FFIEC Information Technology Examination Handbook

Development, Acquisition, and Maintenance

AUGUST 2024

INTRODUCTION 1

I .... OVERVIEW OF DEVELOPMENT, ACQUISITION, AND MAINTENANCE 3

II .... GOVERNANCE OF DEVELOPMENT, ACQUISITION, AND MAINTENANCE 5

II.A Policies, Standards, and Procedures 6

II.B Roles and Responsibilities 6

II.B.1 Board, Senior Management, and Other Common Roles 7

II.B.2 IT Project Management Roles 8

II.B.3 Development Roles 10

II.B.4 Acquisition Roles 12

II.B.5 Maintenance Roles 14

II.B.6 Other Common Development, Acquisition, and Maintenance Roles 14

II.B.7 Supply Chain Roles 15

II.B.8 Other Support Functions 15

II.B.9 Audit’s Role 16

III .... RISK MANAGEMENT OF DEVELOPMENT, ACQUISITION, AND MAINTENANCE 18

III.A Risk Identification 19

III.B Risk Measurement 20

III.C Risk Monitoring and Reporting 20

III.D Controlling or Mitigating Risk 21

IV .... COMMON DEVELOPMENT, ACQUISITION, AND MAINTENANCE RISK TOPICS 22

IV.A Open-Source 23

IV.B Commercial-off-the-Shelf 25

IV.C Licenses, Agreements, and Copyright Protection 26

IV.C.1 Software Licenses 27

IV.C.1(a) Free and Open-Source Software Licenses 28

IV.C.1(b) Proprietary Software Licenses 28

IV.C.2 Hardware Licenses 29

IV.C.3 Copyright Protection 29

IV.D Secure Development 30

IV.E Data 32
IV.F Secure Operating Environments 32
IV.G Microservices 33
IV.H Containers 37
IV.I Application Programming Interfaces 43
IV.I.1 API Gateway 44
IV.I.2 API Risk Mitigation 46
IV.J Methodologies 48
IV.J.1 Waterfall 49
IV.J.2 Agile 49
IV.K Quality Management 52
IV.L Documentation Standards 53
IV.M Post-Implementation Review 54
IV.N IT Project Management 55
IV.N.1 IT Project Phases 56
IV.N.1(a) Initiation 57
IV.N.1(b) Planning 58
IV.N.1(c) Execution 59
IV.N.1(d) Closeout 59
IV.N.2 Monitoring and Controlling 60
IV.N.3 IT Project Documentation 61
IV.N.3(a) IT Project Request 61
IV.N.3(b) Business Case 62
IV.N.3(c) Feasibility Study 62
IV.N.3(d) IT Project Plans 63
IV.N.3(e) Closeout Documentation 63
IV.O System Development Life Cycle 64
IV.O.1 SDLC Phases 65
IV.O.1(a) Initiation 66
IV.O.1(b) Development or Acquisition 68
IV.O.1(c) Implementation and Assessment 69
IV.O.1(d) Operations and Maintenance 71
IV.O.1(e) Sunset and Disposal 72
IV.P Third-Party Relationship Risk Management 74
IV.P.1 Planning 74

IV.P.2 Due Diligence and Third-Party Selection 74
IV.P.3 Contract Negotiation 75
IV.Q Supply Chain Considerations 75
IV.Q.1 Supply Chain Risk Management 77
IV.Q.2 Software Bill of Material 89
IV.Q.3 Enterprise Risk Management and Supply Chain Risks 90

V .... DEVELOPMENT 91
V.A Development Standards and Controls 93
V.B Testing 96
V.C DevOps and DevSecOps 99
V.C.1 DevOps 100
V.C.2 DevSecOps 100
V.D Functional Development Types 105
V.D.1 Model Development 106
V.D.2 Database Development 107

VI .... ACQUISITION 109
VI.A Acquisition Policies, Standards, and Procedures 111
VI.B Acquisition Projects 113
VI.C Solicitation 113
VI.D Evaluation 116
VI.E Contracts and Other Agreements 117
VI.E.1 Statement of Work 117
VI.E.2 Master Services Agreement 119
VI.E.3 Service Level Agreement 120
VI.E.4 Contracts 120
VI.E.5 Escrowed Source Code Agreements and Documentation 124
VI.E.6 Exit Strategy 125

VII .... MAINTENANCE 126
VII.A Preventive Maintenance 127
VII.B Change Management 127
VII.B.1 Implementing Changes 129
VII.B.2 Additional Control Considerations in Change Management 132
VII.B.2(a) Data Controls in the Testing Environment 132
VII.B.2(b) Library Controls 132

VII.B.2(c) Code Repository Controls ... 133
VII.B.3 Change Types ... 135
VII.B.3(a) Routine Modifications ... 135
VII.B.3(b) Major Modifications ... 136
VII.B.3(c) Emergency Modifications ... 137
VII.B.4 Change Management Documentation ... 139
VII.B.4(a) Change Request Form ... 140
VII.B.4(b) Impact Analysis ... 140
VII.B.4(c) Rollback or Back-Out Plan ... 141
VII.C End-of-Life ... 142
VII.D Termination and Disposal ... 144
VII.E Maintenance Documentation ... 145
APPENDIX A: EXAMINATION PROCEDURES ... 146
APPENDIX B: GLOSSARY ... 182
APPENDIX C: ABBREVIATIONS ... 202
APPENDIX D: REFERENCES ... 204

INTRODUCTION

The “Development, Acquisition, and Maintenance” booklet is one in a series of booklets that compose the *Federal Financial Institutions Examination Council* (*FFIEC*)[1] *Information Technology Examination Handbook (IT Handbook)*. The *FFIEC IT Handbook* is prepared for use by examiners.[2] With the publication of this booklet, the FFIEC members replace the “Development and Acquisition” booklet issued in April 2004. The revised title now reflects the importance of maintenance in the life of a system or component.[3] This booklet

- Describes system and component development, acquisition, and maintenance.
- Highlights key risk management practices when developing, acquiring, or maintaining systems and components.
- Provides an overview of and discusses information technology (IT) project management, the system development life cycle (SDLC), and supply chain risk management (SCRM).
- Addresses the importance of system and software maintenance to an entity’s resilience.

For *FFIEC IT Handbook* purposes, the term “entity” includes depository financial institutions,[4] nonbank financial institutions,[5] bank holding companies,[6] savings and loan holding companies,[7] and third-party service providers.[8]

[1] The FFIEC was established on March 10, 1979, pursuant to Title X of the Financial Institutions Regulatory and Interest Rate Control Act of 1978, Pub. L. 95-630. The FFIEC comprises the principals of the Board of Governors of the Federal Reserve System (FRB), the Consumer Financial Protection Bureau (CFPB), the Federal Deposit Insurance Corporation (FDIC), the National Credit Union Administration (NCUA), the Office of the Comptroller of the Currency (OCC), and the State Liaison Committee (SLC).

[2] Each FFIEC member uses the principles outlined in this booklet consistent with the member’s supervisory authority.

[3] Examples of systems and components include hardware, firmware, software, peripherals, and network components.

[4] The term “depository financial institution” includes national banks, federal savings associations, state savings associations, state member banks, state nonmember banks, and credit unions.

[5] The term “nonbank financial institution” includes nondepository financial institutions under the jurisdiction of either state banking departments or the CFPB.

[6] The term “bank holding company” includes any company that has control over any bank or over any company that is or becomes a bank holding company as defined by the Bank Holding Company Act.

[7] The term “savings and loan holding company” includes any company that directly or indirectly controls a savings association or controls any other company that is a savings and loan holding company as defined by the Home Owners’ Loan Act.

[8] The term “third-party service provider” means third parties that provide services, the provision of which is subject to examination under the Bank Service Company Act, the Home Owners’ Loan Act, the Dodd–Frank Wall Street Reform and Consumer Protection Act, or other relevant law.

This booklet does not impose new requirements on entities. Instead, this booklet describes the principles and practices that examiners can use when assessing an entity's system development, acquisition, and maintenance activities.

Appendix A of this booklet provides objectives-based examination procedures. Application of principles and related examination procedures will vary consistent with the examined entity's complexity and risk profile (including the size of the entity or the nature of the systems and components).

I OVERVIEW OF DEVELOPMENT, ACQUISITION, AND MAINTENANCE

Development, acquisition, and maintenance activities are integral to an entity's operations. This booklet discusses how weaknesses in IT development, acquisition, and maintenance processes may lead to issues with confidentiality, integrity, availability, and resilience of the entity's systems, components, and data. Management determines the systems, products, and services that the entity will provide, whether to develop or acquire them, and how to maintain and service those systems, products, and services. Generally, whether developing or acquiring systems, products, and services, management performs some form of acquisition activity, including procurement. Management also oversees maintenance activities to prolong the life of systems and components (i.e., IT assets) and support continuity of operations. Management should plan for maintenance activities from the outset of acquisition or development to ensure secure continuity of operations. The following definitions and explanations provide an overview of IT development, acquisition, and maintenance.

Development: Systematic application of knowledge toward the production of useful materials, devices, and systems, or processes of defining, designing, testing, and implementing systems or components. Development includes validation and demonstration of a chosen technology, use of test and production environments, improvement of developed prototypes, integration into systems and subsystems, and inclusion of hardware builds. Development activities may be performed by the entity's personnel or third parties who develop systems and components on the entity's behalf. Management may choose to acquire systems and components from third parties and may customize them to meet the entity's needs.

Acquisition: All stages for acquiring products or services, beginning with determining the need for the product or service and ending with contract completion and closeout. Acquisition generally involves creating a relationship with a third party in the supply chain; therefore, effective SCRM is integral to the acquisition process. Acquisition activities include procurement processes that help ensure that management receives the contracted products or services. Acquisition activities, including procurement processes, help management achieve and maintain confidentiality, integrity, availability, as well as resilience, including supply chain resilience, throughout the life of systems and components.

Maintenance: Any act that either prevents the failure or malfunction of equipment or restores its operating capability. This includes incremental changes to improve performance. Maintenance activities include the processes to monitor systems and components and make changes (e.g., install patches and add new functions) to prevent their failure or malfunction and continue to meet user and customer needs. Management should perform preventive maintenance throughout the IT asset's useful life to prevent or minimize catastrophic failure and promote confidentiality, integrity, availability, and resilience. Maintenance may be performed by the entity's personnel or third parties. Maintenance activities should be performed regardless of the origin or location (e.g., geographic or virtual) of the systems or components.

The next section of this booklet addresses governance and risk management elements. Also addressed are common risk topics related to development, acquisition, and maintenance. The booklet explains key topics, such as IT project management, SDLC, and supply chain risk management considerations, which are integral to development, acquisition, and maintenance activities to provide for ongoing operations. Appendices provide examination procedures, agency and industry references, and a glossary.

II GOVERNANCE OF DEVELOPMENT, ACQUISITION, AND MAINTENANCE

Action Summary

Management should establish, and the board of directors (board) should oversee, an effective governance framework for development, acquisition, and maintenance activities. Additionally, the board should oversee related IT project management processes to manage projects related to those activities.

Examiners should review the following:

- Enterprise-wide IT policies, procedures, and standards describing the entity's requirements throughout the development, acquisition, and maintenance life cycle.
- Charters (e.g., board, management, or committee), organizational charts, relevant documentation, and practices to determine whether appropriate roles and responsibilities are identified and assigned with suitable decision-making authority.
- Project plans and meeting minutes of the board and committees to ensure that activities and projects align with the entity's strategic objectives and the board's risk appetite.
- Project audit reports to determine whether the audit function provides independent, objective assurance of the effectiveness of an entity's development, acquisition, and maintenance activities.
- Documentation of controls for personnel with access to program code to limit placement into the production environment.
- Quality assurance (QA) reports to evaluate processes for the detection of potential coding errors.

The board[9] should oversee and management should establish an effective governance structure that allows for the effective oversight and management of the development, acquisition, and maintenance of the entity's systems and components. Additionally, the board should oversee related IT project management processes and projects related to those activities. The board should oversee significant projects to ensure that they align with an entity's strategic plans. Projects that do not align may cause lost revenue or diminished economies of scale resulting in budget shortfalls. Misalignment may cause unexpected harm to the entity or its reputation (e.g., customer dissatisfaction, user frustration, noncompliance with laws and regulations, and IT or operational performance issues). Ineffective communication or coordination by the board with appropriate stakeholders can lead to problems (e.g., scope creep on projects and cost overruns) and processes that do not facilitate appropriate communication across business departments. When IT activities are siloed away from business departments or other business line IT departments (i.e., when an entity has multiple lines of business that have their own IT departments), it may result in management establishing counterproductive objectives.

[9] Most financial institutions have boards; however, not all third-party service providers do. When an entity does not have a board, senior leaders may have the responsibilities of a board as described in this booklet.

Management should consider the needs of internal and external stakeholders[10] when making decisions regarding IT development, acquisition, and maintenance activities. Effective communication helps stakeholders carry out their roles and responsibilities and supports entity-wide alignment of these IT activities with an entity's strategic plans. Effective communication also helps stakeholders appropriately identify, assess, and mitigate risks and their potential impacts. Additionally, when contemplating IT projects and activities, management should consider business strategies and objectives, resilience needs, information security requirements, legal and regulatory requirements, and allocation of resources (e.g., personnel, budget, and time).

II.A Policies, Standards, and Procedures

Effective management develops and implements comprehensive, entity-wide IT policies, standards, and procedures covering the development, acquisition, and maintenance life cycle. Large or complex entities may have multiple policies for different business line functions and their underlying systems and components. The board and designated owners or committees should regularly review and approve IT policies. Policies should clearly delineate development, acquisition, and maintenance responsibilities and provide for communication to all personnel, stakeholders, and appropriate third parties. Standards, on the other hand, may be used to define the processes and rules to support IT policies. For example, software developers often use coding standards in their work. Procedures are often developed to provide instructions to perform a function or task to align with IT policies and standards and may be adjusted to meet changes in the entity's IT environment. Periodic adjustments to standards and procedures may be needed to continue to align with the board-defined risk appetite, threat environment, and risks to the entity and its customers. Management should review and approve deviations from policies, standards, and procedures related to development, acquisition, and maintenance. For example, if the entity's policy prohibits the use of open-source software, management should approve any exceptions when developers use open-source software.

II.B Roles and Responsibilities

The board should oversee and management should implement the entity's development, acquisition, and maintenance activities by identifying and assigning appropriate roles and responsibilities. Assigning roles and responsibilities helps management to assign decision-making authority to an appropriate individual or team. Designated roles and responsibilities allow management and stakeholders to identify relevant lines of communication. Additionally, management may specify qualifications, competence, skills, and expertise needed in various roles. Examples of common roles related to development, acquisition, and maintenance activities are described in the following sections of this booklet. In smaller or less complex entities, formal roles may not be established; however, management should identify and assign the responsibilities described in these sections. As appropriate for the needs of the entity, it is important to consider training needs to support development, acquisition, and maintenance

[10] For purposes of this booklet, stakeholders may include management or designated individuals from all departments affected by the proposed project (e.g., information technology [IT], legal, compliance, independent risk management, business process owners, and audit), and board members. At times, stakeholders can also include external entities (e.g., clients, consumers, customers, or customers of financial institution clients of a third-party service provider).

responsibilities. For more information about IT roles and responsibilities in general, refer to the *FFIEC IT Handbook's* "Management" booklet.

II.B.1 Board, Senior Management, and Other Common Roles

Primary roles and responsibilities for the board and senior management for development, acquisition, and maintenance activities should include the following:

- **Board:** The board, or board-designated committee,[11] typically confirms that IT-related development, acquisition, and maintenance activities align with entity strategic objectives and the board's risk appetite. The audit committee of the board is involved with reviewing and validating the entity's audit-related activities, including development, acquisition, and maintenance.[12] Additionally, the board generally approves
    - Significant IT investments.
    - Policies and standards.
    - Significant IT projects that may affect the entity's strategic direction.[13]

  Board members should have appropriate knowledge of risks to provide a credible challenge[14] to management responsible for development, acquisition, and maintenance functions.

- **Senior management:** Senior management approves and champions development, acquisition, and maintenance projects within its authority, ensures that project execution is in alignment with entity objectives and risk appetite, and ensures that adequate resources, including qualified staff, are available to complete IT projects.

- **Chief information officer (CIO):** The CIO leads day-to-day IT system and component development, acquisition, and maintenance in accordance with the entity's policies and business strategy. Prudent practices include the CIO consulting with the entity's risk management function and legal counsel to ensure that development, acquisition, and maintenance decisions are consistent with sound risk management principles and applicable laws and regulations. Additionally, the CIO facilitates the integration of any new projects, systems, and components into the entity's operations to ensure that they align with strategic objectives and IT strategies. The CIO helps determine the feasibility of proposed projects or changes.

[11] For purposes of this booklet, when the role of the board is mentioned, it can be inferred that this role may be fulfilled by a board-designated committee, as appropriate, unless otherwise stated. An example of a board-designated committee is the architectural review board, which is common in larger or more complex entities.

[12] For more on the general role of audit and its responsibilities, refer to the *FFIEC IT Handbook's* "Audit" booklet.

