3.17 SYSTEM AND SERVICES ACQUISITION

Quick link to System and Services Acquisition Summary Table

SA-1 POLICY AND PROCEDURES

Control:

a. Develop, document, and disseminate to [*Assignment: organization-defined personnel or roles*]:

   1. [*Selection (one or more): Organization-level; Mission/business process-level; System-level*] system and services acquisition policy that:

      (a) Addresses purpose, scope, roles, responsibilities, management commitment, coordination among organizational entities, and compliance; and

      (b) Is consistent with applicable laws, executive orders, directives, regulations, policies, standards, and guidelines; and

   2. Procedures to facilitate the implementation of the system and services acquisition policy and the associated system and services acquisition controls;

b. Designate an [*Assignment: organization-defined official*] to manage the development, documentation, and dissemination of the system and services acquisition policy and procedures; and

c. Review and update the current system and services acquisition:

   1. Policy [*Assignment: organization-defined frequency*] and following [*Assignment: organization-defined events*]; and

   2. Procedures [*Assignment: organization-defined frequency*] and following [*Assignment: organization-defined events*].

Discussion: System and services acquisition policy and procedures address the controls in the SA family that are implemented within systems and organizations. The risk management strategy is an important factor in establishing such policies and procedures. Policies and procedures contribute to security and privacy assurance. Therefore, it is important that security and privacy programs collaborate on the development of system and services acquisition policy and procedures. Security and privacy program policies and procedures at the organization level are preferable, in general, and may obviate the need for mission- or system-specific policies and procedures. The policy can be included as part of the general security and privacy policy or be represented by multiple policies that reflect the complex nature of organizations. Procedures can be established for security and privacy programs, for mission or business processes, and for systems, if needed. Procedures describe how the policies or controls are implemented and can be directed at the individual or role that is the object of the procedure. Procedures can be documented in system security and privacy plans or in one or more separate documents. Events that may precipitate an update to system and services acquisition policy and procedures include assessment or audit findings, security incidents or breaches, or changes in laws, executive orders, directives, regulations, policies, standards, and guidelines. Simply restating controls does not constitute an organizational policy or procedure.

Related Controls: PM-9, PS-8, SA-8, SI-12.

Control Enhancements: None.

References: [OMB A-130], [SP 800-12], [SP 800-30], [SP 800-39], [SP 800-100], [SP 800-160-1].

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

SA-2 ALLOCATION OF RESOURCES

Control:

a. Determine the high-level information security and privacy requirements for the system or system service in mission and business process planning;

b. Determine, document, and allocate the resources required to protect the system or system service as part of the organizational capital planning and investment control process; and

c. Establish a discrete line item for information security and privacy in organizational programming and budgeting documentation.

Discussion: Resource allocation for information security and privacy includes funding for system and services acquisition, sustainment, and supply chain-related risks throughout the system development life cycle.

Related Controls: PL-7, PM-3, PM-11, SA-9, SR-3, SR-5.

Control Enhancements: None.

References: [OMB A-130], [SP 800-37], [SP 800-160-1].

SA-3 SYSTEM DEVELOPMENT LIFE CYCLE

Control:

a. Acquire, develop, and manage the system using [*Assignment: organization-defined system development life cycle*] that incorporates information security and privacy considerations;

b. Define and document information security and privacy roles and responsibilities throughout the system development life cycle;

c. Identify individuals having information security and privacy roles and responsibilities; and

d. Integrate the organizational information security and privacy risk management process into system development life cycle activities.

Discussion: A system development life cycle process provides the foundation for the successful development, implementation, and operation of organizational systems. The integration of security and privacy considerations early in the system development life cycle is a foundational principle of systems security engineering and privacy engineering. To apply the required controls within the system development life cycle requires a basic understanding of information security and privacy, threats, vulnerabilities, adverse impacts, and risk to critical mission and business functions. The security engineering principles in SA-8 help individuals properly design, code, and test systems and system components. Organizations include qualified personnel (e.g., senior agency information security officers, senior agency officials for privacy, security and privacy architects, and security and privacy engineers) in system development life cycle processes to ensure that established security and privacy requirements are incorporated into organizational systems. Role-based security and privacy training programs can ensure that individuals with key security and privacy roles and responsibilities have the experience, skills, and expertise to conduct assigned system development life cycle activities.

The effective integration of security and privacy requirements into enterprise architecture also helps to ensure that important security and privacy considerations are addressed throughout the system life cycle and that those considerations are directly related to organizational mission and business processes. This process also facilitates the integration of the information security and privacy architectures into the enterprise architecture, consistent with the risk management strategy of the organization. Because the system development life cycle involves multiple organizations, (e.g., external suppliers, developers, integrators, service providers), acquisition

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

and supply chain risk management functions and controls play significant roles in the effective management of the system during the life cycle.

Related Controls: AT-3, PL-8, PM-7, SA-4, SA-5, SA-8, SA-11, SA-15, SA-17, SA-22, SR-3, SR-4, SR-5, SR-9.

Control Enhancements:

(1) SYSTEM DEVELOPMENT LIFE CYCLE | MANAGE PREPRODUCTION ENVIRONMENT

Protect system preproduction environments commensurate with risk throughout the system development life cycle for the system, system component, or system service.

Discussion: The preproduction environment includes development, test, and integration environments. The program protection planning processes established by the Department of Defense are examples of managing the preproduction environment for defense contractors. Criticality analysis and the application of controls on developers also contribute to a more secure system development environment.

Related Controls: CM-2, CM-4, RA-3, RA-9, SA-4.

(2) SYSTEM DEVELOPMENT LIFE CYCLE | USE OF LIVE OR OPERATIONAL DATA

(a) Approve, document, and control the use of live data in preproduction environments for the system, system component, or system service; and

(b) Protect preproduction environments for the system, system component, or system service at the same impact or classification level as any live data in use within the preproduction environments.

Discussion: Live data is also referred to as operational data. The use of live or operational data in preproduction (i.e., development, test, and integration) environments can result in significant risks to organizations. In addition, the use of personally identifiable information in testing, research, and training increases the risk of unauthorized disclosure or misuse of such information. Therefore, it is important for the organization to manage any additional risks that may result from the use of live or operational data. Organizations can minimize such risks by using test or dummy data during the design, development, and testing of systems, system components, and system services. Risk assessment techniques may be used to determine if the risk of using live or operational data is acceptable.

Related Controls: PM-25, RA-3.

(3) SYSTEM DEVELOPMENT LIFE CYCLE | TECHNOLOGY REFRESH

Plan for and implement a technology refresh schedule for the system throughout the system development life cycle.

Discussion: Technology refresh planning may encompass hardware, software, firmware, processes, personnel skill sets, suppliers, service providers, and facilities. The use of obsolete or nearing obsolete technology may increase the security and privacy risks associated with unsupported components, counterfeit or repurposed components, components unable to implement security or privacy requirements, slow or inoperable components, components from untrusted sources, inadvertent personnel error, or increased complexity. Technology refreshes typically occur during the operations and maintenance stage of the system development life cycle.

Related Controls: MA-6.

References: [OMB A-130], [SP 800-30], [SP 800-37], [SP 800-160-1], [SP 800-171], [SP 800-172].

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

SA-4 ACQUISITION PROCESS

Control: Include the following requirements, descriptions, and criteria, explicitly or by reference, using [*Selection (one or more): standardized contract language;* [*Assignment: organization-defined contract language*]] in the acquisition contract for the system, system component, or system service:

a. Security and privacy functional requirements;

b. Strength of mechanism requirements;

c. Security and privacy assurance requirements;

d. Controls needed to satisfy the security and privacy requirements.

e. Security and privacy documentation requirements;

f. Requirements for protecting security and privacy documentation;

g. Description of the system development environment and environment in which the system is intended to operate;

h. Allocation of responsibility or identification of parties responsible for information security, privacy, and supply chain risk management; and

i. Acceptance criteria.

Discussion: Security and privacy functional requirements are typically derived from the high-level security and privacy requirements described in SA-2. The derived requirements include security and privacy capabilities, functions, and mechanisms. Strength requirements associated with such capabilities, functions, and mechanisms include degree of correctness, completeness, resistance to tampering or bypass, and resistance to direct attack. Assurance requirements include development processes, procedures, and methodologies as well as the evidence from development and assessment activities that provide grounds for confidence that the required functionality is implemented and possesses the required strength of mechanism. [SP 800-160-1] describes the process of requirements engineering as part of the system development life cycle.

Controls can be viewed as descriptions of the safeguards and protection capabilities appropriate for achieving the particular security and privacy objectives of the organization and for reflecting the security and privacy requirements of stakeholders. Controls are selected and implemented in order to satisfy system requirements and include developer and organizational responsibilities. Controls can include technical, administrative, and physical aspects. In some cases, the selection and implementation of a control may necessitate additional specification by the organization in the form of derived requirements or instantiated control parameter values. The derived requirements and control parameter values may be necessary to provide the appropriate level of implementation detail for controls within the system development life cycle.

Security and privacy documentation requirements address all stages of the system development life cycle. Documentation provides user and administrator guidance for the implementation and operation of controls. The level of detail required in such documentation is based on the security categorization or classification level of the system and the degree to which organizations depend on the capabilities, functions, or mechanisms to meet risk response expectations. Requirements can include mandated configuration settings that specify allowed functions, ports, protocols, and services. Acceptance criteria for systems, system components, and system services are defined in the same manner as the criteria for any organizational acquisition or procurement.

Related Controls: CM-6, CM-8, PS-7, SA-3, SA-5, SA-8, SA-11, SA-15, SA-16, SA-17, SA-21, SR-3, SR-5.

Control Enhancements:

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

(1) ACQUISITION PROCESS | FUNCTIONAL PROPERTIES OF CONTROLS

Require the developer of the system, system component, or system service to provide a description of the functional properties of the controls to be implemented.

Discussion: Functional properties of security and privacy controls describe the functionality (i.e., security or privacy capability, functions, or mechanisms) visible at the interfaces of the controls and specifically exclude functionality and data structures internal to the operation of the controls.

Related Controls: None.

(2) ACQUISITION PROCESS | DESIGN AND IMPLEMENTATION INFORMATION FOR CONTROLS

Require the developer of the system, system component, or system service to provide design and implementation information for the controls that includes: [*Selection (one or more): security-relevant external system interfaces; high-level design; low-level design; source code or hardware schematics;* [*Assignment: organization-defined design and implementation information*]] at [*Assignment: organization-defined level of detail*].

Discussion: Organizations may require different levels of detail in the documentation for the design and implementation of controls in organizational systems, system components, or system services based on mission and business requirements, requirements for resiliency and trustworthiness, and requirements for analysis and testing. Systems can be partitioned into multiple subsystems. Each subsystem within the system can contain one or more modules. The high-level design for the system is expressed in terms of subsystems and the interfaces between subsystems providing security-relevant functionality. The low-level design for the system is expressed in terms of modules and the interfaces between modules providing security-relevant functionality. Design and implementation documentation can include manufacturer, version, serial number, verification hash signature, software libraries used, date of purchase or download, and the vendor or download source. Source code and hardware schematics are referred to as the implementation representation of the system.

Related Controls: None.

(3) ACQUISITION PROCESS | DEVELOPMENT METHODS, TECHNIQUES, AND PRACTICES

Require the developer of the system, system component, or system service to demonstrate the use of a system development life cycle process that includes:

(a) **[*Assignment: organization-defined systems engineering methods*];**

(b) **[*Assignment: organization-defined* [*Selection (one or more): systems security; privacy*] *engineering methods*]; and**

(c) **[*Assignment: organization-defined software development methods; testing, evaluation, assessment, verification, and validation methods; and quality control processes*].**

Discussion: Following a system development life cycle that includes state-of-the-practice software development methods, systems engineering methods, systems security and privacy engineering methods, and quality control processes helps to reduce the number and severity of latent errors within systems, system components, and system services. Reducing the number and severity of such errors reduces the number of vulnerabilities in those systems, components, and services. Transparency in the methods and techniques that developers select and implement for systems engineering, systems security and privacy engineering, software development, component and system assessments, and quality control processes provides an increased level of assurance in the trustworthiness of the system, system component, or system service being acquired.

Related Controls: None.

(4) ACQUISITION PROCESS | ASSIGNMENT OF COMPONENTS TO SYSTEMS

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

[Withdrawn: Incorporated into CM-8(9).]

(5) ACQUISITION PROCESS | SYSTEM, COMPONENT, AND SERVICE CONFIGURATIONS

Require the developer of the system, system component, or system service to:

(a) Deliver the system, component, or service with [*Assignment: organization-defined security configurations*] implemented; and

(b) Use the configurations as the default for any subsequent system, component, or service reinstallation or upgrade.

Discussion: Examples of security configurations include the U.S. Government Configuration Baseline (USGCB), Security Technical Implementation Guides (STIGs), and any limitations on functions, ports, protocols, and services. Security characteristics can include requiring that default passwords have been changed.

Related Controls: None.

(6) ACQUISITION PROCESS | USE OF INFORMATION ASSURANCE PRODUCTS

(a) Employ only government off-the-shelf or commercial off-the-shelf information assurance and information assurance-enabled information technology products that compose an NSA-approved solution to protect classified information when the networks used to transmit the information are at a lower classification level than the information being transmitted; and

(b) Ensure that these products have been evaluated and/or validated by NSA or in accordance with NSA-approved procedures.

Discussion: Commercial off-the-shelf IA or IA-enabled information technology products used to protect classified information by cryptographic means may be required to use NSA-approved key management. See [NSA CSFC].

Related Controls: SC-8, SC-12, SC-13.

