(s), attack complexity, privileges required, user interaction, scope, and impact.

d. List possible attack patterns using publicly available catalogs for the software product and target system or environment (see table 3–1). For each identified attack pattern, create a new entry in the SARR. List the typical severity, likelihood, prerequisites, impact, and consequences for the attack pattern.

e. Evaluate each software product using the "Application Security and Development" STIG and record all STIG findings in the SARR.

f. Identify and analyze software assurance risks based on intended functions.

(1) Based on the life cycle phase, perform developmental and operational test and evaluation to assess functionality. Record any areas not addressed by the above as an entry in the SARR.

(2) Perform a software architecture review which may include one or more of the following: attack surface modeling, architecture tradeoff analysis method, architecture analysis and design language, system theoretic accident model and process, and systems theoretic process analysis. Record any areas not addressed by the above as an entry in the SARR.

3–2. Identify and Document the Management Approach for Software Assurance Risk

The output of this step is the initial management approach for each risk in the SARR. Software assurance risk is managed in one or more of the following ways: remediate, accept, mitigate, or avoid. It is documented in accordance with the DoD RMF process.

a. Remediate software assurance risk by changing the software to directly remove the source of the risk.

b. Accept software assurance risk by documenting the risk for action officer's approval.

c. Mitigate software assurance risk by implementing measures to reduce the risk.

d. Avoid software assurance risk by restructuring activities to eliminate the likelihood of exploitation or severity of the consequence.

Chapter 4
Manage Software Assurance Risk throughout the Software Life Cycle

PMs or ISOs prioritize and manage software assurance risk throughout the software life cycle by reviewing and implementing the activities in this chapter. Determine the applicable life cycle phase(s) for each product in the SPL. Implement and monitor the risk controls identified in SARR during the appropriate life cycle phase(s).

4–1. Management Actions for All Life Cycle Phases

a. PMs or ISOs complete the following management actions across the software life cycle:

(1) Address software assurance early to avoid rapid cost growth.
(2) Address software assurance throughout the life cycle.
(3) Collect evidence to support assurance claims.
(4) Designate roles for software assurance responsibilities.
(5) Create and manage software assurance plans, reports, and artifacts.
(6) Consider risks in all sources of software.
(7) Budget for software assurance.
(8) Determine effectiveness of software assurance activities to ensure mission success.

b. Army software developers complete the following actions across the software life cycle:

(1) Use threat models to develop and implement threat mitigations in design, code, and test cases.

(2) Identify and document security requirements early in the development life cycle. Evaluate development artifacts for compliance with security requirements.

(3) Create a software architecture, and design software to implement and enforce security policies.

(4) Keep the design as simple and as small as possible.

(5) Base access decisions on permission rather than exclusion. Implement software whereby default access is denied and the protection scheme identifies conditions under which access is permitted.

(6) Adhere to the principle of least privilege by ensuring each process executes with the least set of privileges necessary to function as intended.

(7) Filter data sent to other systems according to security guidelines. This includes complex subsystems such as command shells, relational databases, and COTS components.

(8) Practice defense in depth. Manage risk with multiple defensive strategies so that if one layer of defense turns out to be inadequate, another layer of defense can prevent a security flaw from becoming an exploitable vulnerability, or limit the consequences of a successful exploitation, or both.

(9) Validate input from all untrusted data sources. Anything outside the application should be considered untrusted.

(10) Implement a secure coding standard (see table 3–1).

(11) Heed compiler warnings. Compile code using the highest warning level available for your compiler, and eliminate warnings by modifying the code. Use source code and binary analysis tools and methods to continually identify risk and update the SARR.

(12) Perform quality assurance using techniques such as fuzz testing, penetration testing, and source code audits.

c. Related activities within each life cycle phase are described in the remainder of this chapter. Work products, measurements, and additional information for each action are available in the resources listed in table 3–1.

4–2. Concept Phase

a. Demonstrate understanding of the software assurance risks by completing the following activities:

(1) Perform an initial threat assessment to form a basis for deriving software assurance and cybersecurity requirements.