[13] Generally, the board approves critical operations and core business functions. For more information, refer to the following guidance: FDIC Financial Institution Letter (FIL) 103-2020, "The FDIC Publishes Sound Practices to Strengthen Operational Resilience"; FRB Supervision and Regulation (SR) Letter 20-24, "Interagency Paper on Sound Practices to Strengthen Operational Resilience"; and OCC Bulletin 2020-94, "Operational Risk: Sound Practices to Strengthen Operational Risk."

[14] A credible challenge involves being actively engaged, asking thoughtful questions, and exercising independent judgment.

- **Chief information security officer (CISO):** The CISO typically develops enterprise-wide cybersecurity and information security policies and standards, including procedures promoting secure development, acquisition, and maintenance practices, and provides input for the consideration and inclusion of security and resilience in IT projects. The CISO is responsible for validating that security and resilience are maintained throughout the life of a project, system, or component.

- **IT steering committee:** An IT steering committee typically reviews and approves major development, acquisition, and maintenance proposals. Additionally, the committee provides a credible challenge to IT project managers and management responsible for development, acquisition, and maintenance activities. The credible challenge allows management to verify that projects adhere to entity policies, standards, and procedures; that project managers identify and appropriately mitigate risks; and that project costs stay within the allocated budget. The IT steering committee is responsible for reporting to the board and overseeing any issues (e.g., project issues or failure to meet standards) or additional project needs (e.g., additional funds and resources) by translating technical concepts in a way that is clear and understandable to relevant stakeholders.

II.B.2 IT Project Management Roles

There are multiple IT project management roles that, when performed effectively, can play a significant part in a project's success. A project can be part of larger programs, which may include multiple related projects. Roles may overlap in smaller or less complex entities or projects, which is acceptable if mitigating controls are in place. Segregation of duties and dual controls or other compensating controls are important for development, acquisition, and maintenance projects. For example, segregation of duties helps ensure that one person cannot use projects to misappropriate entity funds; purchase substandard software, hardware, or components; or initiate inappropriate contracts. Additionally, a key concern is that seemingly small problems in IT projects can quickly escalate in scale. Dual control may be used to mitigate this risk because it allows for problems to be identified and addressed earlier since no single person or group can make final decisions without agreement from others. For example, during development, code review involves a second individual to review code before it is released to test environments. Segregation of duties and dual control help foster security and resilience. For more information, refer to the "IT Project Management" section of this booklet. Common IT project management roles and responsibilities include the following:

- **Sponsor:** A project's sponsor is the most senior role in the project or program, which may contain multiple projects. Depending on the project's size and complexity, there may be more than one sponsor. Responsibilities include obtaining stakeholder endorsement for the project based on its objectives and business case,[15] providing resources, and appointing the project owner. The sponsor represents the project in discussions with senior management, provides both formal and informal support to project teams, and promotes the positive aspects of the changes resulting from the project. The sponsor works with senior stakeholders

[15] A business case provides justification to undertake a project, and the business case identifies the benefit, costs, and provides a rationale for the preferred solution. For more information, refer to the Project Management Institute (PMI), "Is This Really Worth the Effort? The Need for a Business Case."

to resolve strategic issues across projects in a program or among projects. Additionally, the sponsor approves milestones and confirms the project's successful delivery.

- **Project owners:** A project owner typically helps determine a project's vision and guides progress toward the objectives to meet that vision throughout the project life cycle. The project owner manages key project risks to maintain alignment between the entity's objectives and those of the project. Additionally, the project owner obtains resources from the sponsor, serves as the primary communication link with management and other stakeholders, and is responsible for the successful completion of the project.
- **Product owner:** A product owner represents the business line stakeholders (e.g., team, department, or customer) for the team that undertakes the project. The product owner ensures that the project team delivers the most value by helping to prioritize deliverables. The product owner is responsible for communicating project status with the business line stakeholders.
- **Project or program managers:** Project or program managers are responsible for working with the business change manager to plan and monitor the project or program tasks. Additionally, they verify that tasks align with the objectives throughout the project's or program's life cycle. They are responsible for defining success and providing oversight for the completion of tasks and successful delivery of the project's or program's objectives. Project or program managers tend to have practical project management experience rather than specific business line expertise. They monitor and manage resources to stay within timelines and budget, including coordinating between projects and managing project interdependencies. Additionally, project or program managers manage issues and direct corrective action when necessary. They monitor, report, and communicate a project's or program's status to management and other stakeholders. Lastly, project or program managers maintain all project documentation.
- **Business change manager:** A business change manager represents the business line for projects or programs in discussions with others. The role includes highlighting and focusing on benefits and risks for the business line and identifying the appropriate timing (for example, during periods of lower volumes) of implementation and releases. The business change manager plans and manages the integration of the changes in business line systems or processes, including identifying new business processes, to meet strategic objectives. A project or program may have multiple business change managers representing their respective lines of business. The business change manager defines and monitors key performance indicators to objectively measure the project's or program's success. The business change manager translates the technical aspects and benefits of a project or program into a format that is understandable to and executable by business line management and staff. Business change managers plan and manage business continuity during and immediately after the change takes place.
- **Project or program management office or organization (PMO) personnel:** Traditional project management methodologies, such as waterfall, typically centralize project governance through the PMO. The entity's size, complexity, and project types determine the number and qualifications of PMO personnel. Larger or more complex entities often have numerous PMO personnel to coordinate the projects and programs undertaken by an entity. Smaller or less complex entities often do not have designated PMOs or dedicated project personnel. The IT project's size and complexity determine the PMO's extent of engagement. Beyond keeping projects on track and identifying and removing obstacles, PMO activities

generally involve analyzing the project's feasibility and risk; determining whether to develop, acquire, or outsource systems and components; ensuring quality of deliverables; and verifying that the project delivers value to the entity and affected business units.

- **Stakeholders:** Stakeholders should be involved early in the planning process to provide input before finalizing project requirements. In addition, regular communication between project managers and stakeholders helps to ensure that stakeholders agree with the project's proposed direction. The list of potential stakeholders can be internal or external and may include the following individuals:[16]

Internal stakeholders	External stakeholders
Board of directors	Customers, consumers of critical services, or financial institution clients of a third party
Senior management	Third parties (e.g., service providers, suppliers, or partners)
Heads of business lines, especially critical service owners	User groups
Relevant internal business unit personnel (e.g., IT, human resources, compliance, and legal)	Shareholders

II.B.3 Development Roles

Development personnel typically encompass the technology professionals charged with designing, building, and maintaining solutions to solve a business problem or fulfill technology needs identified by business strategy. Development personnel may create solutions to support new products offered to customers, expand existing system capacity to support growth, or integrate systems after a merger. Larger or more complex entities may have separate personnel or departments to fulfill development roles, or varying development roles may have more granular responsibilities (e.g., user interface developers, software engineers, or mobile application developers). In smaller or less complex entities, roles may overlap, which may be acceptable if appropriate mitigating controls are in place. Third parties may help the entity with or serve in development roles.

Segregation of duties and dual controls or other compensating controls are important for development, acquisition, and maintenance activities because they help foster security and resilience. For example, segregation of development and implementation duties helps ensure that developers cannot directly alter the production environment. Inappropriate developer access may adversely affect the production environment's confidentiality, integrity, availability, or resilience. Effective dual control allows management and other personnel to identify and address problems earlier since no single person or group can make changes or modifications without appropriate approval. Without dual control, seemingly small coding, building, and patching problems can quickly escalate in scale.

[16] Refer to Cybersecurity and Infrastructure Security Agency (CISA), *CRR Supplemental Resource Guide, Vol. 3: Configuration and Change Management, Version 1.1*. Also refer to CISA, *CRR Supplemental Resource Guide, Vol. 5: Incident Management, Version 1.1*.

Typically, development deliverables are software applications or smaller modules of a larger application or system. Some larger or more complex entities, however, employ software engineering specialists to modify firmware, customize third-party-developed in-house applications, or adapt hardware products, such as automated teller machines (ATM), cash dispensers, and passbook or receipt printers to fit the entity's specific needs. Larger or more complex entities, especially those providing cloud, managed security, and payments services, often build proprietary hardware (e.g., servers or point-of-sale terminals) used in-house and by customers they service.

Staff roles that support development include the following:

- **Network architects:** Network architects generally design, select, and implement the appropriate system architecture to satisfy stakeholder requirements and achieve the desired results under given constraints. Network architects often have extensive knowledge of the entity's business plan to design a network to help the entity achieve its strategic objectives. They create plans and layouts for data communication networks and present those plans to management; explain risks; and justify recommended designs, changes, and consequences of inaction. They consider security when designing networks and systems. Once network architects define the appropriate architecture, they generally document and communicate it to the developers. Network architects upgrade hardware and software as needed to support solutions. After deployment, they may support those networks and troubleshoot any issues. When needed, network architects research new or emerging technologies to determine whether they are appropriate to implement for the entity. They evaluate network traffic to estimate growth and determine capacity management needs. Network architects may work with other IT personnel, such as network or system administrators and information systems managers, to ensure that networking needs are met. They work with third parties to manage upgrades and support the networks. Larger or more complex entities may define additional roles for enterprise architecture, including software and hardware architecture.
- **Developers:** Developers create software and the underlying systems and components to support an entity's business lines and strategic objectives, including security and resilience. They analyze users' needs and design and develop software to meet those needs, including how the systems or components will work together. Developers should have a clear understanding of user requirements and needs to ensure that deliverables meet user expectations. Developers document every aspect of a system or component as a reference for future maintenance and upgrades. They help ensure that a program continues to function normally through regular maintenance and, when necessary, recommend upgrades to keep pace with technology and security needs. They communicate with quality management personnel and testers to ensure that the system or component operates as envisioned. Developers may specialize in a particular language, platform, or business sector and may work individually in small or large teams.
- **Software engineers:** A software engineer is a type of developer who has a broad view of a project's system and software requirements and plans its scope and order of work. Software engineers may direct software developers, QA analysts, and testers.[17]
- **Hardware engineers:** Hardware engineers typically design new hardware. As part of the

[17] Refer to the U.S. Bureau of Labor Statistics' *Occupational Outlook Handbook*, "Software Developers, Quality Assurance Analysts, and Testers."

design process, hardware engineers create schematics of equipment to be built and test the hardware. They analyze test results and modify the design as needed. Hardware engineers often work with developers to ensure that hardware components work together with the latest software. Additionally, hardware engineers oversee the manufacturing process for hardware. Many hardware engineers design devices used in manufactured products that incorporate processors and other computer components and that connect to the internet (e.g., point-of-sale terminals, card readers, and ATMs).

- **Information security analysts:** Information security analysts plan and carry out security measures to protect an entity’s networks and systems. They work with developers to build security into systems and components and maintain the security aspects once developed. The role specializes in building and performing security assessments, identifying threats to systems, and ensuring that the software complies with the entity’s security standards and policies. Information security analysts should have an enterprise-wide understanding of the entity’s security architecture and interoperability of systems and components.
- **Systems analysts:** Systems analysts can also be referred to as systems architects. Systems analysts use their knowledge of the entity’s systems, procedures, and technology needs to help IT personnel design systems that operate more efficiently and support the enterprise-wide business and strategic objectives. Systems analysts work with other IT personnel to help an entity’s business leaders understand how systems and components support the entity’s strategic and business line goals and objectives. Systems analysts devise ways to add functionality to systems by researching different technologies and analyzing costs and benefits of changing, upgrading, or implementing new IT systems to help managers decide which, if any, to install. Systems analysts validate the communication and interface between systems to ensure that they continue to function. They work with other IT personnel to evaluate system capacity needs. They work with business personnel to write instruction manuals and train the systems’ end users. Additionally, systems analysts may work with other IT personnel to solve problems that arise after the initial system setup.

II.B.4 Acquisition Roles

Acquisition personnel typically procure systems or components. Larger or more complex entities may have separate individuals, teams, or departments responsible for acquisition. Acquisition personnel are often assisted by representatives from several departments, such as project management, IT, contracting, finance, information security, legal, and third-party risk management. Acquisition-related roles may be broken down further into the product or service type purchased (e.g., technology component). At smaller or less complex entities, the acquisition role often is one of multiple duties assigned to an individual (e.g., IT manager or president), which is acceptable if mitigating controls are in place. Additionally, third parties may perform the entity’s acquisition activities (e.g., cloud access security broker). Segregation of duties, dual controls, and other compensating controls are important in acquisition activities to avoid conflicts of interest (e.g., choosing third parties based on relationships versus whether the third party provides the appropriate product, best service, or most advantageous price). Depending on whether the entity chooses to develop IT solutions or procure them from third parties, acquisition roles may vary.

General responsibilities include specifying what is required, requesting proposals, receiving bids from third parties, evaluating bids and bidders, initiating contracts, and managing contractual relationships. Responsibilities also include evaluating supply chain risks related to third parties (e.g., geopolitical risk for third parties outside the United States). These responsibilities may require specialized knowledge to evaluate pricing and other aspects (e.g., contract terms or support services) of the relationship. Responsibilities are carried out throughout the entire life cycle of the third-party relationship.[18] For more information, refer to the "Acquisition" section of this booklet.

Roles that support acquisitions include the following:

- **Procurement manager:** A procurement manager can also be referred to as a third-party risk manager or business relationship manager. The procurement manager is responsible for the procurement strategy being consistent with the entity's overall strategy and risk appetite. Procurement managers are responsible for developing the entity's procurement policies, standards, and procedures to help ensure that procurement personnel avoid potential conflicts of interest or inappropriate third-party relationships. Procurement managers evaluate third parties based on the price, quality, functionality, and delivery speed of products and services. They verify that vendor processes align with the entity's requirements (e.g., security and resilience) by interviewing vendors and visiting third-party facilities. They analyze proposals, financial reports, and other information to determine reasonable costs, negotiate contracts on the entity's behalf, and arrange agreements with third parties (e.g., service-level agreements [SLA] or product delivery time frames). Procurement managers evaluate and monitor contracts to ensure that third-party management complies with the contract's terms and conditions (including compliance with applicable laws and regulations) and determine the need for changes. When necessary, procurement managers meet with entity and third-party management to discuss defective or unacceptable products or services and determine corrective action. Procurement managers maintain and review records of items bought, costs, deliveries, product performance, and inventories. They may attend meetings, trade shows, and conferences to learn about new industry trends and make contacts with potential third-party partners. Procurement managers may work with other personnel, including those in program management, IT, information security, or business lines involved in the project.
- **Third-party risk manager:** This role can also be referred to as third-party relationship manager and may also be the same person as the procurement manager. This role involves performing due diligence on third parties before procuring systems, components, or contracting for services (e.g., development or information security activities). When appropriate, the third-party risk manager performs ongoing oversight over third parties throughout the relationship's life cycle.

[18] The term ''business arrangement'' may also be used and is meant to be interpreted broadly; it is synonymous with the term ''third-party relationship.'' A third-party relationship may exist despite a lack of a contract or remuneration. Third-party relationships can include outsourced services, use of independent consultants, referral arrangements, merchant payment processing services, services provided by affiliates and subsidiaries, and joint ventures.

II.B.5 Maintenance Roles

Appropriate personnel should perform maintenance throughout the life cycle of systems and components. Some responsibilities assigned as part of maintenance include asset management (including end-of-life [EOL] and end-of-support [EOS] management), patch management, vulnerability management, and performance and capacity management. Management should understand maintenance roles and related responsibilities. Maintenance personnel should have knowledge and understanding of all relevant systems and components they are expected to operate and maintain. Both maintenance personnel and management should analyze, document, and understand the costs of maintenance (e.g., budget, time, or personnel) versus the costs of not performing maintenance (e.g., system failures, data breaches, and customer dissatisfaction). Larger or more complex entities may have separate individuals, teams, or departments charged with maintenance. At smaller or less complex entities, maintenance roles may be part of other roles or duties that an individual may have, which is acceptable if mitigating controls are in place (e.g., segregation of duties and dual controls). Additionally, third parties may perform responsibilities related to maintenance for the entity, according to contractual agreements.

IT staff are generally responsible for maintaining an entity's systems and components to ensure that they continue to operate as expected. Maintenance generally includes performing updates and applying patches, monitoring the systems' or components' status, and monitoring the servers and infrastructure that support the entity's systems and components. Maintenance roles may have more granular responsibilities (e.g., network support) depending on the complexity of the entity's IT environment. Maintenance roles should be independent from development roles to prevent developers from accessing production environments. For more information, refer to the "Maintenance" section of this booklet.