(7) ACQUISITION PROCESS | NIAP-APPROVED PROTECTION PROFILES

(a) Limit the use of commercially provided information assurance and information assurance-enabled information technology products to those products that have been successfully evaluated against a National Information Assurance partnership (NIAP)-approved Protection Profile for a specific technology type, if such a profile exists; and

(b) Require, if no NIAP-approved Protection Profile exists for a specific technology type but a commercially provided information technology product relies on cryptographic functionality to enforce its security policy, that the cryptographic module is FIPS-validated or NSA-approved.

Discussion: See [NIAP CCEVS] for additional information on NIAP. See [NIST CMVP] for additional information on FIPS-validated cryptographic modules.

Related Controls: IA-7, SC-12, SC-13.

(8) ACQUISITION PROCESS | CONTINUOUS MONITORING PLAN FOR CONTROLS

Require the developer of the system, system component, or system service to produce a plan for continuous monitoring of control effectiveness that is consistent with the continuous monitoring program of the organization.

Discussion: The objective of continuous monitoring plans is to determine if the planned, required, and deployed controls within the system, system component, or system service continue to be effective over time based on the inevitable changes that occur. Developer continuous monitoring plans include a sufficient level of detail such that the information can be incorporated into continuous monitoring programs implemented by organizations. Continuous monitoring plans can include the types of control assessment and monitoring

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

activities planned, frequency of control monitoring, and actions to be taken when controls fail or become ineffective.

Related Controls: CA-7.

(9) ACQUISITION PROCESS | FUNCTIONS, PORTS, PROTOCOLS, AND SERVICES IN USE

Require the developer of the system, system component, or system service to identify the functions, ports, protocols, and services intended for organizational use.

Discussion: The identification of functions, ports, protocols, and services early in the system development life cycle (e.g., during the initial requirements definition and design stages) allows organizations to influence the design of the system, system component, or system service. This early involvement in the system development life cycle helps organizations avoid or minimize the use of functions, ports, protocols, or services that pose unnecessarily high risks and understand the trade-offs involved in blocking specific ports, protocols, or services or requiring system service providers to do so. Early identification of functions, ports, protocols, and services avoids costly retrofitting of controls after the system, component, or system service has been implemented. SA-9 describes the requirements for external system services. Organizations identify which functions, ports, protocols, and services are provided from external sources.

Related Controls: CM-7, SA-9.

(10) ACQUISITION PROCESS | USE OF APPROVED PIV PRODUCTS

Employ only information technology products on the FIPS 201-approved products list for Personal Identity Verification (PIV) capability implemented within organizational systems.

Discussion: Products on the FIPS 201-approved products list meet NIST requirements for Personal Identity Verification (PIV) of Federal Employees and Contractors. PIV cards are used for multi-factor authentication in systems and organizations.

Related Controls: IA-2, IA-8, PM-9.

(11) ACQUISITION PROCESS | SYSTEM OF RECORDS

Include [*Assignment: organization-defined Privacy Act requirements*] in the acquisition contract for the operation of a system of records on behalf of an organization to accomplish an organizational mission or function.

Discussion: When, by contract, an organization provides for the operation of a system of records to accomplish an organizational mission or function, the organization, consistent with its authority, causes the requirements of the [PRIVACT] to be applied to the system of records.

Related Controls: PT-6.

(12) ACQUISITION PROCESS | DATA OWNERSHIP

(a) Include organizational data ownership requirements in the acquisition contract; and

(b) Require all data to be removed from the contractor's system and returned to the organization within [*Assignment: organization-defined time frame*].

Discussion: Contractors who operate a system that contains data owned by an organization initiating the contract have policies and procedures in place to remove the data from their systems and/or return the data in a time frame defined by the contract.

Related Controls: None.

References: [PRIVACT], [OMB A-130], [ISO 15408-1], [ISO 15408-2], [ISO 15408-3], [ISO 29148], [FIPS 140-3], [FIPS 201-2], [SP 800-35], [SP 800-37], [SP 800-70], [SP 800-73-4], [SP 800-137], [SP 800-160-1], [SP 800-161], [IR 7539], [IR 7622], [IR 7676], [IR 7870], [IR 8062], [NIAP CCEVS], [NSA CSFC].

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

SA-5 SYSTEM DOCUMENTATION

Control:

a. Obtain or develop administrator documentation for the system, system component, or system service that describes:

   1. Secure configuration, installation, and operation of the system, component, or service;

   2. Effective use and maintenance of security and privacy functions and mechanisms; and

   3. Known vulnerabilities regarding configuration and use of administrative or privileged functions;

b. Obtain or develop user documentation for the system, system component, or system service that describes:

   1. User-accessible security and privacy functions and mechanisms and how to effectively use those functions and mechanisms;

   2. Methods for user interaction, which enables individuals to use the system, component, or service in a more secure manner and protect individual privacy; and

   3. User responsibilities in maintaining the security of the system, component, or service and privacy of individuals;

c. Document attempts to obtain system, system component, or system service documentation when such documentation is either unavailable or nonexistent and take [*Assignment: organization-defined actions*] in response; and

d. Distribute documentation to [*Assignment: organization-defined personnel or roles*].

Discussion: System documentation helps personnel understand the implementation and operation of controls. Organizations consider establishing specific measures to determine the quality and completeness of the content provided. System documentation may be used to support the management of supply chain risk, incident response, and other functions. Personnel or roles that require documentation include system owners, system security officers, and system administrators. Attempts to obtain documentation include contacting manufacturers or suppliers and conducting web-based searches. The inability to obtain documentation may occur due to the age of the system or component or the lack of support from developers and contractors. When documentation cannot be obtained, organizations may need to recreate the documentation if it is essential to the implementation or operation of the controls. The protection provided for the documentation is commensurate with the security category or classification of the system. Documentation that addresses system vulnerabilities may require an increased level of protection. Secure operation of the system includes initially starting the system and resuming secure system operation after a lapse in system operation.

Related Controls: CM-4, CM-6, CM-7, CM-8, PL-2, PL-4, PL-8, PS-2, SA-3, SA-4, SA-8, SA-9, SA-10, SA-11, SA-15, SA-16, SA-17, SI-12, SR-3.

Control Enhancements:

(1) SYSTEM DOCUMENTATION | FUNCTIONAL PROPERTIES OF SECURITY CONTROLS

[Withdrawn: Incorporated into SA-4(1).]

(2) SYSTEM DOCUMENTATION | SECURITY-RELEVANT EXTERNAL SYSTEM INTERFACES

[Withdrawn: Incorporated into SA-4(2).]

(3) SYSTEM DOCUMENTATION | HIGH-LEVEL DESIGN

[Withdrawn: Incorporated into SA-4(2).]

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

(4) SYSTEM DOCUMENTATION | LOW-LEVEL DESIGN

[Withdrawn: Incorporated into SA-4(2).]

(5) SYSTEM DOCUMENTATION | SOURCE CODE

[Withdrawn: Incorporated into SA-4(2).]

References: [SP 800-160-1].

SA-6 SOFTWARE USAGE RESTRICTIONS

[Withdrawn: Incorporated into CM-10 and SI-7.]

SA-7 USER-INSTALLED SOFTWARE

[Withdrawn: Incorporated into CM-11 and SI-7.]

SA-8 SECURITY AND PRIVACY ENGINEERING PRINCIPLES

Control: Apply the following systems security and privacy engineering principles in the specification, design, development, implementation, and modification of the system and system components: [*Assignment: organization-defined systems security and privacy engineering principles*].

Discussion: Systems security and privacy engineering principles are closely related to and implemented throughout the system development life cycle (see SA-3). Organizations can apply systems security and privacy engineering principles to new systems under development or to systems undergoing upgrades. For existing systems, organizations apply systems security and privacy engineering principles to system upgrades and modifications to the extent feasible, given the current state of hardware, software, and firmware components within those systems.

The application of systems security and privacy engineering principles helps organizations develop trustworthy, secure, and resilient systems and reduces the susceptibility to disruptions, hazards, threats, and the creation of privacy problems for individuals. Examples of system security engineering principles include: developing layered protections; establishing security and privacy policies, architecture, and controls as the foundation for design and development; incorporating security and privacy requirements into the system development life cycle; delineating physical and logical security boundaries; ensuring that developers are trained on how to build secure software; tailoring controls to meet organizational needs; and performing threat modeling to identify use cases, threat agents, attack vectors and patterns, design patterns, and compensating controls needed to mitigate risk.

Organizations that apply systems security and privacy engineering concepts and principles can facilitate the development of trustworthy, secure systems, system components, and system services; reduce risk to acceptable levels; and make informed risk management decisions. System security engineering principles can also be used to protect against certain supply chain risks, including incorporating tamper-resistant hardware into a design.

Related Controls: PL-8, PM-7, RA-2, RA-3, RA-9, SA-3, SA-4, SA-15, SA-17, SA-20, SC-2, SC-3, SC-32, SC-39, SR-2, SR-3, SR-4, SR-5.

Control Enhancements:

(1) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | CLEAR ABSTRACTIONS

Implement the security design principle of clear abstractions.

Discussion: The principle of clear abstractions states that a system has simple, well-defined interfaces and functions that provide a consistent and intuitive view of the data and how the data is managed. The clarity, simplicity, necessity, and sufficiency of the system interfaces—

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

combined with a precise definition of their functional behavior—promotes ease of analysis, inspection, and testing as well as the correct and secure use of the system. The clarity of an abstraction is subjective. Examples that reflect the application of this principle include avoidance of redundant, unused interfaces; information hiding; and avoidance of semantic overloading of interfaces or their parameters. Information hiding (i.e., representation-independent programming), is a design discipline used to ensure that the internal representation of information in one system component is not visible to another system component invoking or calling the first component, such that the published abstraction is not influenced by how the data may be managed internally.

Related Controls: None.

(2) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | LEAST COMMON MECHANISM

Implement the security design principle of least common mechanism in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of least common mechanism states that the amount of mechanism common to more than one user and depended on by all users is minimized [POPEK74]. Mechanism minimization implies that different components of a system refrain from using the same mechanism to access a system resource. Every shared mechanism (especially a mechanism involving shared variables) represents a potential information path between users and is designed with care to ensure that it does not unintentionally compromise security [SALTZER75]. Implementing the principle of least common mechanism helps to reduce the adverse consequences of sharing the system state among different programs. A single program that corrupts a shared state (including shared variables) has the potential to corrupt other programs that are dependent on the state. The principle of least common mechanism also supports the principle of simplicity of design and addresses the issue of covert storage channels [LAMPSON73].

Related Controls: None.

(3) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | MODULARITY AND LAYERING

Implement the security design principles of modularity and layering in [*Assignment: organization-defined systems or system components*].

Discussion: The principles of modularity and layering are fundamental across system engineering disciplines. Modularity and layering derived from functional decomposition are effective in managing system complexity by making it possible to comprehend the structure of the system. Modular decomposition, or refinement in system design, is challenging and resists general statements of principle. Modularity serves to isolate functions and related data structures into well-defined logical units. Layering allows the relationships of these units to be better understood so that dependencies are clear and undesired complexity can be avoided. The security design principle of modularity extends functional modularity to include considerations based on trust, trustworthiness, privilege, and security policy. Security-informed modular decomposition includes the allocation of policies to systems in a network, separation of system applications into processes with distinct address spaces, allocation of system policies to layers, and separation of processes into subjects with distinct privileges based on hardware-supported privilege domains.

Related Controls: SC-2, SC-3.

(4) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | PARTIALLY ORDERED DEPENDENCIES

Implement the security design principle of partially ordered dependencies in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of partially ordered dependencies states that the synchronization, calling, and other dependencies in the system are partially ordered. A fundamental concept in system design is layering, whereby the system is organized into well-defined, functionally

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

related modules or components. The layers are linearly ordered with respect to inter-layer dependencies, such that higher layers are dependent on lower layers. While providing functionality to higher layers, some layers can be self-contained and not dependent on lower layers. While a partial ordering of all functions in a given system may not be possible, if circular dependencies are constrained to occur within layers, the inherent problems of circularity can be more easily managed. Partially ordered dependencies and system layering contribute significantly to the simplicity and coherency of the system design. Partially ordered dependencies also facilitate system testing and analysis.

Related Controls: None.

(5) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | EFFICIENTLY MEDIATED ACCESS

Implement the security design principle of efficiently mediated access in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of efficiently mediated access states that policy enforcement mechanisms utilize the least common mechanism available while satisfying stakeholder requirements within expressed constraints. The mediation of access to system resources (i.e., CPU, memory, devices, communication ports, services, infrastructure, data, and information) is often the predominant security function of secure systems. It also enables the realization of protections for the capability provided to stakeholders by the system. Mediation of resource access can result in performance bottlenecks if the system is not designed correctly. For example, by using hardware mechanisms, efficiently mediated access can be achieved. Once access to a low-level resource such as memory has been obtained, hardware protection mechanisms can ensure that out-of-bounds access does not occur.

Related Controls: AC-25.

(6) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | MINIMIZED SHARING

Implement the security design principle of minimized sharing in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of minimized sharing states that no computer resource is shared between system components (e.g., subjects, processes, functions) unless it is absolutely necessary to do so. Minimized sharing helps to simplify system design and implementation. In order to protect user-domain resources from arbitrary active entities, no resource is shared unless that sharing has been explicitly requested and granted. The need for resource sharing can be motivated by the design principle of least common mechanism in the case of internal entities or driven by stakeholder requirements. However, internal sharing is carefully designed to avoid performance and covert storage and timing channel problems. Sharing via common mechanism can increase the susceptibility of data and information to unauthorized access, disclosure, use, or modification and can adversely affect the inherent capability provided by the system. To minimize sharing induced by common mechanisms, such mechanisms can be designed to be reentrant or virtualized to preserve separation. Moreover, the use of global data to share information is carefully scrutinized. The lack of encapsulation may obfuscate relationships among the sharing entities.