(2) Establish a requirements management system and document software assurance requirements.

(3) Document requirements elicitation and participation by stakeholders.

(4) Assign software assurance roles and responsibilities.

(5) Incorporate software assurance requirements into acquisition and procurement contract documents. Table 3–1 provides resources for incorporating DoD software assurance requirements into contract documents.

(6) Train program personnel in the tools and processes of software assurance analysis and verification.

b. Develop a baseline life cycle software assurance strategy by completing the following activities:

(1) Identify and inventory software to be managed using the procedures in chapter 2.

(2) Create a mission-driven schedule to evaluate software assurance using the procedures in chapter 3.

c. Incorporate software assurance into the software architecture by completing the following activities:

(1) Establish misuse and abuse cases (based on threat expectations) that will be addressed by the architecture.

(2) Establish and review baseline architecture using procedures described in chapters 2 and 3.

(3) Review options for programming language, architecture, and operational environments.

(4) Identify sources to perform software assurance such as organizational assets or service providers.

d. Include secure design principles by completing the following activities:

(1) Incorporate into development the principles of system element isolation, least common mechanism, least privilege, fault isolation, and input checking and validation.

(2) Include verification and enforcement during structured design reviews.

(3) Define functional requirements for software operations in a cyber-contested environment.

(4) Derive non-functional software assurance requirements.

(5) Develop a secure architecture and obtain data rights.

4–3. Development Phase

a. Enforce secure coding practices to remove potential software vulnerabilities by conducting the following activities:

(1) Implement software assurance process for development, including configuration control, tools, and automation.

(2) Perform code inspection and analysis using secure coding rules and guidelines, tools, processes, and methods described in *paragraph 3–1a*, such as knowledge management. Address identified coding violations as risks in the SARR.

b. Enforce software assurance testing by completing the following activities:

(1) Perform testing such as fuzz testing, sandbox testing, penetration testing, test (code) coverage metrics, or branch points.

(2) Develop and execute test cases as informed by threat modeling (use results from paragraph 3–1a(3)), misuse and abuse cases, and assign to testing events.

(3) Perform regression testing. Include in the regression analysis negative test cases that fail if security mechanisms work properly.

(4) Perform functional testing using documents identified in paragraph 2–3b(2).

c. Enforce configuration management and version control in accordance with the Security Plan required by DoDI 8510.01, Enclosure 6.

(1) Maintain traceability between configuration management processes and risk management processes by listing which software files (source code, binary, documentation, or configuration) implement the risk controls selected in *paragraph 3–2a*.

(2) Update the SARR to list which software files implement the selected risk controls. Apply version control, revision control, and configuration management to all software files.

4–4. Production Phase

a. Establish integration assurances during production by conducting the following activities:

(1) Integrate software assurance risks in the SARR with non-software system-level risks to appropriately secure the integrated production system.

(2) Automate regression testing (post integration) to address software assurance test cases assigned to integration testing using documents identified in paragraph 2–3b(2).

(3) Establish responsibility from development to sustainment for all items in the SPL.

(4) Confirm configuration management of source code is traceable to the responsible activity.

(5) Ensure test coverage of all functions and components using documents identified in paragraph 2–3b(2).

(6) Conduct third-party software assurance testing if required by controls selected in paragraph 3–2. Third-party software assurance providers are listed in table 3–1.

b. Establish deployment assurances by conducting the following activities:

(1) Confirm receipt of software from all sources.

(2) Assign operational responsibility for management of each risk in the SARR.

(3) Provide the system operator with artifacts necessary to continue software assurance risk management in the production phase. Artifacts may include the following: build environments, test suites, scanning tools, coding rules, guidelines, and data rights.

4–5. Utilization Phase

a. Maintain the SPL using procedures described in chapter 2.

b. Incorporate mission and operational monitoring into software assurance evaluations by updating threat models, raising software assurance posture, adapting to mission evolution, and adapting to technology evolution.

c. Monitor and address threats and attacks, which exploit any software in the SPL. Review threat intelligence, attack, and breach reports.

d. Gather and manage data to model typical user behavior and identify any new functional risks.

e. Update the SARR based on findings using procedures in chapter 3.