II.B.6 Other Common Development, Acquisition, and Maintenance Roles

Depending on the entity's size and complexity, the following roles are common to development, acquisition, and maintenance activities:

- **Configuration or change control board (CCB):** The CCB is "a group of qualified people with responsibility for the process of regulating and approving changes to hardware, firmware, software, and documentation throughout the development and operational life cycle of an information system."[19] The CCB typically is part of an entity's change management process, and it reviews and prioritizes change requests related to a project or production environment.
- **Testers and quality assurance analysts:** During development, acquisition, and maintenance, testers and quality assurance analysts create test plans, scenarios, and procedures for new or changed systems and components to confirm that they function as intended. Testers are responsible for testing systems, components, and controls regardless of whether they are developed internally or procured from a third party. Testers design and conduct tests to check systems and components for problems. They identify, document, and

[19] Refer to NIST Glossary.

report risks from test results and recommend steps to minimize or resolve risks or defects. They assess the usability and functionality of systems and components to identify issues, including difficulties a user might have. They can use manual or automated processes, with additional testing as necessary, and evaluate results. Testers document the results of testing and report defects or problems to developers, project teams, and other stakeholders. Once the system or component is released to production, testers run additional tests to look for errors and usability problems. They perform tests after any upgrades, maintenance, or implementation of patches and fixes on a system or component to validate the changes made. Depending on the types of tests performed (e.g., unit, security, regression, and integration), other testers may be involved at various SDLC phases, based on the testers' roles and knowledge of the system or component and its use. For more information, see the "Testing" section in this booklet.

II.B.7 Supply Chain Roles

Typically, supply chain personnel are responsible for implementing and managing systems and components throughout the entity's supply chain. Larger or more complex entities may have separate individuals, teams, or departments charged with supply chain roles and responsibilities. These personnel may include representatives from several key departments, such as development, procurement (including logistics and product delivery), maintenance, project management, IT, contracting, finance, information security, and third-party risk management.

Common internal roles may be the chief risk officer, CIO, CISO, program executive, program manager, system engineers or system security engineers, and other personnel. Common external supply chain roles may be suppliers, developers, system integrators, third-party system service providers, and other external personnel responsible for supply chain activities as noted in agreements (e.g., contracts) between the entity, third parties, and their subcontractors.[20]

II.B.8 Other Support Functions

Several lines of business and other support functions should be involved in overseeing or validating development, acquisition, and maintenance activities early in the process. One of those functions may be end-user and customer support, often referred to as the IT help desk. The end-user and customer support role works with development, acquisition, and maintenance personnel to respond to user or customer concerns. This role may also assist with resolving problems that cannot be corrected quickly. Moreover, this role provides feedback to development, acquisition, and maintenance personnel about the functionality of products and services. This role records the details of customer or user interactions and the resolution of problems. End-user and customer support staff should understand the escalation process and the personnel who can address more complex problems.

[20] For more information on common external supply chain roles, refer to NIST Special Publication (SP) 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*.

II.B.9 Audit's Role

An effective audit function provides independent, objective validation of controls and statements of assurance on the effectiveness of an entity's development, acquisition, and maintenance activities. Although auditors should not have any direct involvement in management decisions, they should raise objections if they believe the control environment is inadequate. Auditors may plan for their advisory capacity in development, acquisition, and maintenance activities by developing an understanding of the proposed system or changes. Auditors may require specialized knowledge of IT to advise on or review these activities. They should validate schedule adherence and management. Auditors should determine the effectiveness of controls in the entity's development, acquisition, and maintenance activities and recommend appropriate mitigation. They may communicate with developers regarding appropriate control standards and frameworks throughout IT projects.

Auditors do more than consult when reviewing controls in new or modified systems or components. In smaller or less complex entities, auditors typically assess an entity's development, acquisition, and maintenance activities during an "IT general controls"[21] audit. Larger or more complex entities may have a dedicated IT audit function, which generally follows a risk-based approach to prioritize the review of numerous IT processes and controls. These entities may have separate audits to assess general processes, such as development, acquisition, or maintenance, or may review these activities as part of business line, project, or other technology audits. During each SDLC phase, auditors review the internal controls, testing, and audit trails included in systems and components.

Auditors may participate in an advisory capacity in IT projects and may perform pre- and post-implementation reviews. Post-implementation reviews should occur shortly after implementing the new or revised system or components. These reviews help validate that the system or component operates as expected and provide financial statement users with relevant information in ways that justify the cost of providing it.

In larger or more complex entities with numerous IT systems and components, formal quality management or change management groups may have primary responsibility for post-implementation reviews. In such cases, auditors may choose not to perform a separate review; however, they may participate in validating minimum test criteria and evaluating the results of reviews.

Auditors' validation activities regarding IT projects or change management may include reviews for the following:

- Project or changes were authorized.
- Project or changes were subjected to a risk-impact assessment.
- Project or changes were handled effectively and formally documented, especially when it was an emergency change.

[21] While "IT general controls" is a typical name for all-encompassing IT audits, entities may use various names for these reviews.

- Project or changes were prioritized among all other IT projects or changes in a manner that is effective for the entity.
- Project or changes were tracked by a formal process.

For more information, refer to the *FFIEC IT Handbook's* "Audit" booklet.

III RISK MANAGEMENT OF DEVELOPMENT, ACQUISITION, AND MAINTENANCE

Action Summary

Management should implement continual risk management processes within the entity's development, acquisition, and maintenance activities to identify, measure, monitor, and control reasonably foreseeable internal and external risks and threats, including those that could result in unauthorized disclosure, misuse, alteration, or destruction of customer information or customer information systems.

Examiners should review the following:

- Policies, standards, and procedures for identifying, measuring, mitigating, monitoring, and reporting risks related to development, acquisition, and maintenance activities.
- Documented processes and metrics used to measure the level of risk.
- Detailed documentation of processes used to review, accept, and document risks that management cannot mitigate or transfer.
- Documentation of risk assessment processes to identify key risks at the onset of IT projects and throughout their life cycles.
- Risk assessments that highlight internal and external risks related to development, acquisition, and maintenance activities.
- Documentation of ongoing processes and reports to monitor and communicate risk, including emerging risks related to development, acquisition, and maintenance activities.
- Reports to stakeholders that are timely, accurate, and include clear, relevant metrics.

Management should identify, measure, monitor, and control reasonably foreseeable internal and external risks and threats, including those that could result in unauthorized disclosure, misuse, alteration, or destruction of customer information or customer information systems.[22] While the requirement is relative to risks to sensitive customer information, it is a prudent practice relative to any sensitive information type. Additionally, management should consider risks that could result in a severe disruption or material compromise to critical service delivery. It is also important to consider known risks related to developed or acquired products and systems,[23] and

[22] Refer to 15 USC 6801 and 6805(b), "Financial Services Modernization Act of 1999" (Gramm–Leach–Bliley Act or GLBA), further implemented by FFIEC members as follows: FDIC: 12 CFR 364, appendix B, "Interagency Guidelines Establishing Information Security Standards"; FRB: Regulation H, 12 CFR 208, appendix D-2, "Interagency Guidelines Establishing Information Security Standards"; and Regulation Y, 12 CFR 225, appendix F, "Interagency Guidelines Establishing Information Security Standards"; NCUA: 12 CFR 748, appendix A, "Guidelines for Safeguarding Member Information"; and OCC: 12 CFR 30, appendix B, "Interagency Guidelines Establishing Information Security Standards." Additionally, refer to Federal Trade Commission (FTC) 16 CFR 314, "Standards for Safeguarding Customer Information."

[23] Developed or acquired products and services may include hardware (e.g., servers, laptops), software (e.g., application programming interfaces), or services (e.g., banking-as-a-service).

any known potential misuses. IT risk management activities should include policies, standards, and procedures for identifying, measuring, mitigating, monitoring, and reporting risks related to development, acquisition, and maintenance activities. Additionally, management should consider using threat models to assess security policies, standards, and procedures. While there are many widely used threat modeling methodologies, it is important for entity personnel to consider implementing a model commensurate with the threats in the entity's overall environment.

Risk management activities include identification of risks associated with development, acquisition, maintenance, and associated IT projects, including the risks associated with not initiating a project.[24] Examples of risk management activities to consider include the following:

- Measuring each activity's level of risk.
- Mitigating risks to an acceptable, board-approved level of residual risk.
- Monitoring risks related to development, acquisition, and maintenance activities.
- Reporting updates to senior management and the board, as necessary.

At times, management may choose to transfer some of the risk, such as by purchasing insurance (e.g., errors and omissions or cyber) to address the potential for unknown errors in source code. When management cannot mitigate or transfer risk, there should be a process to review, accept, and document the risk and its acceptance by the board. For an acquisition, management and the board may decline to move forward if the risk level exceeds the board's risk appetite (e.g., purchase of a product that uses an amount of open-source code above approved levels). Management and the board should regularly review and approve risk acceptance decisions consistent with the entity's governance structure. For more information about risk management, refer to the *FFIEC IT Handbook's* "Management" booklet.

III.A Risk Identification

Risk identification involves identifying and documenting risks that may affect development, acquisition, and maintenance activities. Effective risk identification is a continuous process that includes iterative reassessment of risks. Identifying key risks related to development, acquisition, and maintenance activities gives management opportunities to implement appropriate controls and mitigate risks. Identifying key risks at the onset of an IT project allows for early assessment of risks, risk effects, and risk mitigation options. For example, a code review helps developers find errors before migrating code between the test and the production environments. Ideally, risk identification and assessments involve stakeholders with business process knowledge who may be affected by the development, acquisition, or maintenance activities or the related IT projects. As always, information security concerns should also be considered.

Risk identification highlights internal and external risks related to development, acquisition, and maintenance activities. Examples of internal risks include insider threats (e.g., trusted employees who use authorized access in an unauthorized manner for personal gain, entity harm, or errors and omissions) and scheduling difficulties (e.g., loss of knowledgeable personnel or

[24] There is a strong relationship between the phases in the SDLC, the SCRM life cycle (both discussed in this booklet), and the third-party risk management life cycle, referenced in the *Interagency Guidance on Third-Party Relationships: Risk Management*.

unavailability of skilled personnel). Examples of external risks include those arising from third-party relationships, industry threats and changes (e.g., new products or services, regulatory changes, or new cyber threats), and natural disasters. Additionally, financial technology (fintech)[25] firms and other third-party relationships may not be subject to the same regulatory requirements as the entity. Management is responsible for holding the third party accountable to the entity’s standards (e.g., compliance with regulations and adherence to security and resilience standards) and the board’s risk appetite.

III.B Risk Measurement

Management should implement effective standards to measure risk in the entity’s development, acquisition, and maintenance activities. Risk measurement should be an ongoing process and commensurate with the size and complexity of an entity’s activities. As the complexity of an entity’s activities and processes increases, management may benefit from using automated tools to help measure risk. Effective risk measurement processes include management setting and monitoring objective standards for performance. Examples of risk measurement related to development, acquisition, and maintenance include the following:

- Comparisons of estimated versus actual completion dates for IT project deliverables.
- Number of open issues related to projects and the length of time issues remain open.
- Percentage of occasions when management needed to back code out of production.
- Cost overruns greater than the entity’s established limits.
- Service levels below those defined in contracts and agreements.
- Number of unapplied patches or patches applied beyond the policy time frames.
- Policy exceptions.

III.C Risk Monitoring and Reporting

Effective management monitors and reports risks related to development, acquisition, and maintenance activities. Examples of risks monitored by management include project management (e.g., milestone date shifts, negative test results, and major requirements changes), cybersecurity, supply chain, and emerging technology. Reports of these risks should be timely and accurate; include clear, relevant metrics (e.g., trends related to the number of open issues or code back-outs); and note deviations from policies, standards, and procedures. Reports should be distributed to appropriate stakeholders, including the board when necessary. Management should communicate technical aspects that are easily understood (e.g., explaining acronyms or technical jargon) when reporting to the entity’s board and relate the IT issues to the entity’s business concerns. Effective monitoring and reporting inform management about problems and performance issues in a timely manner, so that management can minimize the potential for an increase in the levels of risk associated with identified issues. Monitoring and reporting should enable all stakeholders to understand the risks and agree with the proposed courses of action.

[25] For purposes of this booklet, “fintech” refers to using technology in novel ways to provide financial services.

III.D Controlling or Mitigating Risk

Risk mitigation relies on identification and monitoring of risk in conjunction with the board’s risk appetite. These limits should assist in limiting exposure to the various risks associated with the entity’s IT-related activities. Implementation of effective operational controls to address these risks are described throughout this booklet. One example of risk mitigation is implementing code reviews during development to detect and correct errors earlier in the development process that might cause the system or component to fail. Effective management adopts repeatable control processes to ensure that risks are consistently addressed across the entity over time.

IV COMMON DEVELOPMENT, ACQUISITION, AND MAINTENANCE RISK TOPICS

Action Summary

Management should implement effective risk mitigation throughout development, acquisition, and maintenance activities regardless of the phase of the project in the life cycle and agnostic as to the type of technology. Specific risks and controls should be considered depending on management's chosen solution.

Examiners should review the following:

- Systems, components, and services, including related contracts and licenses, throughout the supply chain for appropriate risk identification and mitigation.
- Security controls (e.g., secure coding requirements and baseline configuration use) used to harden systems and components.
- Inventory of all systems and components (e.g., open-source, proprietary, application programming interfaces [API], and container images and registries), including related licenses, and data as part of IT asset management (ITAM).
- Access, authentication, and authorization controls for systems, components, data, and related documentation throughout the supply chain to ensure appropriate security.
- Segregation of duties in development, acquisition, and maintenance activities.
- Process of selecting and implementing methodologies to enable effective management and control of development, acquisition, or maintenance projects and alignment with entity objectives.
- Operating parameters (e.g., timing, speed, throughput, and data validation) for systems and components to determine performance.
- Activity logs related to systems, components, and data to identify operating risks.
- Monitoring processes of development and maintenance activity for identification of anomalies and unauthorized access or modification to systems, components, and data.
- Reporting processes for decision-making and measuring the level of project success.
- Training on development, acquisition, and maintenance concepts (e.g., methodologies); effectiveness of training; and capability of personnel to implement concepts learned.
- Documentation of internally developed systems and components and externally procured products and services to effectively operate and maintain the systems and components.
- Evaluation process (e.g., post-implementation review, stakeholder interview, problem documentation and resolution, cost-benefit analysis, and reports to senior management) for development, acquisition, and maintenance projects.

IT systems are designed, built, and implemented to achieve strategic goals and business objectives. While there are risks specific to each of the development, acquisition, and maintenance activities, certain risk considerations are common to all three. Common development, acquisition, and maintenance risk topics are discussed in the following subsections of this booklet:

- "Open-Source"
- "Commercial-off-the-Shelf"
- "Licenses, Agreements, and Copyright Protection"
- "Secure Development"
- "Data"
- "Secure Operating Environments"
- "Microservices"
- "Containers"
- "Application Programming Interfaces"
- "Quality Management"
- "Documentation Standards"
- "Post-Implementation Review"
- "IT Project Management"
- "System Development Life Cycle"
- "Supply Chain Considerations"

IV.A Open-Source

Open-source software is software released under a license that allows the software and its source code to be accessed, used, modified, and shared by anyone.[26] Effective management identifies and mitigates risks related to the use of open-source components (e.g., open-source software components and hardware components). For example, components may be based on poorly written code, or the developer may not have properly addressed security concerns. Open-source vulnerabilities may be more common depending on the underlying code's quality because the overall review process for open-source is less transparent and more difficult to validate. If the original developer makes updates, they may not be pushed to the entity. In that case, effective management would request updates from the original open-source developer to address vulnerabilities and manually deploy updates. Alternatively, when the original developer does not make updates to open-source systems and components, effective management may engage in-house or third-party software development personnel to update the software.

The Open Worldwide Application Security Project (OWASP) recommends the following component analysis[27] strategies to reduce risks associated with the use of open-source components:[28]

[26] For more information, refer to NIST Directive S 6106.01, "Open Source Code."

[27] OWASP defines "component analysis" as "the process of identifying potential areas of risk from the use of third-party and open-source software and hardware components."

[28] For more information, refer to OWASP, *Component Analysis*.

- Evaluating the type of framework and libraries to reduce architectural changes, regressions (or defects), and code rewrites.
- Identifying and analyzing the purpose of each component to reveal the existence of components with duplicate or similar functionality.
- Evaluating the number of third-party and open-source components in a project because the higher the number, the more difficult it is for development teams to maintain large sets of components over time.
- Using private central repositories[29] or public repositories' code-signing and verification tools to minimize risk of threats[30] related to public repositories.
- Identifying a component's provenance[31] to help ensure knowledge of authorship of software components, manufacturers, suppliers, software repositories, and country of origin.
- Maintaining accurate formulation[32] and pedigree[33] information for source code that is readily available, modifiable, and redistributable.
- Identifying the license for a given component and whether it allows certain types of usage, contains distribution requirements or limitations, or requires specific actions if the component is modified.
- Aggregating the risk of all direct, transitive, runtime, and environmental dependencies providing a holistic view of inherited risk.[34]

[29] A "repository" is defined as "a place, room, or container where something is deposited or stored." For purposes of IT, repositories may store requirements, policies, processes, data, software libraries, projects, configurations, performance goals, and applications, with the potential of supporting both software development and operations management.

[30] Some of the threats against public repositories include typosquatting (i.e., naming a component to take advantage of common misspelling); organization/group abuse (i.e., pretending to be a public person or entity and abusing the perceived trust); malware through transfer (i.e., leveraging weak or absent code-signing requirements to spread malware through the transfer of an open-source project from one maintainer to another); and cross-build injection (i.e., abusing dependency resolution schemes and weak infrastructure controls to inject malicious components in place of safe ones).