Related Controls: SC-31.

(7) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | REDUCED COMPLEXITY

Implement the security design principle of reduced complexity in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of reduced complexity states that the system design is as simple and small as possible. A small and simple design is more understandable, more analyzable, and less prone to error. The reduced complexity principle applies to any aspect of a system, but it has particular importance for security due to the various analyses performed to obtain evidence about the emergent security property of the system. For such analyses to be

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

successful, a small and simple design is essential. Application of the principle of reduced complexity contributes to the ability of system developers to understand the correctness and completeness of system security functions. It also facilitates the identification of potential vulnerabilities. The corollary of reduced complexity states that the simplicity of the system is directly related to the number of vulnerabilities it will contain; that is, simpler systems contain fewer vulnerabilities. An benefit of reduced complexity is that it is easier to understand whether the intended security policy has been captured in the system design and that fewer vulnerabilities are likely to be introduced during engineering development. An additional benefit is that any such conclusion about correctness, completeness, and the existence of vulnerabilities can be reached with a higher degree of assurance in contrast to conclusions reached in situations where the system design is inherently more complex. Transitioning from older technologies to newer technologies (e.g., transitioning from IPv4 to IPv6) may require implementing the older and newer technologies simultaneously during the transition period. This may result in a temporary increase in system complexity during the transition.

Related Controls: None.

(8) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | SECURE EVOLVABILITY

Implement the security design principle of secure evolvability in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of secure evolvability states that a system is developed to facilitate the maintenance of its security properties when there are changes to the system's structure, interfaces, interconnections (i.e., system architecture), functionality, or configuration (i.e., security policy enforcement). Changes include a new, enhanced, or upgraded system capability; maintenance and sustainment activities; and reconfiguration. Although it is not possible to plan for every aspect of system evolution, system upgrades and changes can be anticipated by analyses of mission or business strategic direction, anticipated changes in the threat environment, and anticipated maintenance and sustainment needs. It is unrealistic to expect that complex systems remain secure in contexts not envisioned during development, whether such contexts are related to the operational environment or to usage. A system may be secure in some new contexts, but there is no guarantee that its emergent behavior will always be secure. It is easier to build trustworthiness into a system from the outset, and it follows that the sustainment of system trustworthiness requires planning for change as opposed to adapting in an ad hoc or non-methodical manner. The benefits of this principle include reduced vendor life cycle costs, reduced cost of ownership, improved system security, more effective management of security risk, and less risk uncertainty.

Related Controls: CM-3.

(9) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | TRUSTED COMPONENTS

Implement the security design principle of trusted components in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of trusted components states that a component is trustworthy to at least a level commensurate with the security dependencies it supports (i.e., how much it is trusted to perform its security functions by other components). This principle enables the composition of components such that trustworthiness is not inadvertently diminished and the trust is not consequently misplaced. Ultimately, this principle demands some metric by which the trust in a component and the trustworthiness of a component can be measured on the same abstract scale. The principle of trusted components is particularly relevant when considering systems and components in which there are complex chains of trust dependencies. A trust dependency is also referred to as a trust relationship and there may be chains of trust relationships.

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

The principle of trusted components also applies to a compound component that consists of subcomponents (e.g., a subsystem), which may have varying levels of trustworthiness. The conservative assumption is that the trustworthiness of a compound component is that of its least trustworthy subcomponent. It may be possible to provide a security engineering rationale that the trustworthiness of a particular compound component is greater than the conservative assumption. However, any such rationale reflects logical reasoning based on a clear statement of the trustworthiness objectives as well as relevant and credible evidence. The trustworthiness of a compound component is not the same as increased application of defense-in-depth layering within the component or a replication of components. Defense-in-depth techniques do not increase the trustworthiness of the whole above that of the least trustworthy component.

Related Controls: None.

(10) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | HIERARCHICAL TRUST

Implement the security design principle of hierarchical trust in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of hierarchical trust for components builds on the principle of trusted components and states that the security dependencies in a system will form a partial ordering if they preserve the principle of trusted components. The partial ordering provides the basis for trustworthiness reasoning or an assurance case (assurance argument) when composing a secure system from heterogeneously trustworthy components. To analyze a system composed of heterogeneously trustworthy components for its trustworthiness, it is essential to eliminate circular dependencies with regard to the trustworthiness. If a more trustworthy component located in a lower layer of the system were to depend on a less trustworthy component in a higher layer, this would, in effect, put the components in the same "less trustworthy" equivalence class per the principle of trusted components. Trust relationships, or chains of trust, can have various manifestations. For example, the root certificate of a certificate hierarchy is the most trusted node in the hierarchy, whereas the leaves in the hierarchy may be the least trustworthy nodes. Another example occurs in a layered high-assurance system where the security kernel (including the hardware base), which is located at the lowest layer of the system, is the most trustworthy component. The principle of hierarchical trust, however, does not prohibit the use of overly trustworthy components. There may be cases in a system of low trustworthiness where it is reasonable to employ a highly trustworthy component rather than one that is less trustworthy (e.g., due to availability or other cost-benefit driver). For such a case, any dependency of the highly trustworthy component upon a less trustworthy component does not degrade the trustworthiness of the resulting low-trust system.

Related Controls: None.

(11) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | INVERSE MODIFICATION THRESHOLD

Implement the security design principle of inverse modification threshold in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of inverse modification threshold builds on the principle of trusted components and the principle of hierarchical trust and states that the degree of protection provided to a component is commensurate with its trustworthiness. As the trust placed in a component increases, the protection against unauthorized modification of the component also increases to the same degree. Protection from unauthorized modification can come in the form of the component's own self-protection and innate trustworthiness, or it can come from the protections afforded to the component from other elements or attributes of the security architecture (to include protections in the environment of operation).

Related Controls: None.

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

(12) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | HIERARCHICAL PROTECTION

Implement the security design principle of hierarchical protection in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of hierarchical protection states that a component need not be protected from more trustworthy components. In the degenerate case of the most trusted component, it protects itself from all other components. For example, if an operating system kernel is deemed the most trustworthy component in a system, then it protects itself from all untrusted applications it supports, but the applications, conversely, do not need to protect themselves from the kernel. The trustworthiness of users is a consideration for applying the principle of hierarchical protection. A trusted system need not protect itself from an equally trustworthy user, reflecting use of untrusted systems in "system high" environments where users are highly trustworthy and where other protections are put in place to bound and protect the "system high" execution environment.

Related Controls: None.

(13) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | MINIMIZED SECURITY ELEMENTS

Implement the security design principle of minimized security elements in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of minimized security elements states that the system does not have extraneous trusted components. The principle of minimized security elements has two aspects: the overall cost of security analysis and the complexity of security analysis. Trusted components are generally costlier to construct and implement, owing to the increased rigor of development processes. Trusted components require greater security analysis to qualify their trustworthiness. Thus, to reduce the cost and decrease the complexity of the security analysis, a system contains as few trustworthy components as possible. The analysis of the interaction of trusted components with other components of the system is one of the most important aspects of system security verification. If the interactions between components are unnecessarily complex, the security of the system will also be more difficult to ascertain than one whose internal trust relationships are simple and elegantly constructed. In general, fewer trusted components result in fewer internal trust relationships and a simpler system.

Related Controls: None.

(14) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | LEAST PRIVILEGE

Implement the security design principle of least privilege in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of least privilege states that each system component is allocated sufficient privileges to accomplish its specified functions but no more. Applying the principle of least privilege limits the scope of the component's actions, which has two desirable effects: the security impact of a failure, corruption, or misuse of the component will have a minimized security impact, and the security analysis of the component will be simplified. Least privilege is a pervasive principle that is reflected in all aspects of the secure system design. Interfaces used to invoke component capability are available to only certain subsets of the user population, and component design supports a sufficiently fine granularity of privilege decomposition. For example, in the case of an audit mechanism, there may be an interface for the audit manager, who configures the audit settings; an interface for the audit operator, who ensures that audit data is safely collected and stored; and, finally, yet another interface for the audit reviewer, who only has need to view the audit data that has been collected but no need to perform operations on that data.

In addition to its manifestations at the system interface, least privilege can be used as a guiding principle for the internal structure of the system itself. One aspect of internal least privilege is to construct modules so that only the elements encapsulated by the module are

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

directly operated on by the functions within the module. Elements external to a module that may be affected by the module's operation are indirectly accessed through interaction (e.g., via a function call) with the module that contains those elements. Another aspect of internal least privilege is that the scope of a given module or component includes only those system elements that are necessary for its functionality and that the access modes for the elements (e.g., read, write) are minimal.

Related Controls: AC-6, CM-7.

(15) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | PREDICATE PERMISSION

Implement the security design principle of predicate permission in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of predicate permission states that system designers consider requiring multiple authorized entities to provide consent before a highly critical operation or access to highly sensitive data, information, or resources is allowed to proceed. [SALTZER75] originally named predicate permission the separation of privilege. It is also equivalent to separation of duty. The division of privilege among multiple parties decreases the likelihood of abuse and provides the safeguard that no single accident, deception, or breach of trust is sufficient to enable an unrecoverable action that can lead to significantly damaging effects. The design options for such a mechanism may require simultaneous action (e.g., the firing of a nuclear weapon requires two different authorized individuals to give the correct command within a small time window) or a sequence of operations where each successive action is enabled by some prior action, but no single individual is able to enable more than one action.

Related Controls: AC-5.

(16) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | SELF-RELIANT TRUSTWORTHINESS

Implement the security design principle of self-reliant trustworthiness in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of self-reliant trustworthiness states that systems minimize their reliance on other systems for their own trustworthiness. A system is trustworthy by default, and any connection to an external entity is used to supplement its function. If a system were required to maintain a connection with another external entity in order to maintain its trustworthiness, then that system would be vulnerable to malicious and non-malicious threats that could result in the loss or degradation of that connection. The benefit of the principle of self-reliant trustworthiness is that the isolation of a system will make it less vulnerable to attack. A corollary to this principle relates to the ability of the system (or system component) to operate in isolation and then resynchronize with other components when it is rejoined with them.

Related Controls: None.

(17) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | SECURE DISTRIBUTED COMPOSITION

Implement the security design principle of secure distributed composition in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of secure distributed composition states that the composition of distributed components that enforce the same system security policy result in a system that enforces that policy at least as well as the individual components do. Many of the design principles for secure systems deal with how components can or should interact. The need to create or enable a capability from the composition of distributed components can magnify the relevancy of these principles. In particular, the translation of security policy from a stand-alone to a distributed system or a system-of-systems can have unexpected or emergent results. Communication protocols and distributed data consistency mechanisms help to ensure consistent policy enforcement across a distributed system. To ensure a

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

system-wide level of assurance of correct policy enforcement, the security architecture of a distributed composite system is thoroughly analyzed.

Related Controls: None.

(18) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | TRUSTED COMMUNICATIONS CHANNELS

Implement the security design principle of trusted communications channels in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of trusted communication channels states that when composing a system where there is a potential threat to communications between components (i.e., the interconnections between components), each communication channel is trustworthy to a level commensurate with the security dependencies it supports (i.e., how much it is trusted by other components to perform its security functions). Trusted communication channels are achieved by a combination of restricting access to the communication channel (to ensure an acceptable match in the trustworthiness of the endpoints involved in the communication) and employing end-to-end protections for the data transmitted over the communication channel (to protect against interception and modification and to further increase the assurance of proper end-to-end communication).

Related Controls: SC-8, SC-12, SC-13.

(19) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | CONTINUOUS PROTECTION

Implement the security design principle of continuous protection in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of continuous protection states that components and data used to enforce the security policy have uninterrupted protection that is consistent with the security policy and the security architecture assumptions. No assurances that the system can provide the confidentiality, integrity, availability, and privacy protections for its design capability can be made if there are gaps in the protection. Any assurances about the ability to secure a delivered capability require that data and information are continuously protected. That is, there are no periods during which data and information are left unprotected while under control of the system (i.e., during the creation, storage, processing, or communication of the data and information, as well as during system initialization, execution, failure, interruption, and shutdown). Continuous protection requires adherence to the precepts of the reference monitor concept (i.e., every request is validated by the reference monitor; the reference monitor is able to protect itself from tampering; and sufficient assurance of the correctness and completeness of the mechanism can be ascertained from analysis and testing) and the principle of secure failure and recovery (i.e., preservation of a secure state during error, fault, failure, and successful attack; preservation of a secure state during recovery to normal, degraded, or alternative operational modes).

Continuous protection also applies to systems designed to operate in varying configurations, including those that deliver full operational capability and degraded-mode configurations that deliver partial operational capability. The continuous protection principle requires that changes to the system security policies be traceable to the operational need that drives the configuration and be verifiable (i.e., it is possible to verify that the proposed changes will not put the system into an insecure state). Insufficient traceability and verification may lead to inconsistent states or protection discontinuities due to the complex or undecidable nature of the problem. The use of pre-verified configuration definitions that reflect the new security policy enables analysis to determine that a transition from old to new policies is essentially atomic and that any residual effects from the old policy are guaranteed to not conflict with the new policy. The ability to demonstrate continuous protection is rooted in the clear articulation of life cycle protection needs as stakeholder security requirements.

Related Controls: AC-25.