4–6. Support Phase

a. Maintain software assurance for all software in the SPL. Plan and implement continual software assurance risk control actions for all risks in the SARR using procedures described in chapter 3.

b. Plan for software upgrades, updates, and patching based on available maintenance, assurance, and support resources. Control the tools, processes, and methods for deploying software upgrades, technical refreshes, maintenance releases, and patches to maintain situational awareness of software assurance risk.

c. Reevaluate software confidence level as a result of patches, version upgrades, and changes in documentation.

4–7. Retirement Phase

Create, maintain, and dispose of software records to include the SPL, SARR, licenses, and source code in accordance with AR 25–400–2.

Appendix A

References

Section I

Required Publications

AR 25–2
Army Cybersecurity (Cited in para 1–3.)

Section II

Prescribed Forms

This section contains no entries.

Glossary of Terms

Attack patterns
Similar cyber events or behaviors that may indicate an attack has occurred or is occurring, resulting in a security violation or a potential security violation. For software, descriptions of common methods for exploiting software systems.

Attack surface
The set of ways in which an adversary can enter a system and potentially cause damage. An information System's (IS's) characteristics that permit an adversary to probe, attack, or maintain presence in the IS.

Commercial off-the-shelf
Software that is a non-developmental item of supply that is both commercial and sold in substantial quantities in the commercial marketplace, and that can be procured or utilized under government contract in the same precise form as available to the general public.

Common attack pattern enumeration and classification
A comprehensive dictionary and classification taxonomy of known attacks that can be used by analysts, developers, testers, and educators to advance community understanding and enhance defenses. The CAPEC effort provides a publicly available catalog of attack patterns along with a comprehensive schema and classification taxonomy. CAPEC is sponsored by the United States Computer Emergency Readiness Team (US–CERT) in the Office of Cybersecurity and Communications at U.S. Department of Homeland Security (DHS).

Common vulnerabilities and exposures
A dictionary of common names (that is, common vulnerabilities and exposures identifiers) for publicly known cybersecurity vulnerabilities. Common vulnerabilities and exposures common identifiers make it easier to share data across separate network security databases and tools and provide a baseline for evaluating the coverage of an organization's security tools. The National Cybersecurity, a federally funded research and development center operated by The MITRE Corporation, maintains the system with funding from the National Cybersecurity Division of DHS.

Common weakness enumeration
A community-developed list of common software security weaknesses. CWE is a formal list of software weakness types created to: (1) serve as a common language for describing software security weaknesses in architecture, design, or code; (2) serve as a standard measuring stick for software security tools targeting these weaknesses; and (3) provide a common baseline standard for weakness identification, mitigation, and prevention efforts. CWE is sponsored by US–CERT in the Office of Cybersecurity and Communications at DHS.

Cybersecurity
Prevention of damage to, protection of, and restoration of computers, electronic communications systems, electronic communications services, wire communications, and electronic communications, including information contained therein, to ensure their availability, integrity, authentication, confidentiality, and nonrepudiation.

Fault isolation
Software mechanisms that isolate faults include functions to trap, log, and otherwise protect element failures from affecting other elements and the larger system. Logs help trace the sources of operational faults. Logs also can be examined to help assess whether the fault is indicative of a malicious attack.

Government off-the-shelf
Software that is ready-to-use software products created and owned by the Government

Information Assurance
The protection of systems and information in storage, processing, or transit from unauthorized access or modification; denial of service to unauthorized users; or the provision of service to authorized users. It also includes those measures necessary to detect, document, and counter such threats. Measures that protect and defend information and information systems (ISs) by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation. This includes providing for restoration of ISs by incorporating protection, detection, and reaction capabilities. This regulation designates IA as the security

discipline that encompasses Communications Security, Information Security, and control of compromising emanations (telecommunications electronics materiel protected from emanating spurious transmissions).

Information System owner
The system owner is an organizational official responsible for the procurement, development, integration, modification, operation, maintenance, and disposal of a system.