[31] For purposes of this discussion, the identification of a component's provenance refers to the chronology of the origin, development, ownership, location, and changes to a system or system component and associated data. It can include personnel and processes used to interact with or make modifications to the system, component, or associated data.

[32] OWASP describes "formulation" as "how components were built…and a comprehensive list of parallel and sequential steps that were taken to build, test, and deliver a component."

[33] NIST Glossary defines "pedigree" as "the validation of the composition and provenance of technologies, products, and services is referred to as the pedigree. For microelectronics, this includes material composition of components. For software this includes the composition of open source and proprietary code, including the version of the component at a given point in time. Pedigrees increase the assurance that the claims suppliers assert about the internal composition and provenance of the products, services, and technologies they provide are valid."

[34] Inherited risks are derived from an application's transitive dependency (i.e., when an application or component has a direct dependency on another component and that component then has a dependency on another component) or direct dependency on every component. Transitive dependencies have their own risk that is inherited by every component and application that relies on them. Refer to OWASP, *Component Analysis*.

- Evaluating the health of an open-source project through quality controls and metrics (e.g., frequency of code changes and patches), community engagement (e.g., user base and involvement), and vulnerability analysis.
- Including external services (such as those relied on for functionality) in the overall inventory of components.

During software evaluation, a review of the open-source developer's formal documentation helps management determine potential software maintenance risks after software selection. Open-source documentation may be less comprehensive when compared with documentation for proprietary systems and components. Effective management proactively maintains updated application and user documentation, especially when code changes are made.

For more information, refer to the *FFIEC IT Handbook's* "Information Security" and "Architecture, Infrastructure, and Operations" booklets.

IV.B Commercial-off-the-Shelf

Commercial off-the-shelf (COTS) systems are systems, components, or solutions that are ready-made and available for sale, lease, or license to the general public. COTS is also referred to as off-the-shelf. Entities typically use COTS for operating systems (OS), applications, network infrastructure, servers, desktops, laptops, and mobile devices. Entities often choose to use COTS because it generally is less costly than developing systems and components in-house and requires fewer resources to maintain. COTS solutions generally provide a default set functionality. For example, a default installation of a server OS may include mail, web, and file-sharing services whether or not the solution requires those functions. This is because COTS providers may be developing their products to serve a wide variety of customers, not just those in the financial industry. Therefore, unnecessary services may be present in COTS and represent potential security weaknesses (e.g., unnecessary services creating a wider attack surface), which should be mitigated (e.g., turned off or patched). Protection against those risks begins when the systems or components are constructed and installed through a process referred to as "hardening." Management should consult third-party providers and vendors regarding recommended security controls to harden the COTS solution. Hardening can involve employing recommendations in a documented installation procedure and generally includes the following:

- Changing default passwords.
- Installing the most secure, up-to-date versions of applications.
- Configuring security settings as appropriate.
- Enabling logging.
- Configuring backups to increase resilience.

A COTS solution may be integrated with existing systems and components to provide additional or enhanced business services to internal and external customers. Management should mitigate any risks related to integration (e.g., lack of interoperability or integration between systems and components). If modifications are made, the process to develop and implement them may be time-consuming and challenging. These modifications may result in additional configuration changes for COTS, or interface changes between COTS and other systems and components (e.g., APIs, middleware, and cloud). These changes may then require additional resources and personnel to perform. For more information, refer to the *FFIEC IT Handbook's* "Information Security" and "Outsourcing Technology Services" booklets.

IV.C Licenses, Agreements, and Copyright Protection

Systems and components are often licensed, not purchased. Typically, licenses do not give a client (sometimes referred to as a licensee) ownership rights over the licensed product. The licensor[35] provides licenses to clients that grant certain rights to use the system or components. The licensor typically grants rights to a client for a specific duration and may charge annual and other fees for system or component use. An effective license and agreement review process at the entity is important to ensure that terms are clearly defined and understood prior to engagement. Before negotiating licenses, management should accurately assess current and future needs and ensure that licenses will continue to meet the entity's needs.

The defined scope of a license is a key issue in evaluating and, when feasible, negotiating the terms of a licensing agreement. When reviewing a license, management should confirm that it clearly states whether system or component usage is exclusive, the number of user licenses, and whether there are any time, place, manner, or other types of limitations with the system or component's use. Effective management reviews licenses to determine whether they allow backup copies related to mission-critical systems or components on which the entity may need to rely for disaster recovery or business continuity purposes. The entity may need to negotiate with licensors to ensure the ability to backup hardware and software if backups are not provided for in the license. In difficult license agreement negotiations, including when an entity has limited negotiating power, management should understand any resulting limitations and consequent risks. Possible actions that an entity might take in such circumstances include determining whether the product or service can still meet the entity's needs, whether the product or service would result in increased risk to the entity, and whether residual risks are acceptable. If the license agreement is unacceptable for the entity, management may consider other approaches, such as employing other third parties or conducting the activity in-house. In certain circumstances, entities may gain an advantage by negotiating license agreements as a group with other entities (e.g., user groups and banking associations).[36]

If the entity plans to provide the systems or components to other related entities (e.g., subsidiaries or contractors), management should include those entities as users in the licenses. If management is responsible for developing systems and components and providing them to other entities or its subsidiaries, management should implement appropriate licenses for the systems or components developed.

Knowledge of license expiration dates is crucial for the planning and management of license renewal. If there is no license expiration or the license renews automatically (i.e., a perpetual

[35] A "licensor" is defined as "the person who gives or grants a license."

[36] For more information, refer to *Interagency Guidance on Third-Party Relationships: Risk Management*. Specific agency references to this document include (FDIC) FIL 29-2023, (FRB) SR Letter 23-4, and (OCC) Bulletin 2023-17.

license), the license agreement should explicitly outline those terms. Failure to specify a fixed term or termination date does not automatically provide the entity with a perpetual license. Management should specify in its agreement the appropriate time periods for all licenses and the minimum amount of notice required for license termination. Effective licensing and ITAM processes include tracking license time frames.

Effective management periodically reviews system and component licenses to compare all installed licensed systems and components with the respective license terms. This allows management to detect unauthorized installations and validate appropriate license use. If there is a discrepancy between the total number of licenses allowed by the agreement and the installed software, management should address the inappropriate usage. Audit personnel should consider this when developing the scope of their reviews.

There are several other types of agreements management should consider and review in addition to licenses. For example, a third party may offer maintenance agreements, which outline available maintenance services (e.g., provision of new versions, releases, or updates). Another type of agreement is a development agreement, which outlines terms for the development of systems or components. This agreement may be internal or with a third party. If the agreement involves a third party, the entity may not retain ownership rights even if the entity paid to have the system or component developed.

A license or agreement should address the availability and cost of system and component updates and modifications (i.e., maintenance). When drafting agreements, effective management determines whether a third-party vendor provides access to source or object code ("code"). Management should have the vendor's permission and participation for modifications to the code for systems and components. Unauthorized modifications to the code may void maintenance agreements.

A license or agreement should direct third-party vendors to deliver appropriate documentation. This should include application and user documentation. Management should ensure that the agreement specifies that updated application and user documentation will be provided when any changes are made to procured systems or components.

IV.C.1 Software Licenses

Effective ITAM includes software license oversight. Software licenses have varying provisions and requirements for ownership and usage. It is important to consider license provisions and all consequences for violating provisions. In more critical instances (i.e., critical to providing business services), entities may engage specialized legal expertise to understand and mitigate software license risks. If an entity develops software and licenses that software to others, management should be aware of the liabilities that come with licensing activities, including considerations related to software maintenance, integration, compatibility, and fraud. Tracking and managing the usage and implementation of different types of licenses for which an entity paid helps ensure that the entity is operating within the licenses' specific requirements. There are several types of software licenses, including free and open-source software licenses and proprietary licenses.

IV.C.1(a) *Free and Open-Source Software Licenses*

There are two types of free and open-source software (FOSS) licenses:

- **Public domain (and equivalent) license:** Public domain software is software not protected by copyright laws of any nation that may be freely used without permission of or payment to the creator and that carries no creator warranties.[37] This reduces creator liability relative to the software's use. Software creators may renounce software ownership and any usage limitations to ensure that the software is able to be used by anyone regardless of jurisdiction license requirements and to limit their liability.[38]
- **Open-source license:** Typically, open-source software used by entities has some type of attached license. Companies often license open-source software through a third party that makes software use and maintenance easier. Just as with proprietary software licenses, it is important to understand the license provisions such as who owns the software source code, and any restrictions on what can be done with the software. Examples of open-source software license subtypes are lesser general public licenses, permissive licenses, general public licenses, and copyleft licenses.[39]

FOSS is typically acquired and used in an as-is manner. An entity may not be aware of the developer(s) in public domain FOSS. There may or may not be updates made available for the software, and changes made to the software may be made by other individuals besides the developer. Risks associated with FOSS include the potential for unidentified or unpatched vulnerabilities and development changes by unknown parties. An accurate inventory of FOSS should be maintained as part of an effective ITAM process, and management should understand and abide by the requirements of FOSS licenses. Additionally, some licenses require users to contribute software additions or enhancements, or they may prohibit the use for commercial purpose or profit. Management should be aware of what information is shared when contributing an enhancement to the FOSS development community.

IV.C.1(b) *Proprietary Software Licenses*

In proprietary software licenses, the software publisher retains software ownership, and the entity and end users agree to accept and abide by the terms of the software licenses. A licensee and its employees may only use the software as allowed by the license and typically cannot copy, modify, or distribute the software and its underlying code. An end user license agreement (EULA) provides specifics regarding provisions or restrictions. EULAs typically apply to for-profit open-source software and non-open-source software, and they are often used for commercial purposes. Some examples of proprietary licenses include noncommercial use-only

[37] Refer to NIST Glossary.

[38] For more information, refer to public-domain equivalent licenses, such as Creative Commons' CC0.

[39] Copyleft is a general licensing method for making a program free while requiring all modified and extended versions of the program to be free as well. There are strong and weak copyleft licenses. Refer to Department of Defense (DOD), *DoD Open Source Software FAQ.*

licenses, proprietary licenses, and trade secret licenses. Regardless of the types of licenses encountered, they often cover the following:

- Ownership rights of the software.
- Permission to use the software and usage rights.
- Number of users allowed to use the software.
- Inclusions or add-ons (e.g., support, maintenance, and software upgrades).
- Warranty terms.
- Installation location and frequency.
- Permissions and limitations regarding copying, modifying, and distributing software and the underlying code.
- Copyrights.
- Terms for license termination.
- Software performance guarantees.
- Penalties and fees for noncompliance.
- Length of the agreement and its terms.

For more information on ITAM, which includes management of software licenses, refer to the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations" booklet.

IV.C.2 Hardware Licenses

Entities identify and manage hardware licenses in ITAM programs to ensure that their hardware use is consistent with agreements between the manufacturer, any third party, and the entity. Entities may access components (e.g., circuit boards, servers, switches, tokens, universal serial buses, mobile devices, service ports, and cameras) in solution development. Hardware licenses, like software licenses, should identify any limitations on hardware use.

If an entity licenses COTS hardware, the hardware may come as is and any alterations or modifications may void the warranty in the license. If the hardware is proprietary, items such as warranties, modifications, and maintenance should be included in the license agreement. If an entity builds hardware and then licenses that hardware to other entities, effective management maintains awareness of the liabilities that come with licensing activities, including considerations related to hardware maintenance, integration, compatibility, and fraud.

For more information on ITAM, which includes management of hardware and software licenses, refer to the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations" booklet.

IV.C.3 Copyright Protection

Copyright laws[40] protect proprietary as well as open-source software. Examples of elements covered by copyright protections generally include the following:

[40] Refer to U.S. Copyright Office, *Copyright Law of the United States (Title 17)*.

- **Reproduction:** The copyright owner maintains rights in any reproduction of the original work.
- **Adaptation:** The copyright owner has rights to adaptations based on the original work.
- **Publication:** The copyright owner has the right to distribute the work to the public.
- **Performance:** The copyright owner has the right to perform the work publicly.
- **Display:** The copyright owner has the right to display the work publicly.

Unlicensed system or component use or licensing agreement violations expose entities to litigation. Common challenges in deploying software, consistent with licenses and copyrights, include keeping track of deployed instances when the entity is large and decentralized, keeping track of deployed instances when deployments are made to virtual environments and are not automatically tracked, instances being shared when the license does not provide for sharing, and license limits on concurrent system or component use.

Measures that management may employ to protect against copyright violations include obtaining a site license that authorizes system or component use at all of the entity's locations, informing users of the rules governing site licenses, and acquiring an automated program that scans for unauthorized system or component use or copyright violations. While these measures may help identify copyright violations, the best control mechanism is a well-communicated corporate policy that management enforces and auditors validate for compliance.

IV.D Secure Development

Secure development is an approach to creating products and services that incorporates security into every phase of the product's or service's development and use. Secure software development includes practices "to reduce the number of vulnerabilities in released software, to reduce the potential impact of the exploitation of undetected or unaddressed vulnerabilities, and to address the root causes of vulnerabilities to prevent recurrences."[41] While this booklet primarily focuses on the secure development of software, there are also techniques for securely developing hardware.[42] System, component, and service development, acquisition, and maintenance inherently presents security risks. This is true whether the system, component, or service is developed internally or acquired.

When management outsources development to third parties, it should monitor the third parties' conformance to contract requirements, Information Security Standards,[43] and other legal and

[41] Refer to NIST SP 800-218, *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities*.

[42] For more information on mitigation of hardware development risks, refer to DOD's *Securing Defense-Critical Supply Chains*.

[43] Refer to 15 USC 6801 and 6805(b), GLBA, further implemented by FFIEC members as follows: FDIC: 12 CFR 364, appendix B, "Interagency Guidelines Establishing Information Security Standards," and 12 CFR 364, supplement A to appendix B, "Interagency Guidance on Response Programs for Unauthorized Access to Customer Information and Customer Notice"; FRB: Regulation H, 12 CFR 208, appendix D-2, "Interagency Guidelines Establishing Information Security Standards," and Regulation Y, 12 CFR 225, appendix F, "Interagency Guidelines Establishing Information Security Standards"; NCUA: 12 CFR 748, appendix A, "Guidelines for Safeguarding

regulatory requirements. A third party providing development services should maintain policies, standards, and procedures that promote secure development consistent with the entity's requirements and the board's risk appetite. When acquiring systems and components, management should evaluate the third party's secure coding standards (e.g., through independent certification or audit) and consider related supply chain risks.[44] It is also important that third-party system or component developers use secure coding standards, that its use is independently verified, and that management reviews the verification. This is important to verify when products are initially purchased and to review periodically as products are updated and patched. Furthermore, system maintenance should consider security impacts of integration with legacy systems and components. For more information, refer to the *FFIEC IT Handbook's* "Outsourcing Technology Services" and "Information Security" booklets.

General software quality management practices contribute to secure systems and components. For example, when thorough testing results in functional flaws being corrected, security vulnerabilities can also be identified and corrected. There are quality assurance tests that can specifically identify security flaws such as fuzz testing[45] and penetration testing. There are also quality management tools such as secure code reviews and source code security analyzers[46] that can systematically and efficiently identify security vulnerabilities and other flaws for remediation. Entities that use these practices and tools are more likely to produce secure systems and components than those that do not.

Manual or automated code reviews help identify vulnerabilities. Manual code reviews[47] are performed by entity personnel or a third party and may be less costly than an automated code review.[48] However, limitations on reviewers' knowledge or experience may result in unidentified issues, such as syntax errors and code vulnerabilities due to human error. Additionally, manual reviews may not be effective for extensive volumes of testing. Automated code reviews, on the other hand, may be more sophisticated, can reduce human error, and generally provide more timely identification of weaknesses. These tools are scalable and can be continuously updated as technology and vulnerabilities evolve. Entities review software code that has been implemented to mitigate the risk that vulnerabilities may be introduced through routine updates and patches. If

Member Information"; and OCC: 12 CFR 30, appendix B, "Interagency Guidelines Establishing Information Security Standards."

[44] For more information on supply chain risks affecting development, refer to the "Development" and "Supply Chain Considerations" sections of this booklet.

[45] Fuzz testing refers to a black box software testing technique, which basically consists of finding implementation bugs using malformed or semi-malformed data injection in an automated fashion by tools referred to as "fuzzers," which are programs or scripts that submit some combination of inputs to the test target to reveal how it responds. For more information, refer to OWASP's *Fuzzing*.

[46] According to NIST, "Source code security analyzers examine source code to detect and report weaknesses that can lead to security vulnerabilities."

[47] Manual code review is the process of personnel reviewing the source code line by line to identify possible vulnerabilities.

[48] Refer to NIST Internal Report (IR) 8397, *Guidelines on Minimum Standards for Developer Verification of Software*.

an automated code review process is implemented, management should ensure that the embedded rules of the code review tools are appropriately configured and used. Management selects code review types that are effective for the entity's project and available resources, and management validates that the code review process meets its needs.