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

(20) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | SECURE METADATA MANAGEMENT

Implement the security design principle of secure metadata management in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of secure metadata management states that metadata are "first class" objects with respect to security policy when the policy requires either complete protection of information or that the security subsystem be self-protecting. The principle of secure metadata management is driven by the recognition that a system, subsystem, or component cannot achieve self-protection unless it protects the data it relies on for correct execution. Data is generally not interpreted by the system that stores it. It may have semantic value (i.e., it comprises information) to users and programs that process the data. In contrast, metadata is information about data, such as a file name or the date when the file was created. Metadata is bound to the target data that it describes in a way that the system can interpret, but it need not be stored inside of or proximate to its target data. There may be metadata whose target is itself metadata (e.g., the classification level or impact level of a file name), including self-referential metadata.

The apparent secondary nature of metadata can lead to neglect of its legitimate need for protection, resulting in a violation of the security policy that includes the exfiltration of information. A particular concern associated with insufficient protections for metadata is associated with multilevel secure (MLS) systems. MLS systems mediate access by a subject to an object based on relative sensitivity levels. It follows that all subjects and objects in the scope of control of the MLS system are either directly labeled or indirectly attributed with sensitivity levels. The corollary of labeled metadata for MLS systems states that objects containing metadata are labeled. As with protection needs assessments for data, attention is given to ensure that the confidentiality and integrity protections are individually assessed, specified, and allocated to metadata, as would be done for mission, business, and system data.

Related Controls: None.

(21) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | SELF-ANALYSIS

Implement the security design principle of self-analysis in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of self-analysis states that a system component is able to assess its internal state and functionality to a limited extent at various stages of execution, and that this self-analysis capability is commensurate with the level of trustworthiness invested in the system. At the system level, self-analysis can be achieved through hierarchical assessments of trustworthiness established in a bottom-up fashion. In this approach, the lower-level components check for data integrity and correct functionality (to a limited extent) of higher-level components. For example, trusted boot sequences involve a trusted lower-level component that attests to the trustworthiness of the next higher-level components so that a transitive chain of trust can be established. At the root, a component attests to itself, which usually involves an axiomatic or environmentally enforced assumption about its integrity. Results of the self-analyses can be used to guard against externally induced errors, internal malfunction, or transient errors. By following this principle, some simple malfunctions or errors can be detected without allowing the effects of the error or malfunction to propagate outside of the component. Further, the self-test can be used to attest to the configuration of the component, detecting any potential conflicts in configuration with respect to the expected configuration.

Related Controls: CA-7.

(22) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | ACCOUNTABILITY AND TRACEABILITY

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

Implement the security design principle of accountability and traceability in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of accountability and traceability states that it is possible to trace security-relevant actions (i.e., subject-object interactions) to the entity on whose behalf the action is being taken. The principle of accountability and traceability requires a trustworthy infrastructure that can record details about actions that affect system security (e.g., an audit subsystem). To record the details about actions, the system is able to uniquely identify the entity on whose behalf the action is being carried out and also record the relevant sequence of actions that are carried out. The accountability policy also requires that audit trail itself be protected from unauthorized access and modification. The principle of least privilege assists in tracing the actions to particular entities, as it increases the granularity of accountability. Associating specific actions with system entities, and ultimately with users, and making the audit trail secure against unauthorized access and modifications provide non-repudiation because once an action is recorded, it is not possible to change the audit trail. Another important function that accountability and traceability serves is in the routine and forensic analysis of events associated with the violation of security policy. Analysis of audit logs may provide additional information that may be helpful in determining the path or component that allowed the violation of the security policy and the actions of individuals associated with the violation of the security policy.

Related Controls: AC-6, AU-2, AU-3, AU-6, AU-9, AU-10, AU-12, IA-2, IR-4.

(23) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | SECURE DEFAULTS

Implement the security design principle of secure defaults in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of secure defaults states that the default configuration of a system (including its constituent subsystems, components, and mechanisms) reflects a restrictive and conservative enforcement of security policy. The principle of secure defaults applies to the initial (i.e., default) configuration of a system as well as to the security engineering and design of access control and other security functions that follow a "deny unless explicitly authorized" strategy. The initial configuration aspect of this principle requires that any "as shipped" configuration of a system, subsystem, or system component does not aid in the violation of the security policy and can prevent the system from operating in the default configuration for those cases where the security policy itself requires configuration by the operational user.

Restrictive defaults mean that the system will operate "as-shipped" with adequate self-protection and be able to prevent security breaches before the intended security policy and system configuration is established. In cases where the protection provided by the "as-shipped" product is inadequate, stakeholders assess the risk of using it prior to establishing a secure initial state. Adherence to the principle of secure defaults guarantees that a system is established in a secure state upon successfully completing initialization. In situations where the system fails to complete initialization, either it will perform a requested operation using secure defaults or it will not perform the operation. Refer to the principles of continuous protection and secure failure and recovery that parallel this principle to provide the ability to detect and recover from failure.

The security engineering approach to this principle states that security mechanisms deny requests unless the request is found to be well-formed and consistent with the security policy. The insecure alternative is to allow a request unless it is shown to be inconsistent with the policy. In a large system, the conditions that are satisfied to grant a request that is denied by default are often far more compact and complete than those that would need to be checked in order to deny a request that is granted by default.

Related Controls: CM-2, CM-6, SA-4.

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

(24) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | SECURE FAILURE AND RECOVERY

Implement the security design principle of secure failure and recovery in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of secure failure and recovery states that neither a failure in a system function or mechanism nor any recovery action in response to failure leads to a violation of security policy. The principle of secure failure and recovery parallels the principle of continuous protection to ensure that a system is capable of detecting (within limits) actual and impending failure at any stage of its operation (i.e., initialization, normal operation, shutdown, and maintenance) and to take appropriate steps to ensure that security policies are not violated. In addition, when specified, the system is capable of recovering from impending or actual failure to resume normal, degraded, or alternative secure operations while ensuring that a secure state is maintained such that security policies are not violated.

Failure is a condition in which the behavior of a component deviates from its specified or expected behavior for an explicitly documented input. Once a failed security function is detected, the system may reconfigure itself to circumvent the failed component while maintaining security and provide all or part of the functionality of the original system, or it may completely shut itself down to prevent any further violation of security policies. For this to occur, the reconfiguration functions of the system are designed to ensure continuous enforcement of security policy during the various phases of reconfiguration.

Another technique that can be used to recover from failures is to perform a rollback to a secure state (which may be the initial state) and then either shutdown or replace the service or component that failed such that secure operations may resume. Failure of a component may or may not be detectable to the components using it. The principle of secure failure indicates that components fail in a state that denies rather than grants access. For example, a nominally "atomic" operation interrupted before completion does not violate security policy and is designed to handle interruption events by employing higher-level atomicity and rollback mechanisms (e.g., transactions). If a service is being used, its atomicity properties are well-documented and characterized so that the component availing itself of that service can detect and handle interruption events appropriately. For example, a system is designed to gracefully respond to disconnection and support resynchronization and data consistency after disconnection.

Failure protection strategies that employ replication of policy enforcement mechanisms, sometimes called defense in depth, can allow the system to continue in a secure state even when one mechanism has failed to protect the system. If the mechanisms are similar, however, the additional protection may be illusory, as the adversary can simply attack in series. Similarly, in a networked system, breaking the security on one system or service may enable an attacker to do the same on other similar replicated systems and services. By employing multiple protection mechanisms whose features are significantly different, the possibility of attack replication or repetition can be reduced. Analyses are conducted to weigh the costs and benefits of such redundancy techniques against increased resource usage and adverse effects on the overall system performance. Additional analyses are conducted as the complexity of these mechanisms increases, as could be the case for dynamic behaviors. Increased complexity generally reduces trustworthiness. When a resource cannot be continuously protected, it is critical to detect and repair any security breaches before the resource is once again used in a secure context.

Related Controls: CP-10, CP-12, SC-7, SC-8, SC-24, SI-13.

(25) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | ECONOMIC SECURITY

Implement the security design principle of economic security in [*Assignment: organization-defined systems or system components*].

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

Discussion: The principle of economic security states that security mechanisms are not costlier than the potential damage that could occur from a security breach. This is the security-relevant form of the cost-benefit analyses used in risk management. The cost assumptions of cost-benefit analysis prevent the system designer from incorporating security mechanisms of greater strength than necessary, where strength of mechanism is proportional to cost. The principle of economic security also requires analysis of the benefits of assurance relative to the cost of that assurance in terms of the effort expended to obtain relevant and credible evidence as well as the necessary analyses to assess and draw trustworthiness and risk conclusions from the evidence.

Related Controls: RA-3.

(26) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | PERFORMANCE SECURITY

Implement the security design principle of performance security in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of performance security states that security mechanisms are constructed so that they do not degrade system performance unnecessarily. Stakeholder and system design requirements for performance and security are precisely articulated and prioritized. For the system implementation to meet its design requirements and be found acceptable to stakeholders (i.e., validation against stakeholder requirements), the designers adhere to the specified constraints that capability performance needs place on protection needs. The overall impact of computationally intensive security services (e.g., cryptography) are assessed and demonstrated to pose no significant impact to higher-priority performance considerations or are deemed to provide an acceptable trade-off of performance for trustworthy protection. The trade-off considerations include less computationally intensive security services unless they are unavailable or insufficient. The insufficiency of a security service is determined by functional capability and strength of mechanism. The strength of mechanism is selected with respect to security requirements, performance-critical overhead issues (e.g., cryptographic key management), and an assessment of the capability of the threat.

The principle of performance security leads to the incorporation of features that help in the enforcement of security policy but incur minimum overhead, such as low-level hardware mechanisms upon which higher-level services can be built. Such low-level mechanisms are usually very specific, have very limited functionality, and are optimized for performance. For example, once access rights to a portion of memory is granted, many systems use hardware mechanisms to ensure that all further accesses involve the correct memory address and access mode. Application of this principle reinforces the need to design security into the system from the ground up and to incorporate simple mechanisms at the lower layers that can be used as building blocks for higher-level mechanisms.

Related Controls: SC-12, SC-13, SI-2, SI-7.

(27) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | HUMAN FACTORED SECURITY

Implement the security design principle of human factored security in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of human factored security states that the user interface for security functions and supporting services is intuitive, user-friendly, and provides feedback for user actions that affect such policy and its enforcement. The mechanisms that enforce security policy are not intrusive to the user and are designed not to degrade user efficiency. Security policy enforcement mechanisms also provide the user with meaningful, clear, and relevant feedback and warnings when insecure choices are being made. Particular attention is given to interfaces through which personnel responsible for system administration and operation configure and set up the security policies. Ideally, these personnel are able to

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

understand the impact of their choices. Personnel with system administrative and operational responsibilities are able to configure systems before start-up and administer them during runtime with confidence that their intent is correctly mapped to the system's mechanisms. Security services, functions, and mechanisms do not impede or unnecessarily complicate the intended use of the system. There is a trade-off between system usability and the strictness necessary for security policy enforcement. If security mechanisms are frustrating or difficult to use, then users may disable them, avoid them, or use them in ways inconsistent with the security requirements and protection needs that the mechanisms were designed to satisfy.

Related Controls: None.

(28) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | ACCEPTABLE SECURITY

Implement the security design principle of acceptable security in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of acceptable security requires that the level of privacy and performance that the system provides is consistent with the users' expectations. The perception of personal privacy may affect user behavior, morale, and effectiveness. Based on the organizational privacy policy and the system design, users should be able to restrict their actions to protect their privacy. When systems fail to provide intuitive interfaces or meet privacy and performance expectations, users may either choose to completely avoid the system or use it in ways that may be inefficient or even insecure.

Related Controls: None.

(29) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | REPEATABLE AND DOCUMENTED PROCEDURES

Implement the security design principle of repeatable and documented procedures in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of repeatable and documented procedures states that the techniques and methods employed to construct a system component permit the same component to be completely and correctly reconstructed at a later time. Repeatable and documented procedures support the development of a component that is identical to the component created earlier, which may be in widespread use. In the case of other system artifacts (e.g., documentation and testing results), repeatability supports consistency and the ability to inspect the artifacts. Repeatable and documented procedures can be introduced at various stages within the system development life cycle and contribute to the ability to evaluate assurance claims for the system. Examples include systematic procedures for code development and review, procedures for the configuration management of development tools and system artifacts, and procedures for system delivery.

Related Controls: CM-1, SA-1, SA-10, SA-11, SA-15, SA-17, SC-1, SI-1.

(30) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | PROCEDURAL RIGOR

Implement the security design principle of procedural rigor in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of procedural rigor states that the rigor of a system life cycle process is commensurate with its intended trustworthiness. Procedural rigor defines the scope, depth, and detail of the system life cycle procedures. Rigorous system life cycle procedures contribute to the assurance that the system is correct and free of unintended functionality in several ways. First, the procedures impose checks and balances on the life cycle process such that the introduction of unspecified functionality is prevented.

Second, rigorous procedures applied to systems security engineering activities that produce specifications and other system design documents contribute to the ability to understand

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

the system as it has been built rather than trusting that the component, as implemented, is the authoritative (and potentially misleading) specification.

Finally, modifications to an existing system component are easier when there are detailed specifications that describe its current design instead of studying source code or schematics to try to understand how it works. Procedural rigor helps ensure that security functional and assurance requirements have been satisfied, and it contributes to a better-informed basis for the determination of trustworthiness and risk posture. Procedural rigor is commensurate with the degree of assurance desired for the system. If the required trustworthiness of the system is low, a high level of procedural rigor may add unnecessary cost, whereas when high trustworthiness is critical, the cost of high procedural rigor is merited.

Related Controls: None.