Input checking/validation
Input checking and validation should ensure that out-of-bounds values and out-of-sequence operations are handled without causing failures, and that the invalid input events are logged. This checking may be applied to developmental software through coding guidelines and review. It may also apply to COTS and non-developmental item software through constructs such as wrappers and input filtering.

Least common mechanism
The principle of least common mechanism states that mechanisms used to access resources should not be shared.

Mission Assurance
A process to protect or ensure the continued function and resilience of capabilities and assets -including personnel, equipment, facilities, networks, information and IS, infrastructure, and supply chains -critical to the execution of DoD mission-essential functions in any operating environment or condition.

Misuse/abuse case
A type of interaction between a system and one or more actors, where the results of the interaction are harmful to the system, one of the actors, or one of the stakeholders in the system.

Open Source Software
A type of COTS software whose source code is published and made available to the public, enabling anyone to copy, modify, and redistribute the source code without paying royalties or fees subject to the terms of the associated license.

Principle of least privilege
Principle requiring that each subject be granted the most restrictive set of privileges needed for the performance of authorized tasks. Application of this principle limits the damage that can result from accident, error, or unauthorized use of IT.

Product manager
Person who is delegated authority and assigned responsibility for centralized management of a development or acquisition program that does not qualify for project management. For software assurance matters, the product manager is responsible for meeting cybersecurity requirements that are referred to as controls under the RMF.

Program manager
The single individual responsible for a project or program who manages all daily aspects of the project or program.

Risk Management Framework
A structured approach used to oversee and manage risk for an enterprise

Security Technical Implementation Guide
A compendium of DoD policies, security regulations, and best practices for securing an IA or IA-enabled device (for example, operating system, network, and application software). The Application Security STIG ensures that processes are in place to develop secure code.

Service provider
An organization that provides one or more cybersecurity services to implement and protect the DoD information network.

Software
A set of computer programs, procedures, and associated documentation concerned with the operation of a data processing system (for example, compiler, library routines, manuals, and circuit diagrams); usually contrasted with hardware.

Software Assurance
Both the level of confidence in, and processes in place to provide assurance that, software used by the Army functions as intended and is free of intentional and unintentional vulnerabilities.

Software developer
The developer is responsible for programmatic coding regarding applications, software, and Internet/intranet sites, including “secure coding,” as well as coordinating and working with the Configuration Management manager to identify, resolve, and implement controls and other Configuration Management issues.

System element isolation
Software following the principle of system element isolation allows system element functions to operate without interference from other elements. Such isolation limits the cascading effect that could ensue due to compromise of a single element.

Threat
Capabilities, intentions, and attack methods of adversaries to exploit, damage, or alter information or an IS. Also, any circumstance or event with the potential to cause harm to information or an IS. Any circumstance or event with the potential to adversely impact an IS through unauthorized access, destruction, disclosure, modification of data, or denial of service.

Vulnerability
Weakness in an IS, system security procedures, internal controls, or implementation that could be exploited by a threat source.

Weakness
A shortcoming or imperfection in software code, design, architecture, or deployment that, under proper conditions, could become a vulnerability or contribute to the introduction of vulnerabilities.

SUMMARY of CHANGE

DA PAM 25–2–5
Software Assurance

This administrative revision, dated 1 November 2022—

- Changes the proponency from CIO/G–6 to the Deputy Chief of Staff, G–6 (title page.)

This new Department of the Army pamphlet, dated 15 September 2021—

- Provides Army personnel (military, civilians, and contractors) with procedures and resources to implement software assurance as a component of Army readiness (throughout).

- Addresses software as an element of all information technology investments owned and maintained by Army organizations across the Warfighting, Business, Defense Intelligence, and Enterprise Information Environment Mission Areas. These include, but are not limited to research, development, test, and evaluation appropriations; procurement appropriations; military personnel appropriations; operations and maintenance appropriations; and the Defense Working Capital Fund. Software Assurance is a type of product assurance and is a component of life cycle management (throughout).