System vulnerability scanning is another tool for mitigating security risks during development.[49] Scanning software is implemented in development environments so that vulnerabilities are identified and remediated before they are exposed to users and exploited by malicious actors. Requirements for system vulnerability scanning can be included in contracts and ancillary agreements if the underlying system is critical enough. Entities also continue to scan systems for vulnerabilities once systems are deployed and being maintained to mitigate the risk that vulnerabilities are introduced as the system is upgraded. Refer to *FFIEC IT Handbook's* "Information Security" and "Architecture, Infrastructure, and Operations" booklets.

IV.E Data

Mitigating the risk of data compromise is important in addition to controlling software and system development. To mitigate the risk of data compromise, entity personnel establish and maintain data inventories that include data characteristics such as storage location, criticality, and sensitivity. They also implement and maintain controls that protect the data consistent with their characteristics. For example, an entity may implement a policy that highly sensitive data will not be used in development and test environments. Instead, an entity may create synthetic data to use in development and testing that is a proxy for the actual, highly sensitive data. Another control example is documenting system data use so that security engineers can implement controls that protect data throughout process execution.[50] For more information, refer to the *FFIEC IT Handbook's* "Information Security" and "Architecture, Infrastructure, and Operations" booklets.

IV.F Secure Operating Environments

Management should maintain a secure operating environment throughout development, acquisition, and maintenance activities whether entity employees are operating the system or a third party's employees are operating the system. Entities use various controls to secure operating environments, such as configuration management, access control, and business continuity planning and testing. Effective application of formal policies addressing the least privilege principle help control access to systems, components, and data. An effective entity establishes documented policies to communicate and implement operating environment controls and provides for periodic audits for validation of the controls to determine whether systems and

[49] Like network port and service identification, vulnerability scanning identifies hosts and host attributes (e.g., operating systems, applications, or open ports), but it also attempts to identify vulnerabilities rather than relying on human interpretation of the scanning results. Refer to NIST SP 800-115, *Technical Guide to Information Security Testing and Assessment*.

[50] NIST states in SP 800-37, rev. 2, *Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy*, that "such documentation includes, for example, data flow diagrams, entity relationship diagrams, database schemas, and data dictionaries."

components are functioning as intended. Additional operating environment control examples are listed below:

- Identifying personnel allowed to access development, test, and production environments.
- Determining under which conditions they may access systems, components, and data within production environments.
- Maintaining authorization procedures for granting and verifying access.
- Determining how long users require access.
- Segregating operating environments (e.g., development, test, QA, and production) and specifying access controls for each operating environment, while accounting for new business functions.
- Ensuring resilience when developing, acquiring, or maintaining systems, components, services, and data.
- Maintaining control over remote access (e.g., timing, location, personnel, and multifactor authentication [MFA]).
- Providing appropriate controls over access to and by specific devices.
- Maintaining business line and IT representation when determining security and resilience measures.
- Maintaining ITAM processes for current and planned assets.

For more information, refer to the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations," "Outsourcing Technology Services," and "Information Security" booklets.

IV.G Microservices

Microservices refers to an architectural and organizational approach to building systems that involves much smaller components than historical approaches and requires attention to mitigate risks that could be material. NIST defines "microservices" as "a set of containers that work together to compose an application." This includes a series of interconnected containers or applications working in coordination (referred to as "orchestration"). Microservices-based architectures and applications are prevalent in cloud-based systems but can also be run in non-cloud-based virtualized environments.

Traditionally developed, or monolithic,[51] applications comprise multiple lines of code developed as a single application. To work properly, these applications reside on a single server or virtual machine. Any updates rely on an update to the entire application. By contrast, microservices-based applications consist of small, self-contained (i.e., containers), independent services that use APIs to communicate and work together. Each microservice has a singular purpose (e.g., messaging, calculations, or database access). Microservices are generally technologically heterogeneous and can be written in different languages, use different development platforms, and be spread across multiple servers or virtual machines. Because microservices architecture has independent containers working together to create an application, updates can be applied to any one independent container without having to update the entire application. Additionally,

[51] A monolithic application (or monolith) refers to a single unit, as opposed to a microservice architecture, which comprises smaller, independently deployable services.

each microservice can be reused to create new applications. These properties of microservices facilitate scalability and a faster, more agile application development process.

During acquisition or development of systems using microservices, effective control practices are used to mitigate the risk of systems malfunctioning and risk of security incidents. An example of an effective control is deploying and using highly secure and stable service registries for tracking the services included in a system. A microservice has two broad functions:[52]

- Business logic, which implements the business functionalities, computations, and service composition or integration logic.
- Network processes, which manage the interservice communication mechanisms and are built on top of the underlying OS-level network stack.

NIST provides four drivers underlying the design principles of microservices:[53]

- Each microservice must be managed, replicated, scaled, upgraded, and deployed independently of other microservices.
- Each microservice must have a single function and operate in a bounded context (i.e., have limited responsibility and dependence on other services).
- All microservices should be designed for constant failure and recovery, and therefore must be as stateless[54] as possible.
- One should reuse existing trusted services (e.g., databases, caches, and directories) for state management.

Additionally, NIST provides a list of the following design principles for microservices:

- **Autonomy:** Ability to operate independently as a self-contained entity that delivers all of the functions of an operations stack.
- **Loose coupling:** Minimal dependency between services so the change in one service does not require a change in another service.
- **Reusability:** Use of a service in different ways for different purposes.
- **Composability:** Ability to be assembled easily and configured to meet specific needs.
- **Fault tolerance:** Property of a system that allows proper operation even if components fail.
- **Discoverability:** Ability to be located and identified using discovery services.
- **Alignment of APIs with business processes:** Aligning microservices, APIs, and business processes during the design phase allows for easier changes to the services when business processes change.

[52] Refer to NIST SP 800-204A, *Building Secure Microservices-Based Applications Using Service-Mesh Architecture*.

[53] Refer to NIST SP 800-204, *Security Strategies for Microservices-Based Application Systems*.

[54] Refer to NIST Glossary.

Because microservices can be replicated or deployed across servers, a microservice architecture generally has a directory for microservices to publish their locations, also referred to as a service registry. NIST provides the following strategies for service registry configuration:[55]

- Service registry capabilities should be provided through servers that are either dedicated or part of a service mesh[56] architecture.
- Service registry services should be in a network that has been configured with certain quality of service parameters to ensure its availability and resilience.
- Communication between an application service and a service registry should occur through a secure communication protocol (e.g., hypertext transfer protocol secure or transport layer security).[57]
- Service registries should be validated to ensure that only legitimate services perform the registration, refresh operations, and query the database to discover services.
- Service registration and service deregistration functions should follow the principles of bounded context and loose coupling. Management should set parameters for the processes of registration to and deregistration from the service registry.[58]
- If a third-party registration pattern[59] is implemented, registration and deregistration should only take place after the performance of a health check[60] on the application service.
- A distributed service registry should be deployed for large microservices applications, and data consistency should be maintained among multiple service registry instances.[61]

[55] Ibid.

[56] NIST SP 800-204A, *Building Secure Microservices-Based Applications Using Service-Mesh Architecture*, defines "service mesh" as "a distributed computing middleware that optimizes communications between application services."

[57] NIST defines "transport layer security" as "an authentication and security protocol widely implemented in browsers and web servers." For more information, refer to NIST SP 800-63-3, *Digital Identity Guidelines*.

[58] As noted in NIST SP 800-204, *Security Strategies for Microservices-Based Application Systems*, the application service should not have tight coupling with an infrastructure service, such as a service registry service, and service self-registration and deregistration patterns should be avoided. When an application service crashes or is running but unable to handle requests, its inability to perform deregistration affects the integrity of the whole process. Therefore, registration and deregistration of an application service should be enabled using a third-party registration pattern, and the application service should be restricted to querying the service registry for service location information as described under the client-side discovery pattern.

[59] For purposes of this discussion, a pattern refers to repeatable processes to assist in the standardized design, monitoring, and maintenance of microservices.

[60] A health check or health testing is testing in an implementation immediately before or during normal operation to determine that the implementation continues to perform as implemented and as validated. Refer to Law Insider's Dictionary.

[61] Refer to NIST SP 800-204, *Security Strategies for Microservices-Based Application Systems*, for more information on distributed service registry, which involves several service registry agents cooperating to provide controlled access to resources. Distribution of the registry agents results in improved availability, higher concurrency, better response times to user queries, and enhanced flexibility.

Microservices are exposed to similar threats as web-based applications (e.g., cross-site scripting or injection attacks). Controls to prevent or mitigate these attacks should be included in the microservices code.[62] Microservices can have security, reliability, and latency issues. Having multiple microservices can increase the entity's attack surface. Effective management evaluates implementation options that meet the entity's security requirements. Management of microservices-based architectures should include monitoring services, which may be running on different servers or written in different languages. Management should consider the following monitoring controls:

- Monitoring at the gateway and service level.
- Implementing a centralized dashboard to display the status of multiple services and network segments.
- Creating a baseline and implementing intrusion detection to provide alerts on deviations from the baseline.

To preserve resilience when developing and using microservices, NIST recommends the following control examples:[63]

- Load balancing by having multiple instances of the same service and evenly distributing the load of those instances.
- Implementing circuit breaking by setting a threshold for failed responses from a microservice. When the threshold is exceeded (i.e., the circuit breaker is tripped), requests should not be forwarded to that microservice. The goal is to prevent cascading failures and allow time to investigate and address the issue.
- Throttling or limiting the rate of requests to a microservice.
- Redirecting requests to new versions of the microservice.[64]
- Limiting traffic to new versions of a microservice until management can validate the correctness of a response and understand the performance.[65]

For more information on the development and use of microservices, refer to the "Development" section of this booklet.

[62] For more information on specific controls, refer to "OWASP API Security Project."

[63] Refer to NIST SP 800-204A, *Building Secure Microservices-Based Applications Using Service-Mesh Architecture*.

[64] This is referred to as blue/green deployments. When a new version of a microservice is deployed, requests from customers using the old version can be redirected to the new version using the API gateway that can be programmed to maintain awareness of the locations of both versions. For more information, refer to NIST SP800-204A, *Building Secure Microservices-Based Applications Using Service-Mesh Architecture*.

[65] This process is referred to as "canary releases," in which only a limited amount of traffic is initially sent to a new version of a microservice because the correctness of its response or performance metric under all operating scenarios is not fully known. Once sufficient data are gathered about its operating characteristics, the requests can be proxied to the new version of the microservice. For more information, refer to NIST SP 800-204A, *Building Secure Microservices-Based Applications Using Service-Mesh Architecture*.

IV.H Containers

A container is a method for packaging and securely running an application in a virtualized environment. Containers enable applications to be portable, reusable, and automatable in terms of their creation, use, and destruction. Containers provide software developers the ability to segment applications in separate environments to support well-defined functionality. This allows the entity to coordinate the development, testing, implementation, and operation of applications in a more consistent, precise, and efficient manner. Containers enable virtualized segmentation, which allows for increased security and transparency but may require additional configuration to support effective integration. In addition to the use of containers in the production environment, it is beneficial to leverage containers in the development and test environments, which helps to promote consistency.

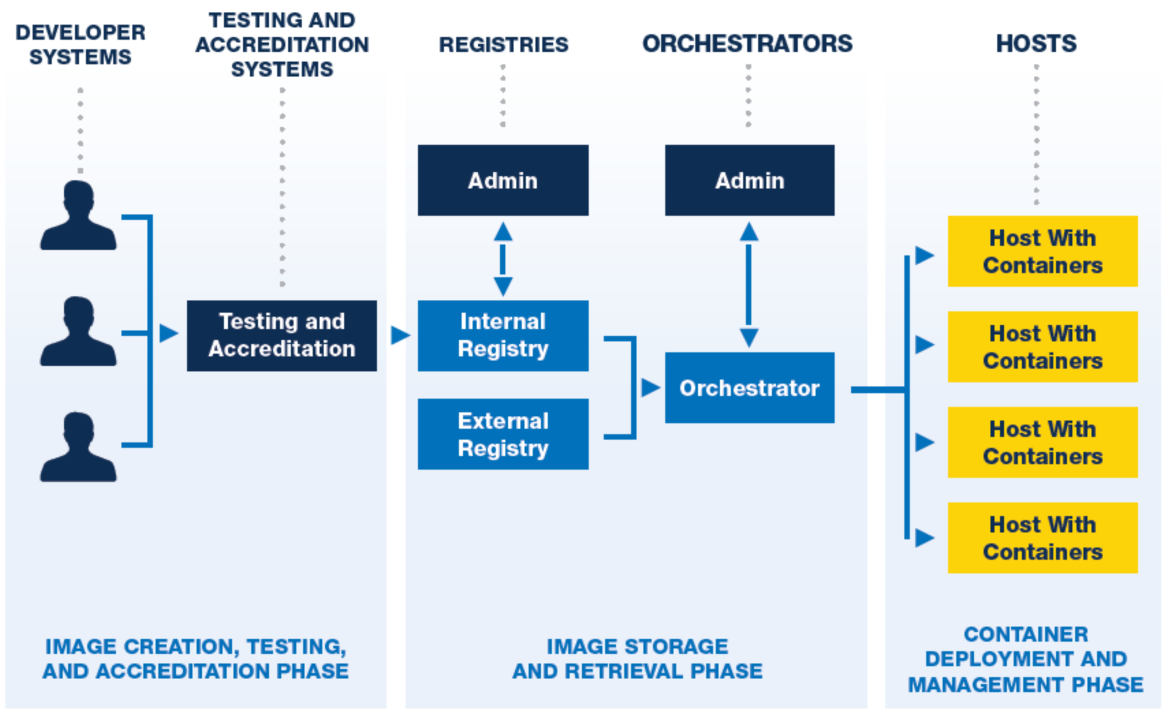
Figure 1 depicts the following five foundational architectural tiers and components and the three life cycle phases for container development and technology architecture.[66] The foundational architectural tiers and components are as follows:

- Developer systems, in the image creation, testing, and accreditation phase, generate container images and send them for testing and accreditation.
- Testing and accreditation systems, in the image creation, testing, and accreditation phase, validate and verify the contents of container images, sign container images, and send container images to the registry.
- Registries (internal and external), in the image storage and retrieval phase, store images and distribute container images to the orchestrator on request.
- Orchestrators facilitate the conversion of container images into containers during the image storage and retrieval phase and deploy containers to hosts in the container deployment and management phase.
- Hosts run and stop containers as directed by the orchestrator in the container deployment and management phase.

The three phases noted in figure 1 are (1) image creation, testing, and accreditation; (2) image storage and retrieval; and (3) container deployment and management for container development and technology architecture.

[66] For more information, refer to NIST SP 800-190, *Application Container Security Guide*.

Figure 1: Container Technology Architecture Tiers and Components and Life Cycle Phases



When setting up containers, it is important to consider data isolation. Data may be stored separately from configuration information and software (i.e., as separate building blocks). When data and configuration information are stored separately from the containerized software, the software, data, or configuration information can be created, used, and destroyed without consequence to the other building blocks in the container. Management may use persistent storage[67] for ongoing access to containerized applications, data, and configuration information. The aspects of isolation and persistent storage allow for new versions of containers to be created and used efficiently, without losing previously used data. When a new version of the software or containerized image is executed or initialized, connections to the persistent storage need to be restored along with any authorization and access rights considerations.

Container images are typically designed to be portable across machines and environments so that an image created in a development lab can be easily moved to a test lab for evaluation, then copied into a production environment to run without needing to make any modifications.[68] This deployment process (i.e., copying the image into production) results in a version of the application that is running and available to respond to requests. According to NIST, "When an image is deployed into a container, the image itself is not changed, but instead a copy of it is

[67] *Merriam-Webster* defines "persistent" as "continuing without change in function or structure." For purposes of this discussion regarding storage in containers, "persistent storage" refers to the long-term access to application information beyond the container's lifetime.

[68] Refer to NIST SP 800-190, *Application Container Security Guide*.

placed in the container and transitioned from being a dormant set of [application] code to a running instance of the [application]."[69]

The use of containers is not without risks; therefore, during container development, management should consider the risks and countermeasures provided in NIST 800-190,[70] such as those in table 1.

Table 1: Container Risks and Countermeasures

Risks	Countermeasures
Image risks	
Image vulnerabilities[71]	Consider containers and images when selecting or using vulnerability management tools.
Image configuration defects	Adopt tools and processes to validate and enforce compliance with secure configuration best practices, such as implementing privileged user controls and validation of image configuration settings.[72]
Embedded malware	Monitor images for embedded malware by monitoring for malware signature sets and detecting behavioral patterns and anomalies.
Clear text secrets[73]	Use orchestrators and APIs to manage security of access to and provision of secrets to specific containers that require them based on predefined settings.
Untrusted images	Maintain a set of trusted images and registries. Only use images from this set to run in the environment, thus mitigating the risk of untrusted or malicious components being deployed. Consider processes to ensure that images and registries are secure, trusted, and periodically validated and that the hosts they reside on are from these approved lists.[74]
Registry risks	
Insecure connections to registries	Configure development tools, orchestrators, and container runtimes to connect only to registries over encrypted channels. The goal is to ensure that all data pushed to and pulled from a registry occur between trusted end points and are encrypted in transit.
Stale images in registries	Reduce registries to minimize unsafe and vulnerable images that should no longer be used. Access images using unchanging names that specify discrete versions of images to be used to ensure that specific and uncorrupted images are deployed as part of each job.[75]

[69] Ibid.