(31) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | SECURE SYSTEM MODIFICATION

Implement the security design principle of secure system modification in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of secure system modification states that system modification maintains system security with respect to the security requirements and risk tolerance of stakeholders. Upgrades or modifications to systems can transform secure systems into systems that are not secure. The procedures for system modification ensure that if the system is to maintain its trustworthiness, the same rigor that was applied to its initial development is applied to any system changes. Because modifications can affect the ability of the system to maintain its secure state, a careful security analysis of the modification is needed prior to its implementation and deployment. This principle parallels the principle of secure evolvability.

Related Controls: CM-3, CM-4.

(32) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | SUFFICIENT DOCUMENTATION

Implement the security design principle of sufficient documentation in [*Assignment: organization-defined systems or system components*].

Discussion: The principle of sufficient documentation states that organizational personnel with responsibilities to interact with the system are provided with adequate documentation and other information such that the personnel contribute to rather than detract from system security. Despite attempts to comply with principles such as human factored security and acceptable security, systems are inherently complex, and the design intent for the use of security mechanisms and the ramifications of the misuse or misconfiguration of security mechanisms are not always intuitively obvious. Uninformed and insufficiently trained users can introduce vulnerabilities due to errors of omission and commission. The availability of documentation and training can help to ensure a knowledgeable cadre of personnel, all of whom have a critical role in the achievement of principles such as continuous protection. Documentation is written clearly and supported by training that provides security awareness and understanding of security-relevant responsibilities.

Related Controls: AT-2, AT-3, SA-5.

(33) SECURITY AND PRIVACY ENGINEERING PRINCIPLES | MINIMIZATION

Implement the privacy principle of minimization using [*Assignment: organization-defined processes*].

Discussion: The principle of minimization states that organizations should only process personally identifiable information that is directly relevant and necessary to accomplish an authorized purpose and should only maintain personally identifiable information for as long as is necessary to accomplish the purpose. Organizations have processes in place, consistent with applicable laws and policies, to implement the principle of minimization.

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

Related Controls: PE-8, PM-25, SC-42, SI-12.

References: [PRIVACT], [OMB A-130], [FIPS 199], [FIPS 200], [SP 800-37], [SP 800-53A], [SP 800-60-1], [SP 800-60-2], [SP 800-160-1], [IR 8062].

SA-9 EXTERNAL SYSTEM SERVICES

Control:

a. Require that providers of external system services comply with organizational security and privacy requirements and employ the following controls: [*Assignment: organization-defined controls*];

b. Define and document organizational oversight and user roles and responsibilities with regard to external system services; and

c. Employ the following processes, methods, and techniques to monitor control compliance by external service providers on an ongoing basis: [*Assignment: organization-defined processes, methods, and techniques*].

Discussion: External system services are provided by an external provider, and the organization has no direct control over the implementation of the required controls or the assessment of control effectiveness. Organizations establish relationships with external service providers in a variety of ways, including through business partnerships, contracts, interagency agreements, lines of business arrangements, licensing agreements, joint ventures, and supply chain exchanges. The responsibility for managing risks from the use of external system services remains with authorizing officials. For services external to organizations, a chain of trust requires that organizations establish and retain a certain level of confidence that each provider in the consumer-provider relationship provides adequate protection for the services rendered. The extent and nature of this chain of trust vary based on relationships between organizations and the external providers. Organizations document the basis for the trust relationships so that the relationships can be monitored. External system services documentation includes government, service providers, end user security roles and responsibilities, and service-level agreements. Service-level agreements define the expectations of performance for implemented controls, describe measurable outcomes, and identify remedies and response requirements for identified instances of noncompliance.

Related Controls: AC-20, CA-3, CP-2, IR-4, IR-7, PL-10, PL-11, PS-7, SA-2, SA-4, SR-3, SR-5.

Control Enhancements:

(1) EXTERNAL SYSTEM SERVICES | RISK ASSESSMENTS AND ORGANIZATIONAL APPROVALS

(a) Conduct an organizational assessment of risk prior to the acquisition or outsourcing of information security services; and

(b) Verify that the acquisition or outsourcing of dedicated information security services is approved by [*Assignment: organization-defined personnel or roles*].

Discussion: Information security services include the operation of security devices, such as firewalls or key management services as well as incident monitoring, analysis, and response. Risks assessed can include system, mission or business, security, privacy, or supply chain risks.

Related Controls: CA-6, RA-3, RA-8.

(2) EXTERNAL SYSTEM SERVICES | IDENTIFICATION OF FUNCTIONS, PORTS, PROTOCOLS, AND SERVICES

Require providers of the following external system services to identify the functions, ports, protocols, and other services required for the use of such services: [*Assignment: organization-defined external system services*].

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

Discussion: Information from external service providers regarding the specific functions, ports, protocols, and services used in the provision of such services can be useful when the need arises to understand the trade-offs involved in restricting certain functions and services or blocking certain ports and protocols.

Related Controls: CM-6, CM-7.

(3) EXTERNAL SYSTEM SERVICES | ESTABLISH AND MAINTAIN TRUST RELATIONSHIP WITH PROVIDERS

Establish, document, and maintain trust relationships with external service providers based on the following requirements, properties, factors, or conditions: [*Assignment: organization-defined security and privacy requirements, properties, factors, or conditions defining acceptable trust relationships*].

Discussion: Trust relationships between organizations and external service providers reflect the degree of confidence that the risk from using external services is at an acceptable level. Trust relationships can help organizations gain increased levels of confidence that service providers are providing adequate protection for the services rendered and can also be useful when conducting incident response or when planning for upgrades or obsolescence. Trust relationships can be complicated due to the potentially large number of entities participating in the consumer-provider interactions, subordinate relationships and levels of trust, and types of interactions between the parties. In some cases, the degree of trust is based on the level of control that organizations can exert on external service providers regarding the controls necessary for the protection of the service, information, or individual privacy and the evidence brought forth as to the effectiveness of the implemented controls. The level of control is established by the terms and conditions of the contracts or service-level agreements.

Related Controls: SR-2.

(4) EXTERNAL SYSTEM SERVICES | CONSISTENT INTERESTS OF CONSUMERS AND PROVIDERS

Take the following actions to verify that the interests of [*Assignment: organization-defined external service providers*] are consistent with and reflect organizational interests: [*Assignment: organization-defined actions*].

Discussion: As organizations increasingly use external service providers, it is possible that the interests of the service providers may diverge from organizational interests. In such situations, simply having the required technical, management, or operational controls in place may not be sufficient if the providers that implement and manage those controls are not operating in a manner consistent with the interests of the consuming organizations. Actions that organizations take to address such concerns include requiring background checks for selected service provider personnel; examining ownership records; employing only trustworthy service providers, such as providers with which organizations have had successful trust relationships; and conducting routine, periodic, unscheduled visits to service provider facilities.

Related Controls: None.

(5) EXTERNAL SYSTEM SERVICES | PROCESSING, STORAGE, AND SERVICE LOCATION

Restrict the location of [*Selection (one or more): information processing; information or data; system services*] to [*Assignment: organization-defined locations*] based on [*Assignment: organization-defined requirements or conditions*].

Discussion: The location of information processing, information and data storage, or system services can have a direct impact on the ability of organizations to successfully execute their mission and business functions. The impact occurs when external providers control the location of processing, storage, or services. The criteria that external providers use for the selection of processing, storage, or service locations may be different from the criteria that organizations use. For example, organizations may desire that data or information storage

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

locations be restricted to certain locations to help facilitate incident response activities in case of information security incidents or breaches. Incident response activities, including forensic analyses and after-the-fact investigations, may be adversely affected by the governing laws, policies, or protocols in the locations where processing and storage occur and/or the locations from which system services emanate.

Related Controls: SA-5, SR-4.

(6) EXTERNAL SYSTEM SERVICES | ORGANIZATION-CONTROLLED CRYPTOGRAPHIC KEYS

Maintain exclusive control of cryptographic keys for encrypted material stored or transmitted through an external system.

Discussion: Maintaining exclusive control of cryptographic keys in an external system prevents decryption of organizational data by external system staff. Organizational control of cryptographic keys can be implemented by encrypting and decrypting data inside the organization as data is sent to and received from the external system or by employing a component that permits encryption and decryption functions to be local to the external system but allows exclusive organizational access to the encryption keys.

Related Controls: SC-12, SC-13, SI-4.

(7) EXTERNAL SYSTEM SERVICES | ORGANIZATION-CONTROLLED INTEGRITY CHECKING

Provide the capability to check the integrity of information while it resides in the external system.

Discussion: Storage of organizational information in an external system could limit visibility into the security status of its data. The ability of the organization to verify and validate the integrity of its stored data without transferring it out of the external system provides such visibility.

Related Controls: SI-7.

(8) EXTERNAL SYSTEM SERVICES | PROCESSING AND STORAGE LOCATION — U.S. JURISDICTION

Restrict the geographic location of information processing and data storage to facilities located within in the legal jurisdictional boundary of the United States.

Discussion: The geographic location of information processing and data storage can have a direct impact on the ability of organizations to successfully execute their mission and business functions. A compromise or breach of high impact information and systems can have severe or catastrophic adverse impacts on organizational assets and operations, individuals, other organizations, and the Nation. Restricting the processing and storage of high-impact information to facilities within the legal jurisdictional boundary of the United States provides greater control over such processing and storage.

Related Controls: SA-5, SR-4.

References: [OMB A-130], [SP 800-35], [SP 800-160-1], [SP 800-161], [SP 800-171].

SA-10 DEVELOPER CONFIGURATION MANAGEMENT

Control: Require the developer of the system, system component, or system service to:

a. Perform configuration management during system, component, or service [*Selection (one or more): design; development; implementation; operation; disposal*];

b. Document, manage, and control the integrity of changes to [*Assignment: organization-defined configuration items under configuration management*];

c. Implement only organization-approved changes to the system, component, or service;

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

d. Document approved changes to the system, component, or service and the potential security and privacy impacts of such changes; and

e. Track security flaws and flaw resolution within the system, component, or service and report findings to [*Assignment: organization-defined personnel*].

Discussion: Organizations consider the quality and completeness of configuration management activities conducted by developers as direct evidence of applying effective security controls. Controls include protecting the master copies of material used to generate security-relevant portions of the system hardware, software, and firmware from unauthorized modification or destruction. Maintaining the integrity of changes to the system, system component, or system service requires strict configuration control throughout the system development life cycle to track authorized changes and prevent unauthorized changes.

The configuration items that are placed under configuration management include the formal model; the functional, high-level, and low-level design specifications; other design data; implementation documentation; source code and hardware schematics; the current running version of the object code; tools for comparing new versions of security-relevant hardware descriptions and source code with previous versions; and test fixtures and documentation. Depending on the mission and business needs of organizations and the nature of the contractual relationships in place, developers may provide configuration management support during the operations and maintenance stage of the system development life cycle.

Related Controls: CM-2, CM-3, CM-4, CM-7, CM-9, SA-4, SA-5, SA-8, SA-15, SI-2, SR-3, SR-4, SR-5, SR-6.

Control Enhancements:

(1) DEVELOPER CONFIGURATION MANAGEMENT | SOFTWARE AND FIRMWARE INTEGRITY VERIFICATION

Require the developer of the system, system component, or system service to enable integrity verification of software and firmware components.

Discussion: Software and firmware integrity verification allows organizations to detect unauthorized changes to software and firmware components using developer-provided tools, techniques, and mechanisms. The integrity checking mechanisms can also address counterfeiting of software and firmware components. Organizations verify the integrity of software and firmware components, for example, through secure one-way hashes provided by developers. Delivered software and firmware components also include any updates to such components.

Related Controls: SI-7, SR-11.

(2) DEVELOPER CONFIGURATION MANAGEMENT | ALTERNATIVE CONFIGURATION MANAGEMENT PROCESSES

Provide an alternate configuration management process using organizational personnel in the absence of a dedicated developer configuration management team.

Discussion: Alternate configuration management processes may be required when organizations use commercial off-the-shelf information technology products. Alternate configuration management processes include organizational personnel who review and approve proposed changes to systems, system components, and system services and conduct security and privacy impact analyses prior to the implementation of changes to systems, components, or services.

Related Controls: None.

(3) DEVELOPER CONFIGURATION MANAGEMENT | HARDWARE INTEGRITY VERIFICATION

Require the developer of the system, system component, or system service to enable integrity verification of hardware components.

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

Discussion: Hardware integrity verification allows organizations to detect unauthorized changes to hardware components using developer-provided tools, techniques, methods, and mechanisms. Organizations may verify the integrity of hardware components with hard-to-copy labels, verifiable serial numbers provided by developers, and by requiring the use of anti-tamper technologies. Delivered hardware components also include hardware and firmware updates to such components.

Related Controls: SI-7.

(4) DEVELOPER CONFIGURATION MANAGEMENT | TRUSTED GENERATION

Require the developer of the system, system component, or system service to employ tools for comparing newly generated versions of security-relevant hardware descriptions, source code, and object code with previous versions.

Discussion: The trusted generation of descriptions, source code, and object code addresses authorized changes to hardware, software, and firmware components between versions during development. The focus is on the efficacy of the configuration management process by the developer to ensure that newly generated versions of security-relevant hardware descriptions, source code, and object code continue to enforce the security policy for the system, system component, or system service. In contrast, SA-10(1) and SA-10(3) allow organizations to detect unauthorized changes to hardware, software, and firmware components using tools, techniques, or mechanisms provided by developers.

Related Controls: None.

(5) DEVELOPER CONFIGURATION MANAGEMENT | MAPPING INTEGRITY FOR VERSION CONTROL

Require the developer of the system, system component, or system service to maintain the integrity of the mapping between the master build data describing the current version of security-relevant hardware, software, and firmware and the on-site master copy of the data for the current version.