[70] Ibid.

[71] Container image vulnerabilities may not be detected by traditional vulnerability management tools.

[72] For more information, refer to NIST SP 800-190, *Application Container Security Guide*.

[73] Many apps require secrets to enable secure communication between components. The risk is that if someone steals the image, that person can learn the secrets, such as username and password, to connect to a backend database, connection strings, and private keys. Refer to NIST SP 800-190, *Application Container Security Guide*.

[74] For more information, refer to NIST SP 800-190, *Application Container Security Guide*.

[75] Another option is using a "latest" tag for images and referencing this tag in deployment automation, but this may not always be effective. Regardless of whether an organization chooses to use discrete names or to use a "latest" tag, it is critical that processes exist to ensure that the most recent unique name or the images tagged "latest" represent

Risks	**Countermeasures**
Insufficient authentication and authorization restrictions	Any access to registries that contain proprietary or sensitive images should involve authentication. Gaining write-access privileges to a registry should entail appropriate authentication to ensure that only images from trusted entities can be added to the registry. Audit all write-access privileges to registries and log any read actions for sensitive images.[76]
Orchestrator Risks	
Access considerations, such as unbounded administrator access[77]	Orchestrators should use a least-privilege access model in which users are only granted the ability to perform the specific actions on the specific hosts, containers, and images that their job roles require. Test team members should have limited or no access to containers used in production.[78]
Unauthorized access	Tightly control access to cluster-wide administrative accounts, as these accounts provide the ability to affect all resources in the environment. Use appropriate authentication methods (e.g., MFA and single sign-on when applicable). Use tools for encrypting data used with containers that allow the data to be accessed properly from containers regardless of the node on which they are running.[79]
Orchestrator node trust[80]	Configure orchestration platforms to provide features that create a secure environment for all the applications they run. Maintaining a secure-by-default posture is particularly important with third-party deployments of the entity's containers. Orchestrators should be able to perform the following: introduce nodes securely to the cluster,[81] maintain a persistent identity for the nodes throughout their life cycle, and maintain an accurate inventory of nodes and their connectivity states. Orchestration platforms should be resilient (i.e., a compromise of one node should not compromise the overall security of the cluster).[82]
Poorly separated inter-container network traffic[83]	Configure orchestrators to separate network traffic into discrete (e.g., internal versus external applications) virtual networks by sensitivity level, and

the most up-to-date versions. For more information, refer to NIST SP 800-190, *Application Container Security Guide*.

[76] For more information, refer to NIST SP 800-190, *Application Container Security Guide*.

[77] A single orchestrator having administrator access. Access should be limited according to need. For example, applications may be managed by different teams with differing sensitivity levels. As many orchestrators were designed with the assumption that user interaction would be with administrators, there is the potential for unauthorized access; therefore, authorization and access concerns are relevant. Refer to NIST SP 800-190, *Application Container Security Guide*.

[78] For more information, refer to NIST SP 800-190, *Application Container Security Guide*.

[79] For more information, refer to NIST SP 800-190, *Application Container Security Guide*, and the *FFIEC IT Handbook's* "Information Security" booklet.

[80] The orchestrator is the most foundational node. Weak orchestrator configurations can expose the orchestrator and other container technology components to increased risk. Refer to SP NIST 800-190, *Application Container Security Guide*.

[81] An example of a cluster is a set of nodes that run containerized.

[82] A compromised node should have the ability to be isolated and removed from the cluster without disrupting or degrading overall cluster operations. For more information, refer to NIST SP 800-190, *Application Container Security Guide*.

[83] If a virtual overlay network is used to manage container traffic, security and management tools may not have clear visibility into the traffic. If traffic has varying sensitivity levels on the same network, more sensitive data may be susceptible to attack due to expanded access to the network users with lesser privileges. For example, if the public-

Risks	Countermeasures
	communication between the two should occur through a small number of well-defined interfaces.[84]
Mixing of workload sensitivity levels on the same host[85]	Configure orchestrators to isolate deployments to specific sets of hosts by sensitivity levels. Implement rules that prevent high-sensitivity workloads from being placed on the same host as those running lower-sensitivity workloads. Segmenting containers by purpose, sensitivity, and threat posture provides additional defense-in-depth.[86]
Container Risks	
General container runtime-related risks[87]	Use tools or processes that continuously assess configuration settings across the environment and actively enforce them. Use segmentation to provide control and isolation over containers to minimize unauthorized access and loss of data. Run containers with the default secure computing profiles[88] provided by their runtime. Consider using additional profiles for higher risk applications. Use tools to identify vulnerabilities in the container runtimes deployed to upgrade any instances at risk and to ensure that orchestrators only allow deployments to properly maintained container runtimes.
Unbounded network access from containers[89]	Employ a combination of network monitoring and filtering tools (e.g., application-aware tools)[90] to mitigate the risk from inter-container traffic across networks of differing sensitivity levels.

facing website is compromised, attackers may be able to use shared networks to attack an internal app using sensitive information. Refer to NIST SP 800-190, *Application Container Security Guide*.

[84] For more information, refer to NIST SP 800-190, *Application Container Security Guide*.

[85] An orchestrator may place a container running a public-facing web server on the same host as one processing sensitive data, because that host happens to have available resources at the time of deployment, as the orchestrator manages scale and density of traffic. In the case of a critical vulnerability in the web server, a container processing sensitive data may be at greater risk of compromise due to the greater access to the host of the container. Refer to NIST SP 800-190, *Application Container Security Guide*.

[86] Concepts such as application tiering and network and host segmentation should be taken into consideration when planning app deployments. By segmenting containers in this manner, it is much more difficult for an attacker who compromises one of the segments to expand that compromise to other segments. In larger-scale environments with hundreds of hosts and thousands of containers, this segmentation is automated to be practical. Common orchestration tools can help management in the segmentation process. For more information, refer to NIST SP 800-190, *Application Container Security Guide*.

[87] Container runtimes include such risks as (1) insecure container runtime configurations and (2) capability for containers running in privileged mode to access host configurations, potentially compromising the host and all other containers on it.

[88] Secure computing profiles can be used to limit the system-level capabilities allocated to containers at runtime. For more information, refer to NIST SP 800-190, *Application Container Security Guide*.

[89] By default, in most container runtimes, individual containers can access each other and the host OS over the network. If a container is compromised and acting maliciously, allowing this network traffic may expose other resources in the environment to risk. Tools and operational processes that are not container-aware are not able to inspect this traffic or determine whether it represents a threat. Refer to NIST SP 800-190, *Application Container Security Guide*.

[90] Application-aware tools provide the ability of a system to recognize and classify applications passing through it. These tools allow systems to make decisions based on an application, its features, and the content it’s carrying. These tools should be able to not just see the inter-container traffic but also dynamically generate the rules used to

Risks	Countermeasures
Rogue containers[91]	Maintain separate environments for development, test, and production with specific controls (e.g., role-based access control) for container deployment to mitigate the potential for rogue containers. There should be a clear audit trail process during container development to give management information that associates any container build or modification to specific users.
Application vulnerabilities, such as risks inherent in the general use of applications	Implement additional container-aware tools[92] using behavioral learning (i.e., heuristics) and appropriate security profiles to detect anomalies and events, such as writing to unexpected locations and file types, sending traffic to unexpected network destinations, and storing or executing malware. Additionally, containers should be run with their root file systems in read-only mode to monitor for compromise, to isolate tampering to these specific locations, and separate them from the rest of the application.
Host OS Risks	
Large attack surface[93]	Minimize unnecessary functionality by using OSs designed to host only containers and have other services and functionality disabled to reduce the attack surface. Additionally, host OSs should have read-only file systems and have hardening practices implemented by default. Regularly scan and update the container runtime and lower-level components (e.g., kernel) in a timely manner.
Shared kernel[94]	Use separate, dedicated hosts for containerized and noncontainerized workloads. Isolating the workloads enables the application of appropriate safeguards to containers and avoids affecting noncontainerized workloads.
Improper user access rights, such as allowing users to log on directly[95] to hosts to manage containers	Employ least privilege access. Audit all authentication to the OS, monitor all login anomalies, and log any privileged user activities to identify anomalous (and potentially unauthorized) access patterns.

filter this traffic based on the specific characteristics of the apps running in the containers. For more information, refer to NIST SP 800-190, Application Container Security Guide.

[91] Rogue containers are unplanned or unsanctioned containers in an environment. They may be used in development builds or test images and may pose additional risk to the organization, especially when they persist in the environment without the awareness of development teams and security administrators. Refer to NIST SP 800-190, Application Container Security Guide.

[92] Container-aware tools provide the ability to monitor the container environment and provide precise detection of anomalous and malicious activity within it. For more information, refer to NIST SP 800-190, Application Container Security Guide.

[93] The larger the attack surface is, the better the odds are that an attacker can find and access a vulnerability, leading to a compromise of the host OS and the containers running on top of it. Refer to NIST SP 800-190, Application Container Security Guide.

[94] The use of a shared kernel results in a larger attack surface as the level of isolation provided by container runtimes is not as high as that provided by hypervisors. Runtime container security means vetting all activities in the container application environment, from analysis of container and host activity to monitoring the protocols and payloads of network connections. Refer to NIST SP 800-190, Application Container Security Guide.

[95] When users log on directly to the host with elevated privileges to manage specific containers, there is the possibility of accessing and affecting other containers on that host. Refer to NIST SP 800-190, Application Container Security Guide.

Risks	Countermeasures
Host OS component vulnerabilities[96]	Use appropriate tools provided either by the OS vendor or other trusted organizations to regularly check for and apply updates to all software components used in the OS to mitigate potential vulnerabilities. This is particularly important for the kernel and container runtime components as newer releases of these components often add additional security protections and capabilities beyond simply correcting vulnerabilities. Deploy application components and their dependencies[97] in a container to identify or prevent anomalies and configuration changes, promote resilience, and allow management to reduce its attack surface, prevent persistent storage of data on the host, promote statelessness[98] on the host, and reduce application-level host dependencies.
Host OS file system tampering[99]	Run containers with the minimal set of file system permissions required. NIST recommends that containers should not be able to make sensitive directories available on a host's file system, especially those directories containing configuration settings for the OS. Monitor what directories are placed, made available, and accessed on the file system.

IV.I Application Programming Interfaces

An API is software code that allows two or more different programs to communicate with each other. APIs are an important part of various applications, including web, mobile, and software-as-a-service (SaaS) applications. APIs often link microservices and may be public, private, or a collaboration between customers, partners, and unaffiliated third parties to share information and allow their software to function together. Another function of APIs is to facilitate communication between containers. For more information on APIs and how they fit with other elements of an entity's IT environment, refer to the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations" booklet.

When developing, acquiring, and maintaining APIs and related services, there are several considerations.[100] Examples include the following:

- Aligning the API with business processes, particularly when designing a microservice.
- Iterating the interface definition with developers before the service is used so that the resulting API serves as a contract between clients and services.

[96] All host Oss, even container-specific ones, provide foundational system elements, such as access and authentication. As a foundational element in container technology, host OS vulnerabilities can affect everything running on the host, including containers and apps. Refer to NIST SP 800-190, *Application Container Security Guide*.

[97] In IT infrastructure, dependencies are relationships of reliance in and among infrastructure assets and systems that must be maintained for those systems to operate and provide services. For more information, refer to CISA's "Infrastructure Dependency Primer."

[98] According to NIST Glossary, statelessness or quality of being "stateless" refers to "a data representation or a process that is self-contained and does not depend on any external data store."

[99] A change to a sensitive system file on the host OS could affect the stability and security of the host and all other containers running on it. Refer to NIST SP 800-190, *Application Container Security Guide*.

[100] For more information on developing, acquiring, and maintaining APIs, refer to NIST SP 800-204, *Security Strategies for Microservices-Based Application Systems*.

- Determining and implementing appropriate security related to APIs.[101]

Risks related to APIs include the following:

- Inappropriate access controls.
- Inadequate user authentication.
- Excessive data exposure.
- Lack of resources for processing access requests, including rate limiting problems.
- Allowing modification of object properties.[102]
- Misconfigured security.
- Injection flaws.
- Lack of appropriate ITAM and documentation leading to maintenance weaknesses and exposed end points.
- Lack of appropriate logging and monitoring.
- Lack of integration with the entity's incident response program.

IV.I.1 API Gateway

Understanding how APIs can act as a gateway to microservices is important. Unlike a legacy application where the endpoint may be only a single server, a microservices-based application consists of multiple endpoints. Therefore, a single-entry point (i.e., gateway) for all clients to access multiple component microservices of the application would allow for improved access. Another situation in which an API gateway may be deployed is as a front-end-to-backend[103] connector, as in the case of legacy enterprise software. This occurs when an entity migrates from a legacy enterprise application by gradually replacing its components with independent microservices over time. Direct communication between clients and multiple end points often results in excessive point-to-point connections; therefore, the API gateway's primary function is to route inbound requests to the correct downstream services to minimize the number of connections. All client requests first go through the API gateway, which then routes requests to the appropriate microservice. The API gateway often handles a request by invoking multiple microservices and aggregating the results. In some instances, the API gateway may be used as part of a back end for front-end services, which enables support for clients with different communication vectors (e.g., browser or mobile device), and then requests are divided into multiple gateways. Many API gateway use cases can support APIs written in different languages.[104]

[101] For more information on API security, refer to the *FFIEC IT Handbook's* "Information Security" and "Architecture, Infrastructure, and Operations" booklets. "OWASP API Security Project" also discusses API security issues and controls.

[102] Object properties provide a simple association between name and value. All properties have a name, and value is one of the attributes linked with the property, which defines the access granted to the property.

[103] The front end of a system involves the parts that users see and interact with, whereas the system's back end includes the logic, data, and structure.

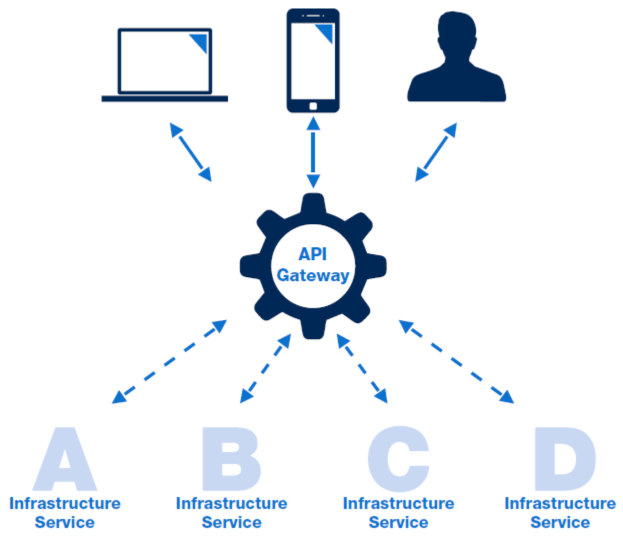
[104] Examples of different languages include Java, JavaScript, and Jolie.

Because the API gateway is the entry point for microservices, management should configure it with the necessary infrastructure services.[105] In addition to its main service of handling requests, an API gateway also helps with the following infrastructure services:

- Service discovery.
- Authentication and access control.
- Load balancing.
- Caching.
- Providing custom APIs for each type of client.
- Performing application-aware health checks.
- Service monitoring.
- Attack detection.
- Attack response.
- Security logging and monitoring.
- Circuit breaking.

These additional infrastructure services can be created or implemented in the code that uses the API gateway. Figure 2 provides a simple illustration of the API gateway location during a request. It shows that the API gateway routes requests from various sources (e.g., laptops, mobile devices, and users) to the various infrastructure services.

Figure 2: Location and Interaction of API Gateway



Microgateways are API gateways used to define and enforce customized policies for microservices-based applications. If used, they should be protected through service-specific security policies. Microgateways are usually implemented as containers and should contain policies for application requests and responses. When policies and their enforcement are implemented as a container, they are unchanging; therefore, they provide a degree of protection

[105] Refer to NIST SP 800-204, *Security Strategies for Microservices-Based Application Systems*.

against accidental and unintended modifications. Modifications could result in security breaches or conflicts. In other words, microgateways implemented as containers may help prevent breaches and conflicts because any security policy updates require the microgateway's redeployment. It is essential that the microgateway deployed for a microservice instance communicate with service registry and monitoring modules, allowing them to keep track of the operational status of the microservice that the microgateway is designed to protect.