Discussion: Mapping integrity for version control addresses changes to hardware, software, and firmware components during both initial development and system development life cycle updates. Maintaining the integrity between the master copies of security-relevant hardware, software, and firmware (including designs, hardware drawings, source code) and the equivalent data in master copies in operational environments is essential to ensuring the availability of organizational systems that support critical mission and business functions.

Related Controls: None.

(6) DEVELOPER CONFIGURATION MANAGEMENT | TRUSTED DISTRIBUTION

Require the developer of the system, system component, or system service to execute procedures for ensuring that security-relevant hardware, software, and firmware updates distributed to the organization are exactly as specified by the master copies.

Discussion: The trusted distribution of security-relevant hardware, software, and firmware updates help to ensure that the updates are correct representations of the master copies maintained by the developer and have not been tampered with during distribution.

Related Controls: None.

(7) DEVELOPER CONFIGURATION MANAGEMENT | SECURITY AND PRIVACY REPRESENTATIVES

Require [*Assignment: organization-defined* security and privacy representatives] to be included in the [*Assignment: organization-defined configuration change management and control process*].

Discussion: Information security and privacy representatives can include system security officers, senior agency information security officers, senior agency officials for privacy, and system privacy officers. Representation by personnel with information security and privacy

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

expertise is important because changes to system configurations can have unintended side effects, some of which may be security- or privacy-relevant. Detecting such changes early in the process can help avoid unintended, negative consequences that could ultimately affect the security and privacy posture of systems. The configuration change management and control process in this control enhancement refers to the change management and control process defined by organizations in SA-10b.

Related Controls: None.

References: [FIPS 140-3], [FIPS 180-4], [FIPS 202], [SP 800-128], [SP 800-160-1].

SA-11 DEVELOPER TESTING AND EVALUATION

Control: Require the developer of the system, system component, or system service, at all post-design stages of the system development life cycle, to:

a. Develop and implement a plan for ongoing security and privacy control assessments;

b. Perform [*Selection (one or more): unit; integration; system; regression*] testing/evaluation [*Assignment: organization-defined frequency*] at [*Assignment: organization-defined depth and coverage*];

c. Produce evidence of the execution of the assessment plan and the results of the testing and evaluation;

d. Implement a verifiable flaw remediation process; and

e. Correct flaws identified during testing and evaluation.

Discussion: Developmental testing and evaluation confirms that the required controls are implemented correctly, operating as intended, enforcing the desired security and privacy policies, and meeting established security and privacy requirements. Security properties of systems and the privacy of individuals may be affected by the interconnection of system components or changes to those components. The interconnections or changes—including upgrading or replacing applications, operating systems, and firmware—may adversely affect previously implemented controls. Ongoing assessment during development allows for additional types of testing and evaluation that developers can conduct to reduce or eliminate potential flaws. Testing custom software applications may require approaches such as manual code review, security architecture review, and penetration testing, as well as and static analysis, dynamic analysis, binary analysis, or a hybrid of the three analysis approaches.

Developers can use the analysis approaches, along with security instrumentation and fuzzing, in a variety of tools and in source code reviews. The security and privacy assessment plans include the specific activities that developers plan to carry out, including the types of analyses, testing, evaluation, and reviews of software and firmware components; the degree of rigor to be applied; the frequency of the ongoing testing and evaluation; and the types of artifacts produced during those processes. The depth of testing and evaluation refers to the rigor and level of detail associated with the assessment process. The coverage of testing and evaluation refers to the scope (i.e., number and type) of the artifacts included in the assessment process. Contracts specify the acceptance criteria for security and privacy assessment plans, flaw remediation processes, and the evidence that the plans and processes have been diligently applied. Methods for reviewing and protecting assessment plans, evidence, and documentation are commensurate with the security category or classification level of the system. Contracts may specify protection requirements for documentation.

Related Controls: CA-2, CA-7, CM-4, SA-3, SA-4, SA-5, SA-8, SA-15, SA-17, SI-2, SR-5, SR-6, SR-7.

Control Enhancements:

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

(1) DEVELOPER TESTING AND EVALUATION | STATIC CODE ANALYSIS

Require the developer of the system, system component, or system service to employ static code analysis tools to identify common flaws and document the results of the analysis.

Discussion: Static code analysis provides a technology and methodology for security reviews and includes checking for weaknesses in the code as well as for the incorporation of libraries or other included code with known vulnerabilities or that are out-of-date and not supported. Static code analysis can be used to identify vulnerabilities and enforce secure coding practices. It is most effective when used early in the development process, when each code change can automatically be scanned for potential weaknesses. Static code analysis can provide clear remediation guidance and identify defects for developers to fix. Evidence of the correct implementation of static analysis can include aggregate defect density for critical defect types, evidence that defects were inspected by developers or security professionals, and evidence that defects were remediated. A high density of ignored findings, commonly referred to as false positives, indicates a potential problem with the analysis process or the analysis tool. In such cases, organizations weigh the validity of the evidence against evidence from other sources.

Related Controls: None.

(2) DEVELOPER TESTING AND EVALUATION | THREAT MODELING AND VULNERABILITY ANALYSES

Require the developer of the system, system component, or system service to perform threat modeling and vulnerability analyses during development and the subsequent testing and evaluation of the system, component, or service that:

(a) Uses the following contextual information: [*Assignment: organization-defined information concerning impact, environment of operations, known or assumed threats, and acceptable risk levels*];

(b) Employs the following tools and methods: [*Assignment: organization-defined tools and methods*];

(c) Conducts the modeling and analyses at the following level of rigor: [*Assignment: organization-defined breadth and depth of modeling and analyses*]; and

(d) Produces evidence that meets the following acceptance criteria: [*Assignment: organization-defined acceptance criteria*].

Discussion: Systems, system components, and system services may deviate significantly from the functional and design specifications created during the requirements and design stages of the system development life cycle. Therefore, updates to threat modeling and vulnerability analyses of those systems, system components, and system services during development and prior to delivery are critical to the effective operation of those systems, components, and services. Threat modeling and vulnerability analyses at this stage of the system development life cycle ensure that design and implementation changes have been accounted for and that vulnerabilities created because of those changes have been reviewed and mitigated.

Related controls: PM-15, RA-3, RA-5.

(3) DEVELOPER TESTING AND EVALUATION | INDEPENDENT VERIFICATION OF ASSESSMENT PLANS AND EVIDENCE

(a) Require an independent agent satisfying [*Assignment: organization-defined independence criteria*] to verify the correct implementation of the developer security and privacy assessment plans and the evidence produced during testing and evaluation; and

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

(b) Verify that the independent agent is provided with sufficient information to complete the verification process or granted the authority to obtain such information.

Discussion: Independent agents have the qualifications—including the expertise, skills, training, certifications, and experience—to verify the correct implementation of developer security and privacy assessment plans.

Related Controls: AT-3, RA-5.

(4) DEVELOPER TESTING AND EVALUATION | MANUAL CODE REVIEWS

Require the developer of the system, system component, or system service to perform a manual code review of [*Assignment: organization-defined specific code*] using the following processes, procedures, and/or techniques: [*Assignment: organization-defined processes, procedures, and/or techniques*].

Discussion: Manual code reviews are usually reserved for the critical software and firmware components of systems. Manual code reviews are effective at identifying weaknesses that require knowledge of the application's requirements or context that, in most cases, is unavailable to automated analytic tools and techniques, such as static and dynamic analysis. The benefits of manual code review include the ability to verify access control matrices against application controls and review detailed aspects of cryptographic implementations and controls.

Related Controls: None.

(5) DEVELOPER TESTING AND EVALUATION | PENETRATION TESTING

Require the developer of the system, system component, or system service to perform penetration testing:

(a) At the following level of rigor: [*Assignment: organization-defined breadth and depth of testing*]; and

(b) Under the following constraints: [*Assignment: organization-defined constraints*].

Discussion: Penetration testing is an assessment methodology in which assessors, using all available information technology product or system documentation and working under specific constraints, attempt to circumvent the implemented security and privacy features of information technology products and systems. Useful information for assessors who conduct penetration testing includes product and system design specifications, source code, and administrator and operator manuals. Penetration testing can include white-box, gray-box, or black-box testing with analyses performed by skilled professionals who simulate adversary actions. The objective of penetration testing is to discover vulnerabilities in systems, system components, and services that result from implementation errors, configuration faults, or other operational weaknesses or deficiencies. Penetration tests can be performed in conjunction with automated and manual code reviews to provide a greater level of analysis than would ordinarily be possible. When user session information and other personally identifiable information is captured or recorded during penetration testing, such information is handled appropriately to protect privacy.

Related Controls: CA-8, PM-14, PM-25, PT-2, SA-3, SI-2, SI-6.

(6) DEVELOPER TESTING AND EVALUATION | ATTACK SURFACE REVIEWS

Require the developer of the system, system component, or system service to perform attack surface reviews.

Discussion: Attack surfaces of systems and system components are exposed areas that make those systems more vulnerable to attacks. Attack surfaces include any accessible areas where weaknesses or deficiencies in the hardware, software, and firmware components provide opportunities for adversaries to exploit vulnerabilities. Attack surface reviews ensure that developers analyze the design and implementation changes to systems and

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

mitigate attack vectors generated as a result of the changes. The correction of identified flaws includes deprecation of unsafe functions.

Related Controls: SA-15.

(7) DEVELOPER TESTING AND EVALUATION | VERIFY SCOPE OF TESTING AND EVALUATION

Require the developer of the system, system component, or system service to verify that the scope of testing and evaluation provides complete coverage of the required controls at the following level of rigor: [*Assignment: organization-defined breadth and depth of testing and evaluation*].

Discussion: Verifying that testing and evaluation provides complete coverage of required controls can be accomplished by a variety of analytic techniques ranging from informal to formal. Each of these techniques provides an increasing level of assurance that corresponds to the degree of formality of the analysis. Rigorously demonstrating control coverage at the highest levels of assurance can be achieved using formal modeling and analysis techniques, including correlation between control implementation and corresponding test cases.

Related Controls: SA-15.

(8) DEVELOPER TESTING AND EVALUATION | DYNAMIC CODE ANALYSIS

Require the developer of the system, system component, or system service to employ dynamic code analysis tools to identify common flaws and document the results of the analysis.

Discussion: Dynamic code analysis provides runtime verification of software programs using tools capable of monitoring programs for memory corruption, user privilege issues, and other potential security problems. Dynamic code analysis employs runtime tools to ensure that security functionality performs in the way it was designed. A type of dynamic analysis, known as fuzz testing, induces program failures by deliberately introducing malformed or random data into software programs. Fuzz testing strategies are derived from the intended use of applications and the functional and design specifications for the applications. To understand the scope of dynamic code analysis and the assurance provided, organizations may also consider conducting code coverage analysis (i.e., checking the degree to which the code has been tested using metrics such as percent of subroutines tested or percent of program statements called during execution of the test suite) and/or concordance analysis (i.e., checking for words that are out of place in software code, such as non-English language words or derogatory terms).

Related Controls: None.

(9) DEVELOPER TESTING AND EVALUATION | INTERACTIVE APPLICATION SECURITY TESTING

Require the developer of the system, system component, or system service to employ interactive application security testing tools to identify flaws and document the results.

Discussion: Interactive (also known as instrumentation-based) application security testing is a method of detecting vulnerabilities by observing applications as they run during testing. The use of instrumentation relies on direct measurements of the actual running applications and uses access to the code, user interaction, libraries, frameworks, backend connections, and configurations to directly measure control effectiveness. When combined with analysis techniques, interactive application security testing can identify a broad range of potential vulnerabilities and confirm control effectiveness. Instrumentation-based testing works in real time and can be used continuously throughout the system development life cycle.

Related Controls: None.

References: [ISO 15408-3], [SP 800-30], [SP 800-53A], [SP 800-154], [SP 800-160-1].

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

SA-12 SUPPLY CHAIN PROTECTION

[Withdrawn: Incorporated into SR Family.]

Control Enhancements:

(1) SUPPLY CHAIN PROTECTION | ACQUISITION STRATEGIES / TOOLS / METHODS
[Withdrawn: Moved to SR-5.]

(2) SUPPLY CHAIN PROTECTION | SUPPLIER REVIEWS
[Withdrawn: Moved to SR-6.]

(3) SUPPLY CHAIN PROTECTION | TRUSTED SHIPPING AND WAREHOUSING
[Withdrawn: Incorporated into SR-3.]

(4) SUPPLY CHAIN PROTECTION | DIVERSITY OF SUPPLIERS
[Withdrawn: Moved to SR-3(1).]

(5) SUPPLY CHAIN PROTECTION | LIMITATION OF HARM
[Withdrawn: Moved to SR-3(2).]

(6) SUPPLY CHAIN PROTECTION | MINIMIZING PROCUREMENT TIME
[Withdrawn: Incorporated into SR-5(1).]

(7) SUPPLY CHAIN PROTECTION | ASSESSMENTS PRIOR TO SELECTION / ACCEPTANCE / UPDATE
[Withdrawn: Moved to SR-5(2).]

(8) SUPPLY CHAIN PROTECTION | USE OF ALL-SOURCE INTELLIGENCE
[Withdrawn: Incorporated into RA-3(2).]

(9) SUPPLY CHAIN PROTECTION | OPERATIONS SECURITY
[Withdrawn: Moved to SR-7.]

(10) SUPPLY CHAIN PROTECTION | VALIDATE AS GENUINE AND NOT ALTERED
[Withdrawn: Moved to SR-4(3).]

(11) SUPPLY CHAIN PROTECTION | PENETRATION TESTING / ANALYSIS OF ELEMENTS, PROCESSES, AND ACTORS
[Withdrawn: Moved to SR-6(1).]

(12) SUPPLY CHAIN PROTECTION | INTER-ORGANIZATIONAL AGREEMENTS
[Withdrawn: Moved to SR-8.]

(13) SUPPLY CHAIN PROTECTION | CRITICAL INFORMATION SYSTEM COMPONENTS
[Withdrawn: Incorporated into MA-6 and RA-9.]

(14) SUPPLY CHAIN PROTECTION | IDENTITY AND TRACEABILITY
[Withdrawn: Moved to SR-4(1) and SR-4(2).]

(15) SUPPLY CHAIN PROTECTION | PROCESSES TO ADDRESS WEAKNESSES OR DEFICIENCIES
[Withdrawn: Incorporated into SR-3.]

SA-13 TRUSTWORTHINESS

[Withdrawn: Incorporated into SA-8.]

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

SA-14 CRITICALITY ANALYSIS

[Withdrawn: Incorporated into RA-9.]

Control Enhancements:

(1) CRITICALITY ANALYSIS | CRITICAL COMPONENTS WITH NO VIABLE ALTERNATIVE SOURCING

[Withdrawn: Incorporated into SA-20.]

SA-15 DEVELOPMENT PROCESS, STANDARDS, AND TOOLS

Control:

a. Require the developer of the system, system component, or system service to follow a documented development process that:

   1. Explicitly addresses security and privacy requirements;
   2. Identifies the standards and tools used in the development process;
   3. Documents the specific tool options and tool configurations used in the development process; and
   4. Documents, manages, and ensures the integrity of changes to the process and/or tools used in development; and

b. Review the development process, standards, tools, tool options, and tool configurations [*Assignment: organization-defined frequency*] to determine if the process, standards, tools, tool options and tool configurations selected and employed can satisfy the following security and privacy requirements: [*Assignment: organization-defined security and privacy requirements*].

Discussion: Development tools include programming languages and computer-aided design systems. Reviews of development processes include the use of maturity models to determine the potential effectiveness of such processes. Maintaining the integrity of changes to tools and processes facilitates effective supply chain risk assessment and mitigation. Such integrity requires configuration control throughout the system development life cycle to track authorized changes and prevent unauthorized changes.

Related Controls: MA-6, SA-3, SA-4, SA-8, SA-10, SA-11, SR-3, SR-4, SR-5, SR-6, SR-9.

Control Enhancements:

(1) DEVELOPMENT PROCESS, STANDARDS, AND TOOLS | QUALITY METRICS

Require the developer of the system, system component, or system service to:

(a) Define quality metrics at the beginning of the development process; and

(b) Provide evidence of meeting the quality metrics [*Selection (one or more):* [*Assignment: organization-defined frequency*]*;* [*Assignment: organization-defined program review milestones*]*; upon delivery*].

Discussion: Organizations use quality metrics to establish acceptable levels of system quality. Metrics can include quality gates, which are collections of completion criteria or sufficiency standards that represent the satisfactory execution of specific phases of the system development project. For example, a quality gate may require the elimination of all compiler warnings or a determination that such warnings have no impact on the effectiveness of required security or privacy capabilities. During the execution phases of development projects, quality gates provide clear, unambiguous indications of progress. Other metrics apply to the entire development project. Metrics can include defining the severity thresholds of vulnerabilities in accordance with organizational risk tolerance, such

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

as requiring no known vulnerabilities in the delivered system with a Common Vulnerability Scoring System (CVSS) severity of medium or high.

Related Controls: None.

(2) DEVELOPMENT PROCESS, STANDARDS, AND TOOLS | SECURITY AND PRIVACY TRACKING TOOLS

Require the developer of the system, system component, or system service to select and employ security and privacy tracking tools for use during the development process.

Discussion: System development teams select and deploy security and privacy tracking tools, including vulnerability or work item tracking systems that facilitate assignment, sorting, filtering, and tracking of completed work items or tasks associated with development processes.

Related Controls: SA-11.

(3) DEVELOPMENT PROCESS, STANDARDS, AND TOOLS | CRITICALITY ANALYSIS

Require the developer of the system, system component, or system service to perform a criticality analysis:

(a) At the following decision points in the system development life cycle: [*Assignment: organization-defined decision points in the system development life cycle*]; and

(b) At the following level of rigor: [*Assignment: organization-defined breadth and depth of criticality analysis*].

Discussion: Criticality analysis performed by the developer provides input to the criticality analysis performed by organizations. Developer input is essential to organizational criticality analysis because organizations may not have access to detailed design documentation for system components that are developed as commercial off-the-shelf products. Such design documentation includes functional specifications, high-level designs, low-level designs, source code, and hardware schematics. Criticality analysis is important for organizational systems that are designated as high value assets. High value assets can be moderate- or high-impact systems due to heightened adversarial interest or potential adverse effects on the federal enterprise. Developer input is especially important when organizations conduct supply chain criticality analyses.

Related Controls: RA-9.

(4) DEVELOPMENT PROCESS, STANDARDS, AND TOOLS | THREAT MODELING AND VULNERABILITY ANALYSIS

[Withdrawn: Incorporated into SA-11(2).]

(5) DEVELOPMENT PROCESS, STANDARDS, AND TOOLS | ATTACK SURFACE REDUCTION

Require the developer of the system, system component, or system service to reduce attack surfaces to [*Assignment: organization-defined thresholds*].

Discussion: Attack surface reduction is closely aligned with threat and vulnerability analyses and system architecture and design. Attack surface reduction is a means of reducing risk to organizations by giving attackers less opportunity to exploit weaknesses or deficiencies (i.e., potential vulnerabilities) within systems, system components, and system services. Attack surface reduction includes implementing the concept of layered defenses, applying the principles of least privilege and least functionality, applying secure software development practices, deprecating unsafe functions, reducing entry points available to unauthorized users, reducing the amount of code that executes, and eliminating application programming interfaces (APIs) that are vulnerable to attacks.

Related Controls: AC-6, CM-7, RA-3, SA-11.

(6) DEVELOPMENT PROCESS, STANDARDS, AND TOOLS | CONTINUOUS IMPROVEMENT

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

Require the developer of the system, system component, or system service to implement an explicit process to continuously improve the development process.

Discussion: Developers of systems, system components, and system services consider the effectiveness and efficiency of their development processes for meeting quality objectives and addressing the security and privacy capabilities in current threat environments.

Related Controls: None.

(7) DEVELOPMENT PROCESS, STANDARDS, AND TOOLS | AUTOMATED VULNERABILITY ANALYSIS

Require the developer of the system, system component, or system service [*Assignment: organization-defined frequency*] to:

(a) Perform an automated vulnerability analysis using [*Assignment: organization-defined tools*];

(b) Determine the exploitation potential for discovered vulnerabilities;

(c) Determine potential risk mitigations for delivered vulnerabilities; and

(d) Deliver the outputs of the tools and results of the analysis to [*Assignment: organization-defined personnel or roles*].

Discussion: Automated tools can be more effective at analyzing exploitable weaknesses or deficiencies in large and complex systems, prioritizing vulnerabilities by severity, and providing recommendations for risk mitigations.

Related Controls: RA-5, SA-11.

(8) DEVELOPMENT PROCESS, STANDARDS, AND TOOLS | REUSE OF THREAT AND VULNERABILITY INFORMATION

Require the developer of the system, system component, or system service to use threat modeling and vulnerability analyses from similar systems, components, or services to inform the current development process.

Discussion: Analysis of vulnerabilities found in similar software applications can inform potential design and implementation issues for systems under development. Similar systems or system components may exist within developer organizations. Vulnerability information is available from a variety of public and private sector sources, including the NIST National Vulnerability Database.

Related Controls: None.

(9) DEVELOPMENT PROCESS, STANDARDS, AND TOOLS | USE OF LIVE DATA

[Withdrawn: Incorporated into SA-3(2).]

(10) DEVELOPMENT PROCESS, STANDARDS, AND TOOLS | INCIDENT RESPONSE PLAN

Require the developer of the system, system component, or system service to provide, implement, and test an incident response plan.

Discussion: The incident response plan provided by developers may provide information not readily available to organizations and be incorporated into organizational incident response plans. Developer information may also be extremely helpful, such as when organizations respond to vulnerabilities in commercial off-the-shelf products.

Related Controls: IR-8.

(11) DEVELOPMENT PROCESS, STANDARDS, AND TOOLS | ARCHIVE SYSTEM OR COMPONENT

Require the developer of the system or system component to archive the system or component to be released or delivered together with the corresponding evidence supporting the final security and privacy review.

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

Discussion: Archiving system or system components requires the developer to retain key development artifacts, including hardware specifications, source code, object code, and relevant documentation from the development process that can provide a readily available configuration baseline for system and component upgrades or modifications.

Related Controls: CM-2.

(12) DEVELOPMENT PROCESS, STANDARDS, AND TOOLS | MINIMIZE PERSONALLY IDENTIFIABLE INFORMATION

Require the developer of the system or system component to minimize the use of personally identifiable information in development and test environments.

Discussion: Organizations can minimize the risk to an individual's privacy by using techniques such as de-identification or synthetic data. Limiting the use of personally identifiable information in development and test environments helps reduce the level of privacy risk created by a system.

Related Controls: PM-25, SA-3, SA-8.

References: [SP 800-160-1], [IR 8179].

SA-16 DEVELOPER-PROVIDED TRAINING

Control: Require the developer of the system, system component, or system service to provide the following training on the correct use and operation of the implemented security and privacy functions, controls, and/or mechanisms: [*Assignment: organization-defined training*].

Discussion: Developer-provided training applies to external and internal (in-house) developers. Training personnel is essential to ensuring the effectiveness of the controls implemented within organizational systems. Types of training include web-based and computer-based training, classroom-style training, and hands-on training (including micro-training). Organizations can also request training materials from developers to conduct in-house training or offer self-training to organizational personnel. Organizations determine the type of training necessary and may require different types of training for different security and privacy functions, controls, and mechanisms.

Related Controls: AT-2, AT-3, PE-3, SA-4, SA-5.

Control Enhancements: None.

References: None.

SA-17 DEVELOPER SECURITY AND PRIVACY ARCHITECTURE AND DESIGN

Control: Require the developer of the system, system component, or system service to produce a design specification and security and privacy architecture that:

a. Is consistent with the organization's security and privacy architecture that is an integral part the organization's enterprise architecture;

b. Accurately and completely describes the required security and privacy functionality, and the allocation of controls among physical and logical components; and

c. Expresses how individual security and privacy functions, mechanisms, and services work together to provide required security and privacy capabilities and a unified approach to protection.

Discussion: Developer security and privacy architecture and design are directed at external developers, although they could also be applied to internal (in-house) development. In contrast, PL-8 is directed at internal developers to ensure that organizations develop a security and privacy

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

architecture that is integrated with the enterprise architecture. The distinction between SA-17 and PL-8 is especially important when organizations outsource the development of systems, system components, or system services and when there is a requirement to demonstrate consistency with the enterprise architecture and security and privacy architecture of the organization. [ISO 15408-2], [ISO 15408-3], and [SP 800-160-1] provide information on security architecture and design, including formal policy models, security-relevant components, formal and informal correspondence, conceptually simple design, and structuring for least privilege and testing.

Related Controls: PL-2, PL-8, PM-7, SA-3, SA-4, SA-8, SC-7.

Control Enhancements:

(1) DEVELOPER SECURITY AND PRIVACY ARCHITECTURE AND DESIGN | FORMAL POLICY MODEL

Require the developer of the system, system component, or system service to:

(a) Produce, as an integral part of the development process, a formal policy model describing the [*Assignment: organization-defined elements of organizational security and privacy policy*] to be enforced; and

(b) Prove that the formal policy model is internally consistent and sufficient to enforce the defined elements of the organizational security and privacy policy when implemented.

Discussion: Formal models describe specific behaviors or security and privacy policies using formal languages, thus enabling the correctness of those behaviors and policies to be formally proven. Not all components of systems can be modeled. Generally, formal specifications are scoped to the behaviors or policies of interest, such as nondiscretionary access control policies. Organizations choose the formal modeling language and approach based on the nature of the behaviors and policies to be described and the available tools.

Related Controls: AC-3, AC-4, AC-25.

(2) DEVELOPER SECURITY AND PRIVACY ARCHITECTURE AND DESIGN | SECURITY-RELEVANT COMPONENTS

Require the developer of the system, system component, or system service to:

(a) Define security-relevant hardware, software, and firmware; and

(b) Provide a rationale that the definition for security-relevant hardware, software, and firmware is complete.

Discussion: The security-relevant hardware, software, and firmware represent the portion of the system, component, or service that is trusted to perform correctly to maintain required security properties.

Related Controls: AC-25, SA-5.

(3) DEVELOPER SECURITY AND PRIVACY ARCHITECTURE AND DESIGN | FORMAL CORRESPONDENCE

Require the developer of the system, system component, or system service to:

(a) Produce, as an integral part of the development process, a formal top-level specification that specifies the interfaces to security-relevant hardware, software, and firmware in terms of exceptions, error messages, and effects;

(b) Show via proof to the extent feasible with additional informal demonstration as necessary, that the formal top-level specification is consistent with the formal policy model;

(c) Show via informal demonstration, that the formal top-level specification completely covers the interfaces to security-relevant hardware, software, and firmware;

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

(d) Show that the formal top-level specification is an accurate description of the implemented security-relevant hardware, software, and firmware; and

(e) Describe the security-relevant hardware, software, and firmware mechanisms not addressed in the formal top-level specification but strictly internal to the security-relevant hardware, software, and firmware.