When used with microservices, an API gateway or microgateway may serve to implement the following core features and functions:

- **Optimized end point:** The API gateway simplifies the process by providing aggregation, which results in fewer requests and responses. The API gateway can serve as a public interface for multiple individual APIs and their requests. Additionally, it performs the necessary protocol translation for client requests.
- **Circuit breaker:** Developers can set a threshold in the API gateway for microservice failed responses to requests and cut off requests when the failure is above the threshold. This avoids the possibility of a cascaded failure and allows management time to analyze logs, implement necessary fixes, and push necessary updates to address the failing instance of the microservice.
- **Load balancing:** When there are multiple instances of the same service, the load can be evenly distributed across the instances to avoid delayed responses or service crashes due to overload on any one instance.
- **Rate limiting (throttling):** The rate of requests going into a microservice should be limited to ensure continued availability of service for all clients.
- **Blue/green deployments:**[106] When a new version of a microservice is deployed, requests from customers using the old version can be redirected to the new version because the API gateway can be programmed to be aware of the locations of both versions.
- **Canary releases:** Only a limited amount of traffic is initially sent to a new version of a microservice to verify the accuracy of its response and performance. Once sufficient data demonstrate that the microservice is operating as designed, then the API gateway can direct the requests to the new version of the microservice.

IV.I.2 API Risk Mitigation

Mitigation strategies for common API risks include the following:[107]

- **Inappropriate access controls:** API endpoints are vulnerable to unauthorized access as attackers can manipulate the specific object ID or other identifier that is sent with the request. Management should apply extra layers of security, such as for broken object-level

[106] This type of deployment provides an application release model when user traffic is progressively transferred from a prior version (i.e., old or blue) of an app or microservice to a nearly identical new (i.e., green) release. Both the blue and green versions are running in production at the same time.

[107] For more information, refer to "OWASP API Security Project."

authorization,[108] beyond those for standard end-point security, and implement appropriate authorization checks at the object level.
- **Inadequate user authentication:** Authentication processes are not functioning correctly or are inappropriate for the way the API is used (e.g., designed for internet of things [IoT] versus a web application) or without considering potential vectors of attack. Management should validate the user authentication process for the use of the API and consider MFA.[109]
- **Security misconfiguration:** Security settings may be missing or inappropriate for the current expected functionality of the API, giving an attacker the opportunity to gain unauthorized access or knowledge of the system. Attackers may use automated tools to detect and exploit these flaws, leading to data or device compromise. Management should implement appropriate security, patch and update APIs, and disable unnecessary functions.
- **Excessive data exposure:** Developers try to implement APIs generically without addressing the sensitivity of the exposed data. Automated tools can have difficulty detecting this type of vulnerability because they cannot differentiate between legitimate data provided from the API and inappropriately provided sensitive data. This problem creates the potential for unauthorized data access. Developers should review the responses from the API to ensure that a request contains only legitimate data and to understand how the requestor will use the data. Developers should consider creating and implementing a scenario-based response process (including error codes) to validate that data returned by the APIs are appropriate for the scenario.
- **Lack of resources and data limiting:** API requests consume resources such as network, central processing unit (CPU), memory, and storage. When APIs do not implement rate limiting or limits are not set properly, it may lead to an API being unresponsive or unavailable (i.e., denial of service). Management should appropriately set all API processing parameters (e.g., execution timeouts, maximum allocatable memory, number of file descriptors, number of processes, request size, number of requests per client or resource, and number of records per page to return in a single request response). If even one of the limits is missing or set inappropriately, an API may be vulnerable.
- **Broken function level authorization:** Implementing proper authorization checks can be difficult because applications can contain many types of roles or groups and complex user hierarchy (e.g., subusers and users with more than one role). Because these checks are difficult to implement, what appears to be a legitimate API call or request may not be authorized, allowing unauthorized access to functionality, particularly administrative functions. Management should ensure that all access is denied by default and employ specific role-based permissions for users to gain access to functions. Appropriate personnel should review API end point security.
- **Mass assignment:**[110] Users can update multiple object properties without appropriate authorization. The design of APIs provides for easier exploitation of mass assignment because they expose the underlying implementation of the application along with the

[108] Object-level authorization involves access-control tools applied at the code level to validate appropriate access to objects (e.g., fields such as names or values) by only authorized users.

[109] For more information, refer to FFIEC's *Authentication and Access to Financial Institution Services and Systems* on multifactor authentication.

[110] The MITRE Corporation defines "mass assignment" as "a feature that allows simultaneous modification of multiple object attributes."

properties' names. Management should deny access to properties unnecessary for the user's role and only allow access when necessary.

- **Injection:** Client and external system data may not be validated and filtered appropriately, allowing malicious data to be introduced into systems and components via the API. Injection can lead to information disclosure, data loss, denial of service, or complete host takeover. Management should validate and filter all data coming to an API with appropriate parameters.
- **Improper asset management:** Management may not properly maintain API and related asset inventories or may not update and patch API security in a timely manner. This may either allow unauthorized access to sensitive data or compromise the server through older, unpatched API versions connected to the same database. Management should maintain current, accurate inventories of APIs and related assets, including services,[111] and regularly review them. Management should maintain logging of API activity (e.g., authentication, errors, redirects, rate limiting, and end points, including their parameters, requests, and responses). Management should consider security measures, such as API security firewalls, segregation of production and nonproduction data, and timely removal of older API versions.
- **Insufficient logging and monitoring:** Without sufficient, timely logging and monitoring, including API activity, it is difficult to identify patterns of potential malicious activity (e.g., threats and potential security weaknesses), which may lead to compromise of systems, components, and data. Management should perform continuous logging and monitoring of relevant API activity (e.g., failed authentication attempts, denied access, and input validation errors). Automated security information and event management tools with appropriately configured dashboards may help management monitor and manage API-related logging. To maintain confidentiality and integrity of sensitive data in activity logs, effective management secures logs and ensures only appropriate access to them.

With the volume and potential significance of API-related vulnerabilities, it is critical to incorporate planning for the exploitation of these vulnerabilities during API development and the entity's incident response plan. Neglecting to do so can result in the compromise of confidentiality, integrity, availability, and resilience as well as inadequate or untimely response during an incident. For more information on security and other topics related to APIs, refer to the *FFIEC IT Handbook's* "Information Security" and "Architecture, Infrastructure, and Operations" booklets.

IV.J Methodologies

A methodology[112] is used to facilitate the development, acquisition, or maintenance of a system or component project. Management should establish appropriate methodologies to enable effective management and control of system and component development, acquisition, and maintenance activities. Management can apply various methodologies, or a hybrid, to different projects based on stakeholder needs and project objectives. Methodologies can be a combination of strategies, design philosophies, and accepted practices designed as a structured, organized set

[111] For more information, refer to the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations" booklet.

[112] *Merriam-Webster* defines "methodology" as "a particular procedure or set of procedures."

of rules that allow a project team to work together effectively. If the entity employs multiple methodologies, the chosen methodology should be defined for each IT project.

Effective management aligns implementation of any methodology with the overall strategic and business objectives. Planning, education, and communication are important elements to help ensure that critical resources are available when needed for IT projects. Management should promote effective planning through IT project management and support training for developers, quality management members, testers, and maintenance personnel. Entity staff should be proficient and qualified in the selected methodologies. Common examples used in the financial industry include waterfall and agile; however, examiners may encounter others.

IV.J.1 Waterfall

The waterfall methodology was the first widely used model. It is considered a traditional model for development, acquisition, and maintenance projects. This methodology provides a logical, rigid approach to managing a sequential series of tasks or steps in which each phase is completed before the next phase of the project can begin. This methodology may be useful for stable, less complex projects with well-defined objectives and requirements that are not prone to change. A primary benefit of this methodology is its ease of use and implementation. Additionally, because of the sequential progression of the project, coding issues are identified earlier, and project status is more easily measured than in other methodologies. However, it may be difficult or cost-prohibitive to return to a previous project phase to make corrections or changes in requirements requested by stakeholders. Other drawbacks include excessive time to produce a product, stakeholder difficulty in defining the desired end product, and difficulty making changes once started.

The waterfall methodology is a multiphase approach that begins with a requirements phase when stakeholders define the project requirements and most of the communication between stakeholders occurs. Once the first phase is completed, a product or service can then be designed, developed, tested, and implemented. This methodology allows only minimal stakeholder changes or iterations, and a final product is not provided until the implementation. Moreover, this methodology progresses through other phases until the final phase, which is the maintenance phase, during which any issues are corrected.

IV.J.2 Agile

An alternative to traditional project management methods (also known as predictive or waterfall management) is the use of adaptive or agile management methods. Incremental and iterative development principles form the foundation of agile methodologies.[113] Such principles involve breaking down a project into smaller, manageable modules that can be designed, developed, vetted, and installed in repeated cycles allowing for additional features, adjustments, and corrections with each iteration. With incremental and iterative development, management does

[113] The agile development methodologies represent a family of development types rather than a single methodology. Organizations exploring agile methodologies are working to define core values and guiding principles for enabling high-performing teams in implementing and executing with agility. These methodologies emphasize close communication and collaboration between the project team and stakeholders.

not need to identify all requirements at the beginning of the project, as changes may be added during later iterations. Agile projects are different from waterfall projects because they rely on (1) defining requirements before and during execution; (2) delivering frequent, iterative, and incremental products (i.e., not always fully completed); and (3) welcoming changes at any point in the process.[114] Incremental and iterative development may be used for larger projects in which the final product's requirements are well-defined, but details may change or need enhancement during the process.

There are different models of incremental and iterative development (e.g., iterative, spiral,[115] and prototyping modeling[116]), but they follow the main principle of a defined repetitive process. An agile methodology is an incremental and iterative development methodology that relies heavily on stakeholder interaction throughout the development process. During each phase, a deliverable is produced for the stakeholders, who provide feedback. The benefits of an agile methodology include earlier availability of a version of the deliverable and improved stakeholder satisfaction due to the regular feedback. A primary drawback of an agile methodology is the additional resource requirements (e.g., time, personnel, and funding) needed to implement the additional iterations.

Generally, an agile methodology involves dividing a project into smaller modules or "sprints." Each sprint represents an independent project, with phases (e.g., planning, requirements, design, coding, testing, and documentation) similar to those of other development methodologies. In some cases, sprints may become part of a larger project. The project is evaluated at the end of each sprint, and adjustments are implemented as appropriate. An agile methodology is designed to anticipate change at regular intervals based on feedback from stakeholders, such as customers and business analysts, and derive business value from the feedback.

When applied effectively, the overall business goal of an agile methodology is to minimize risk of introducing defects, prevent cost overruns, and quickly resolve changes in scope before there is a significant impact to the project. An agile methodology aims to satisfy stakeholder requirements by providing rapid, continuous deliveries of useful systems and components, delivered in short intervals. It aims to release working elements of the systems and components at the end of each iteration, even if the overall product is incomplete. The feedback is used to make changes to the existing deliverable and the overall project plan.

Cross-functional teams representing various business lines are created to facilitate development. The diverse representation of these teams promotes consideration of multiple perspectives, improving visibility during the development process. The result of each iteration should be a

[114] NIST Advanced Manufacturing Series 100-40, *Agile for Model-Based-Standards Development*.

[115] In the spiral model, the project iteratively goes through specific development phases (e.g., identify, design, build, and evaluate) with refinement in each iteration (or spiral) until development is complete. For more information, refer to National Aeronautics and Space Administration, *A Software Development Simulation Model of a Spiral Process*.

[116] Prototyping modeling involves developing a product with the look, feel, and limited functionality of the product under development. It can be used in conjunction with other models allowing developers to build prototypes during development. The prototype is built, tested, and adjusted until a working model is developed and can be produced as a final product or deployed throughout a system.

functional deliverable for the customer. Real-time feedback[117] from customers or stakeholders is a key aspect of an agile methodology.

An agile methodology employs a flatter organizational structure to allow iterative delivery of systems or components. A common role supporting an agile methodology is the scrum master,[118] which is similar to the traditional project manager role. However, unlike the project manager, a scrum master supports and serves the product owner and the development team by helping to remove obstacles to project deliverables.

An agile methodology may create unique challenges in documentation of development; therefore, the entity's documentation requirements should be clearly defined in the entity's project management requirements. Effective documentation relies on real-time communication between the development team and stakeholders, including end users. In any methodology, a lack of clarity or poor communication can negatively affect project deliverables and timelines (e.g., delays, scope creep, costs, and quality). This may have increased significance in an agile methodology because multiple parts of the overall project in process may be affected, leading to a greater impact than if only one segment of a project in process is affected.

Individual agile methodologies[119] may differ in their specific processes; however, they are based on some common principles.[120] However, agile principles could be at odds with security principles. For example, the agile principle of speed to delivery could cause developers to leave out important security controls that would take longer to include but be invisible to the end user. To mitigate the risk of security flaws in an agile project, leaders can establish security standards that must be met before a solution is deployed for production use.

In prototyping models of systems and components (e.g., rapid application development), developers could bypass established change controls, leading to security and integrity risks. Following established control procedures when using these models is as important as when using any other development methodology type.

[117] Feedback loops from customers and stakeholders may take several forms (e.g., emails, meetings) for communicating expectations and needed changes.

[118] NIST defines a "scrum master" as "the role performing the management of the iteration and increments." A scrum is an agile project management approach that helps teams structure and manage their work through a set of values, principles, and practices toward a common goal.

[119] Examples of agile methodologies include Extreme Programming, Scrum, Lean Software Development, Crystal Methodologies, Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Methodology.

[120] Common principles include short development iterations, minimal design upfront, emergent design and architecture, collective code ownership and ability for anyone to change any part of the code, direct communication and minimal or no documentation (i.e., the code is the documentation), and gradual building of test cases. Refer to Noopur Davis, *Secure Software Development Life Cycle Processes: A Technology Scouting Report*, and related guidance, such as CISA, *Securing the Software Supply Chain: Recommended Practices Guide for Developers*, and *Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Security-by-Design and -Default*.

IV.K Quality Management

Quality management refers to coordinated activities to direct and control an organization regarding quality. It helps management assess whether newly developed, procured, or modified systems and components are operating as envisioned, and are designed and comply with an entity’s policies, standards, and procedures. Entities assess quality through manual, automated, or hybrid methods throughout the SDLC. Quality management should be an independent function from development, and prudent practices typically include implementing compensating controls when segregation of duties cannot be fully achieved. For example, developers should not test and validate systems and components that they developed.

Quality management comprises the functions of QA and quality control (QC), which together focus on providing confidence that quality requirements will be fulfilled. Although the goal is similar, QA and QC differ somewhat. QA refers to a planned, systematic pattern of all actions necessary to provide adequate confidence that a system, component, or facility conforms to established requirements. QC refers to the operational techniques and procedures (such as design analysis and inspection for defects) used to achieve quality requirements. QA and QC may be implemented separately or together, depending on the size and complexity of the entity and its development, acquisition, and maintenance activities.

Quality management is a critical part of well-managed development, acquisition, and maintenance activities. Comprehensive quality management, risk management, and testing standards provide a means to manage project risks and promote expected functionality, security, and interoperability in systems and components in a secure, resilient manner. Quality management policies, standards, and procedures should be applied to internally and externally developed programs and should address the following:

- **Commitment:** Successful projects generally include commitment from all involved parties. Senior management should adequately support and promote projects throughout an entity to promote success of a project. Failure by management to implement or support quality management programs hinders its ability to quickly detect project weaknesses and programming errors. Correcting weaknesses and errors detected late in a project life cycle is often more difficult and costly and may be less effective.
- **Expectations:** To improve quality management and minimize deficiencies, stakeholders should clearly define their expectations and effectively communicate their requirements initially and throughout a project. For example, expectations and requirements could include reliability, resilience, performance, simplicity, usability, interpretation of error messages, accessibility, interoperability, and robustness.
- **Completeness:** Each phase of a project life cycle should include procedures to follow and expected deliverables. Quality management personnel should verify the business case for a project, required functional features, and appropriateness of project considerations to determine completeness in each project phase before moving to the next phase. Audit and compliance personnel should verify that quality management includes quality standards to validate that the project meets internal and external requirements.
- **Scalability:** Projects vary in size and complexity. Quality management standards should match project characteristics and risks and may differ by project size and type. Quality

reviews may consider whether the entity's systems and components can scale, as necessary.

- **Measurability:** To accurately evaluate a project's success, management should assess results against defined expectations and requirements to determine project success. Quality management personnel should assess the quality of products and processes against measurable standards, metrics, and expectations.
- **Tracking:** Quality management personnel should validate that project personnel properly document a project's progress; record, report, and monitor problems to resolution; and communicate progress and concerns to management.
- **Independence:** Quality management personnel should be independent of the project they are reviewing to objectively assess elements, such as progress, problem resolution, documentation, and adequacy of the project.
- **Security:** Information security personnel perform a critical function in ensuring the confidentiality, integrity, availability, and resilience of internally developed and procured systems.[121] Quality management personnel should validate the existence and effectiveness of information security considerations. The extent of security validation should be commensurate with risks associated with the project.
- **Legal and regulatory:** Legal and compliance personnel provide guidance throughout a project to ensure that it conforms with applicable statutory and regulatory requirements to mitigate project risks. Quality management personnel should assess statutory and regulatory requirements for deliverables (e.g., digital accessibility)[122] and determine whether the systems and components can meet those requirements.