Discussion: Correspondence is an important part of the assurance gained through modeling. It demonstrates that the implementation is an accurate transformation of the model, and that any additional code or implementation details that are present have no impact on the behaviors or policies being modeled. Formal methods can be used to show that the high-level security properties are satisfied by the formal system description, and that the formal system description is correctly implemented by a description of some lower level, including a hardware description. Consistency between the formal top-level specification and the formal policy models is generally not amenable to being fully proven. Therefore, a combination of formal and informal methods may be needed to demonstrate such consistency. Consistency between the formal top-level specification and the actual implementation may require the use of an informal demonstration due to limitations on the applicability of formal methods to prove that the specification accurately reflects the implementation. Hardware, software, and firmware mechanisms internal to security-relevant components include mapping registers and direct memory input and output.

Related Controls: AC-3, AC-4, AC-25, SA-4, SA-5.

(4) DEVELOPER SECURITY AND PRIVACY ARCHITECTURE AND DESIGN | INFORMAL CORRESPONDENCE

Require the developer of the system, system component, or system service to:

(a) Produce, as an integral part of the development process, an informal descriptive top-level specification that specifies the interfaces to security-relevant hardware, software, and firmware in terms of exceptions, error messages, and effects;

(b) Show via [*Selection: informal demonstration; convincing argument with formal methods as feasible*] that the descriptive top-level specification is consistent with the formal policy model;

(c) Show via informal demonstration, that the descriptive top-level specification completely covers the interfaces to security-relevant hardware, software, and firmware;

(d) Show that the descriptive top-level specification is an accurate description of the interfaces to security-relevant hardware, software, and firmware; and

(e) Describe the security-relevant hardware, software, and firmware mechanisms not addressed in the descriptive top-level specification but strictly internal to the security-relevant hardware, software, and firmware.

Discussion: Correspondence is an important part of the assurance gained through modeling. It demonstrates that the implementation is an accurate transformation of the model, and that additional code or implementation detail has no impact on the behaviors or policies being modeled. Consistency between the descriptive top-level specification (i.e., high-level/low-level design) and the formal policy model is generally not amenable to being fully proven. Therefore, a combination of formal and informal methods may be needed to show such consistency. Hardware, software, and firmware mechanisms strictly internal to security-relevant hardware, software, and firmware include mapping registers and direct memory input and output.

Related Controls: AC-3, AC-4, AC-25, SA-4, SA-5.

(5) DEVELOPER SECURITY AND PRIVACY ARCHITECTURE AND DESIGN | CONCEPTUALLY SIMPLE DESIGN

Require the developer of the system, system component, or system service to:

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

(a) Design and structure the security-relevant hardware, software, and firmware to use a complete, conceptually simple protection mechanism with precisely defined semantics; and

(b) Internally structure the security-relevant hardware, software, and firmware with specific regard for this mechanism.

Discussion: The principle of reduced complexity states that the system design is as simple and small as possible (see SA-8(7)). A small and simple design is easier to understand and analyze and is also less prone to error (see AC-25, SA-8(13)). The principle of reduced complexity applies to any aspect of a system, but it has particular importance for security due to the various analyses performed to obtain evidence about the emergent security property of the system. For such analyses to be successful, a small and simple design is essential. Application of the principle of reduced complexity contributes to the ability of system developers to understand the correctness and completeness of system security functions and facilitates the identification of potential vulnerabilities. The corollary of reduced complexity states that the simplicity of the system is directly related to the number of vulnerabilities it will contain. That is, simpler systems contain fewer vulnerabilities. An important benefit of reduced complexity is that it is easier to understand whether the security policy has been captured in the system design and that fewer vulnerabilities are likely to be introduced during engineering development. An additional benefit is that any such conclusion about correctness, completeness, and existence of vulnerabilities can be reached with a higher degree of assurance in contrast to conclusions reached in situations where the system design is inherently more complex.

Related Controls: AC-25, SA-8, SC-3.

(6) DEVELOPER SECURITY AND PRIVACY ARCHITECTURE AND DESIGN | STRUCTURE FOR TESTING

Require the developer of the system, system component, or system service to structure security-relevant hardware, software, and firmware to facilitate testing.

Discussion: Applying the security design principles in [SP 800-160-1] promotes complete, consistent, and comprehensive testing and evaluation of systems, system components, and services. The thoroughness of such testing contributes to the evidence produced to generate an effective assurance case or argument as to the trustworthiness of the system, system component, or service.

Related Controls: SA-5, SA-11.

(7) DEVELOPER SECURITY AND PRIVACY ARCHITECTURE AND DESIGN | STRUCTURE FOR LEAST PRIVILEGE

Require the developer of the system, system component, or system service to structure security-relevant hardware, software, and firmware to facilitate controlling access with least privilege.

Discussion: The principle of least privilege states that each component is allocated sufficient privileges to accomplish its specified functions but no more (see SA-8(14)). Applying the principle of least privilege limits the scope of the component's actions, which has two desirable effects. First, the security impact of a failure, corruption, or misuse of the system component results in a minimized security impact. Second, the security analysis of the component is simplified. Least privilege is a pervasive principle that is reflected in all aspects of the secure system design. Interfaces used to invoke component capability are available to only certain subsets of the user population, and component design supports a sufficiently fine granularity of privilege decomposition. For example, in the case of an audit mechanism, there may be an interface for the audit manager, who configures the audit settings; an interface for the audit operator, who ensures that audit data is safely collected and stored; and, finally, yet another interface for the audit reviewer, who only has a need to view the audit data that has been collected but no need to perform operations on that data.

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

In addition to its manifestations at the system interface, least privilege can be used as a guiding principle for the internal structure of the system itself. One aspect of internal least privilege is to construct modules so that only the elements encapsulated by the module are directly operated upon by the functions within the module. Elements external to a module that may be affected by the module's operation are indirectly accessed through interaction (e.g., via a function call) with the module that contains those elements. Another aspect of internal least privilege is that the scope of a given module or component includes only those system elements that are necessary for its functionality, and the access modes to the elements (e.g., read, write) are minimal.

Related Controls: AC-5, AC-6, SA-8.

(8) DEVELOPER SECURITY AND PRIVACY ARCHITECTURE AND DESIGN | ORCHESTRATION

Design [*Assignment: organization-defined critical systems or system components*] with coordinated behavior to implement the following capabilities: [*Assignment: organization-defined capabilities, by system or component*].

Discussion: Security resources that are distributed, located at different layers or in different system elements, or are implemented to support different aspects of trustworthiness can interact in unforeseen or incorrect ways. Adverse consequences can include cascading failures, interference, or coverage gaps. Coordination of the behavior of security resources (e.g., by ensuring that one patch is installed across all resources before making a configuration change that assumes that the patch is propagated) can avert such negative interactions.

Related Controls: None.

(9) DEVELOPER SECURITY AND PRIVACY ARCHITECTURE AND DESIGN | DESIGN DIVERSITY

Use different designs for [*Assignment: organization-defined critical systems or system components*] to satisfy a common set of requirements or to provide equivalent functionality.

Discussion: Design diversity is achieved by supplying the same requirements specification to multiple developers, each of whom is responsible for developing a variant of the system or system component that meets the requirements. Variants can be in software design, in hardware design, or in both hardware and a software design. Differences in the designs of the variants can result from developer experience (e.g., prior use of a design pattern), design style (e.g., when decomposing a required function into smaller tasks, determining what constitutes a separate task and how far to decompose tasks into sub-tasks), selection of libraries to incorporate into the variant, and the development environment (e.g., different design tools make some design patterns easier to visualize). Hardware design diversity includes making different decisions about what information to keep in analog form and what information to convert to digital form, transmitting the same information at different times, and introducing delays in sampling (temporal diversity). Design diversity is commonly used to support fault tolerance.

Related Controls: None.

References: [ISO 15408-2], [ISO 15408-3], [SP 800-160-1].

SA-18 TAMPER RESISTANCE AND DETECTION

[Withdrawn: Moved to SR-9.]

Control Enhancements:

(1) TAMPER RESISTANCE AND DETECTION | MULTIPLE PHASES OF SYSTEM DEVELOPMENT LIFE CYCLE

[Withdrawn: Moved to SR-9(1).]

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

(2) TAMPER RESISTANCE AND DETECTION | INSPECTION OF SYSTEMS OR COMPONENTS

[Withdrawn: Moved to SR-10.]

SA-19 COMPONENT AUTHENTICITY

[Withdrawn: Moved to SR-11.]

Control Enhancements:

(1) COMPONENT AUTHENTICITY | ANTI-COUNTERFEIT TRAINING

[Withdrawn: Moved to SR-11(1).]

(2) COMPONENT AUTHENTICITY | CONFIGURATION CONTROL FOR COMPONENT SERVICE AND REPAIR

[Withdrawn: Moved to SR-11(2).]

(3) COMPONENT AUTHENTICITY | COMPONENT DISPOSAL

[Withdrawn: Moved to SR-12.]

(4) COMPONENT AUTHENTICITY | ANTI-COUNTERFEIT SCANNING

[Withdrawn: Moved to SR-11(3).]

SA-20 CUSTOMIZED DEVELOPMENT OF CRITICAL COMPONENTS

Control: Reimplement or custom develop the following critical system components: [*Assignment: organization-defined critical system components*].

Discussion: Organizations determine that certain system components likely cannot be trusted due to specific threats to and vulnerabilities in those components for which there are no viable security controls to adequately mitigate risk. Reimplementation or custom development of such components may satisfy requirements for higher assurance and is carried out by initiating changes to system components (including hardware, software, and firmware) such that the standard attacks by adversaries are less likely to succeed. In situations where no alternative sourcing is available and organizations choose not to reimplement or custom develop critical system components, additional controls can be employed. Controls include enhanced auditing, restrictions on source code and system utility access, and protection from deletion of system and application files.

Related Controls: CP-2, RA-9, SA-8.

Control Enhancements: None.

References: [SP 800-160-1].

SA-21 DEVELOPER SCREENING

Control: Require that the developer of [*Assignment: organization-defined system, system component, or system service*]:

a. Has appropriate access authorizations as determined by assigned [*Assignment: organization-defined official government duties*]; and

b. Satisfies the following additional personnel screening criteria: [*Assignment: organization-defined additional personnel screening criteria*].

Discussion: Developer screening is directed at external developers. Internal developer screening is addressed by PS-3. Because the system, system component, or system service may be used in critical activities essential to the national or economic security interests of the United States, organizations have a strong interest in ensuring that developers are trustworthy. The degree of

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

trust required of developers may need to be consistent with that of the individuals who access the systems, system components, or system services once deployed. Authorization and personnel screening criteria include clearances, background checks, citizenship, and nationality. Developer trustworthiness may also include a review and analysis of company ownership and relationships that the company has with entities that may potentially affect the quality and reliability of the systems, components, or services being developed. Satisfying the required access authorizations and personnel screening criteria includes providing a list of all individuals who are authorized to perform development activities on the selected system, system component, or system service so that organizations can validate that the developer has satisfied the authorization and screening requirements.

Related Controls: PS-2, PS-3, PS-6, PS-7, SA-4, SR-6.

Control Enhancements:

(1) DEVELOPER SCREENING | VALIDATION OF SCREENING

[Withdrawn: Incorporated into SA-21.]

References: None.

SA-22 UNSUPPORTED SYSTEM COMPONENTS

Control:

a. Replace system components when support for the components is no longer available from the developer, vendor, or manufacturer; or

b. Provide the following options for alternative sources for continued support for unsupported components [*Selection (one or more): in-house support;* [*Assignment: organization-defined support from external providers*]].

Discussion: Support for system components includes software patches, firmware updates, replacement parts, and maintenance contracts. An example of unsupported components includes when vendors no longer provide critical software patches or product updates, which can result in an opportunity for adversaries to exploit weaknesses in the installed components. Exceptions to replacing unsupported system components include systems that provide critical mission or business capabilities where newer technologies are not available or where the systems are so isolated that installing replacement components is not an option.

Alternative sources for support address the need to provide continued support for system components that are no longer supported by the original manufacturers, developers, or vendors when such components remain essential to organizational mission and business functions. If necessary, organizations can establish in-house support by developing customized patches for critical software components or, alternatively, obtain the services of external providers who provide ongoing support for the designated unsupported components through contractual relationships. Such contractual relationships can include open-source software value-added vendors. The increased risk of using unsupported system components can be mitigated, for example, by prohibiting the connection of such components to public or uncontrolled networks, or implementing other forms of isolation.

Related Controls: PL-2, SA-3.

Control Enhancements:

(1) UNSUPPORTED SYSTEM COMPONENTS | ALTERNATIVE SOURCES FOR CONTINUED SUPPORT

[Withdrawn: Incorporated into SA-22.]

References: None.

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5

SA-23 SPECIALIZATION

Control: Employ [*Selection (one or more): design; modification; augmentation; reconfiguration*] on [*Assignment: organization-defined systems or system components*] supporting mission essential services or functions to increase the trustworthiness in those systems or components.

Discussion: It is often necessary for a system or system component that supports mission-essential services or functions to be enhanced to maximize the trustworthiness of the resource. Sometimes this enhancement is done at the design level. In other instances, it is done post-design, either through modifications of the system in question or by augmenting the system with additional components. For example, supplemental authentication or non-repudiation functions may be added to the system to enhance the identity of critical resources to other resources that depend on the organization-defined resources.

Related Controls: RA-9, SA-8.

Control Enhancements: None.

References: [SP 800-160-1], [SP 800-160-2].

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.800-53r5