IV.L Documentation Standards

Thorough documentation allows stakeholders to clearly understand system and component functionality, security, control features, and resilience. Documentation conveys how a system or component was developed, how it functions, and where appropriate security and resilience points are included. Entity personnel can refer to documentation to effectively operate and maintain the systems and components. Additionally, well-prepared and maintained documentation reduces resilience risks, such as the loss of institutional knowledge common with personnel turnover. Documentation helps transfer key information (e.g., narratives, flowcharts, and any special system coding or file layouts) over time and across groups. Management should maintain documentation for both internally developed systems and components and externally procured products and services. Due to the sensitive nature of system and component documentation (e.g., internet protocol addresses, server names, and security controls), management should restrict access to sensitive documentation based on job description and need.

Examples of items that documentation may contain include feature and function descriptions, design specifications, programming descriptions, build procedures, and operating instructions. Also, documentation should be updated as systems and components are updated to ensure that the benefits described above continue.

[121] For more information, refer to NIST SP 800-53, rev. 5, *Security and Privacy Controls for Information Systems and Organizations*.

[122] For more information, refer to Section 508 of the Rehabilitation Act of 1973.

For acquired systems and components, management should obtain documentation from the third party and incorporate it into the entity's own stored documentation. Before purchase, effective management ensures (through an internal review or a third-party certification) that a procured system's documentation meets the entity's documentation needs and standards.

System documentation that includes current build and configuration instructions allows administrators to recover the system after a disruptive event. For a recovery process to be effective, these build and configuration instructions should be kept current as the system is maintained and changed. There are many build and configuration processes that are automated, and the scripts that automate those processes should be documented appropriately for future reference. However, in those cases it may be the manual action (unautomated) documentation that is at the highest risk of becoming stale.

Common documentation examples include the following:

- **System or component descriptions and topologies** that provide functional purposes, narrative explanations, and pictorial depictions of operating environments and the interrelated input, processing, and output functions of integrated systems and components.
- **System and program flowcharts and models** that identify the source and type of input information, processing, and control actions (automated and manual), and the nature and location of output information for a system or component. Additionally, program flowcharts present graphical views of the procedural program sequencing and provide a practical way to illustrate complex programs and routines. Administrators can use these flowcharts to view a system holistically or drill down to specific functions or interrelated components of a system or program.
- **System file and information layouts** provide specification useful in information collection, storage, and display.
- **Program documentation** details specific data input, processing, and output instructions, and should include documentation about system security. Examples of program documentation include program listings, source code, narrative comments, technical programming scripts,[123] and nontechnical descriptions of the scripts.
- **Manuals and procedures** inform IT personnel and end users on the operation of systems and components.

IV.M Post-Implementation Review

Organizations formally perform a post-implementation review (PIR) for system development and implementation projects to continuously improve. Processes and techniques that were effective are passed to future projects as effective practices. Problems and their solutions are also passed along to help future projects avoid pitfalls. In agile projects when there are multiple implementations, smaller reviews are completed after each task or deliverable is implemented. Typical PIR steps are

[123] *Merriam-Webster* defines a "script" in computing as "a sequence of instructions or commands for a computer to execute."

- Gathering data including project initiation documents, post-implementation metrics, and post-implementation stakeholder interview results.
- Analyzing any gaps between project initiation goals and objectives and actual outcomes, including the reasons for any gaps.
- Summarizing the analysis and any recommendations in a report to management.[124]

Common project aspects analyzed in a PIR include the following:

- User satisfaction.
- Actual benefits compared to projected benefits.
- System or component defect level when implemented, which helps measure system testing effectiveness.
- Project management effectiveness.
- Actual costs and timeline versus business case projections.

When entities analyze project aspects like those above and pass the results to management and future project teams, they raise future project success probability and mitigate the risk of future project failure.

IV.N IT Project Management

Action Summary

Consistent use of appropriate project management policies, standards, and procedures can help personnel be effective in identifying and mitigating project risks before they materialize.

Examiners should review the following:

- Project plans, proposals, and status reports to determine the effectiveness of IT project manager collaboration with stakeholders (e.g., through the PMO) for oversight and completion of IT projects.
- Board and committee minutes to evaluate how the entity prioritizes projects relative to business goals and objectives, evaluates the project's effect on operations, and monitors and supports projects during execution.
- Project plans and proposals to determine how well they identify the project purpose, the requirements to address business needs, and the deliverables for each project phase.
- Change management documentation to determine the appropriateness of change decision-making levels and the consistency of approved changes with the project charter and plan.
- Testing and quality management plans to determine the adequacy of quality controls.
- Project closeout documentation to evaluate the entity's ability to inform future projects.

[124] Refer to U.S. Government Accountability Office's *The Post-Implementation Review*.

Project management is the use of specific knowledge, skills, tools, and techniques to deliver something of value to people.[125] Effective project management includes established policies, standards, and procedures that project personnel apply enterprise-wide. Examples of projects include developing a new product or service, expanding into a new geographic market, and investing in a new system. When reviewing and prioritizing projects, effective management considers the project's effect on operations and the entity's needs (e.g., IT, information security, lines of business, customer needs, and regulatory and legal compliance). Entity projects often include a supporting IT project due to the underlying systems and components in business operations. For instance, when management considers offering a new product or service or expanding into a new geographic market, there are often IT implications. Effective IT project management processes, which are a subset of and align with the enterprise project management policies, standards, and procedures, should address issues specific to IT projects. Effective management develops and follows consistent processes to identify risks and oversee IT projects to address any risks identified. IT projects generally include IT system development, acquisition of a new IT system or component, or significant maintenance of or change to an IT product or service. If the entity has a PMO, IT project managers should work with the PMO to oversee and carry out IT projects.

An IT project should be managed in relation to the entity's size and complexity and consistent with the project's criticality and risks. An ineffectively managed IT project may result in late deliveries, cost overruns, or systems or components that do not meet entity, user, customer, or regulatory requirements. Systems or components that do not meet entity minimum standards or customer requirements can result in underused, insecure, or unreliable products or services. Adding functions, security, or automated controls into systems or components late in the initial development or after implementation to address missed requirements may incur substantial costs (e.g., personnel, time, and money), resulting in less effective systems or components. Furthermore, poor project management can lead to legal issues, such as violations of law and regulation and monetary penalties.

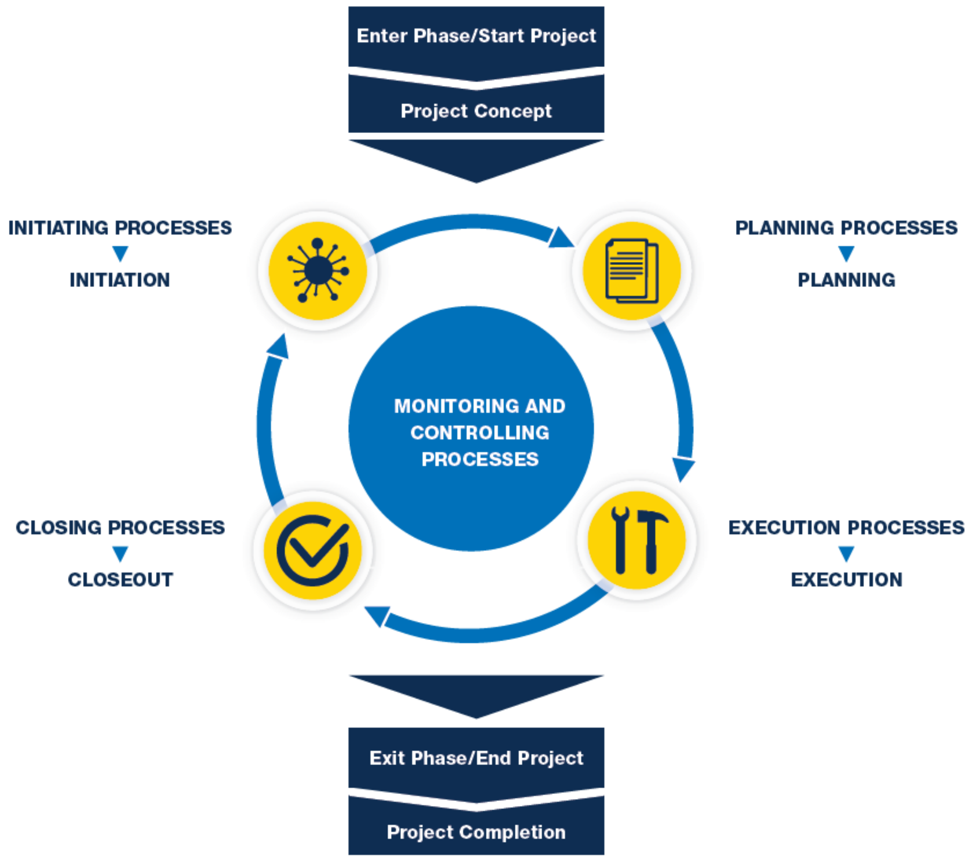
The project management discipline discussed in this section and the SDLC discipline discussed in the next section overlap significantly. In practice, effective system and component development teams integrate discipline use without duplicating effort.

IV.N.1 IT Project Phases

A general project management life cycle is shown in figure 3. The type of IT project (e.g., development, acquisition, or maintenance) will affect the project management life cycle chosen. For example, when an SDLC is chosen for a development project (e.g., adaptive versus predictive), the procurement processes chosen for an acquisition project, and the change control practices chosen for a maintenance project will affect how the project is managed. Furthermore, the activities completed within a project's life cycle are based on the characteristics of a project and the employed project management methodology.

[125] Refer to PMI's "What Is Project Management?"

Figure 3: Project Management Life Cycle



A project team often begins a project by creating a proposal to management that explains the business case including the desired system and component features and project scope, the planned benefits, the estimated time required, and the estimated cost.[126] Management reviews the proposal and determines whether the project is feasible and makes sense for the entity. If management approves the proposal and initiates the project, the business case serves as the basis for developing a detailed project plan.

The four main phases shown in figure 3 (Initiation, Planning, Execution, and Closeout) are described further in the following subsections.

IV.N.1(a) *Initiation*

The project's initiation phase begins with identifying stakeholders who determine the project's purpose, scope, and final deliverables. The following would typically be completed during this phase:

[126] PMI, *A Guide to the Project Management Body of Knowledge*, fifth edition, p. 50.

- A project request that provides the business case including a description of the work, the benefits, alternatives considered, the impact of not doing the work, initial estimates of resources and schedule, and strategic match.
- A project charter that identifies the project owner (i.e., the responsible party), and defines the mission and main goals of the project.
- Initial business and technical requirements.[127]

When the project involves developing a new system or component or making significant changes to an existing system or component, management analyzes and determines whether to develop or acquire (referred to as “build or buy”) the solution. A feasibility study[128] helps in the process of making this determination. Considerations in the feasibility study include availability of internal expertise, cost of building the solution compared with procuring it, availability of an external product that meets the entity’s requirements, solution implementation timeline, and the project’s complexity. Refer to table 2 in the “Feasibility Study” section of this document for additional feasibility considerations. Regardless of whether the decision is to build or buy, the feasibility analysis helps management to verify the reasonableness of the preliminary assumptions and to identify resource requirements in greater detail.

IV.N.1(b) *Planning*

In the planning phase, management forms a project team[129] that defines and refines project deliverables to achieve the objective and develops the project plan. Project plan common elements are tasks and milestones, task timelines, and names of those who are responsible for completing each task. Project teams engage stakeholders to define deliverables that achieve objectives and meet the entity’s functional, IT, information security, and legal and regulatory requirements. Requirements include systems, components, and resources that will support and interact with any new product or service. Clearly defined deliverables (including documentation deliverables) help stakeholders understand expectations. The project team should also define testing requirements and objective acceptance criteria, which are unbiased and predefined criteria for determining whether the project meets the stated objectives through milestones and project completion. Additionally, security and resilience design should be included from the beginning of the project to be most effective. Information regarding project management governance is in the *FFIEC IT Handbook’s* “Management” booklet.

[127] The NIST Glossary defines “requirement” as “a statement that translates or expresses a need and its associated constraints and conditions.”

[128] A feasibility study considers the critical aspects of a proposed project; assesses the degree to which the requirements, designs, or plans can be implemented; and helps determine the likelihood of the project’s success. A formal feasibility study may not be performed at smaller, less complex entities; however, management should perform some analysis of system or component needs for a given project.

[129] Depending on the IT project type and complexity, the project team may include multidisciplinary stakeholders from varying lines of business. An effective project team’s roles and responsibilities are clearly defined and communicated so everyone involved is aware of their responsibilities.

IV.N.1(c) *Execution*

During the project's execution phase, the project team completes the project plan tasks. The team tracks and compares actual task execution with the project plan, maintains a log of problems (e.g., inability to meet milestones, resource changes, and unanticipated risks), tracks problem resolution, reports on effects to the project timeline, and monitors interdependency issues. During the execution phase, scope creep is a common problem that can occur as developers address issues or receive subsequent requests to add or modify a system's features. It may result from inadequately defined requirements, a lack of a project change control process, inaccurate time and budget analysis, or a weak project manager or executive sponsor. Establishing change approval procedures during development and cutoff dates (after which time requested changes are deferred to subsequent versions) helps mitigate scope creep.

Strong change approval procedures enforced by management can mitigate the risks that come with scope creep. Procedures can include independent project monitoring for additions or modifications of functional and nonfunctional features to help enforce management review and approval. Procedures can also include project plan variance management reporting that could indicate scope creep (e.g., delays in completion of tasks or cost overruns). Significant project plan variances may require the entity to reassess and re-baseline project plans. Project managers regularly report the project's status to stakeholders, senior management, and, when appropriate, the board in order to inform of changes and issues.

Strong testing and other QC procedures can mitigate the risk that the entity's project goals and stakeholder requirements are not met during the execution phase. Testing helps find deficiencies or defects and helps ensure that the system or components operate as intended. Depending on the project type, testing may be scheduled throughout the project during any phase. For example, testing may not always occur during every sprint when using an agile development methodology. Users, designers,[130] developers, and IT staff may be involved in testing. Throughout the testing process, management should maintain comprehensive and accurate documentation reflecting the testing methodology employed, tests performed, and test results.

IV.N.1(d) *Closeout*

During the project's closeout phase, the project team delivers the agreed-upon scope items as outlined in the project plan. For example, in a systems development or acquisition project, project closeout is when the project team transitions the systems to the operations staff (i.e., moves application from a staging to a production environment). Often during the project closeout, project teams analyze the project (i.e., post-implementation review) to identify effective project practices and discuss issues encountered and how they were resolved. This information is used to inform and improve project management practices. Project teams review project documentation in the closeout phase to ensure that it is complete, supports maintenance, and can be used by future teams to identify effective practices.

[130] Designers may include personnel engaged in creation and execution of plans for a project or structure (e.g., graphic designers, web developers and digital designers, web and digital interface designers, computer programmers). See Bureau of Labor Statistics, *Occupational Outlook Handbook*, for more information on job descriptions.

IV.N.2 Monitoring and Controlling

Monitoring and controlling are important parts of the processes of IT project management. This takes place throughout all phases of IT project development (refer to figure 3). Monitoring and controlling are helpful for tracking, reviewing, and evaluating the progress of an IT project. Monitoring and controlling helps management identify IT project management risks (e.g., errors, cost overrun, scope creep, and missing milestones). A combination of management and the project manager or sponsor are responsible for monitoring and controlling the entire project throughout all phases.

During the initiation phase, monitoring and controlling activities include the following:

- Validating that relevant stakeholders are identified and all stakeholders' project requirements are well-defined.
- Coordinating the project phases, including tasks and activities, sufficiency of resources, and process steps and timing for each phase.
- Validating that the project scope is sufficiently identified and refined to minimize scope creep.
- Evaluating the application of project management policy, standards, and procedures.
- Validating that appropriate success criteria are identified and the project is feasible.
- Determining effective establishment of reporting lines, reporting processes, and reports.

During the planning phase, monitoring and controlling activities include the following:

- Validating the appropriateness of the project plan's tasks and time frames as well as resource sufficiency.
- Reviewing the project plan iteratively and validating any approvals for any updates.
- Validating stakeholder support and engagement.
- Verifying that the project plan and associated project documents are appropriate, comprehensive, and approved.

During the execution phase, monitoring and controlling activities include the following:

- Meeting regularly with the project team to track progress, address impediments or issues, and track task and milestone completion.
- Monitoring, measuring, and analyzing project performance compared to the project plan and identifying any variance.
- Validating that any changes are authorized, quality is maintained, scope creep is mitigated, and stakeholders are informed.
- Recommending corrective or preventive action.
- Verifying that configuration management policies and procedures are followed.
- Monitoring expenditures compared to the project budget and addressing any cost overruns.

During the closeout phase, monitoring and controlling activities include the following:

- Validating that the project team follows closeout processes when the project is complete.
- If a project is discontinued before implementation, determining and documenting the reasons for the abnormal project closure for use by future project teams.
- Verifying that project deliverables were accepted.
- Overseeing a post-implementation review.
- Confirming that the system or components have been added to inventories that support routine maintenance, such as patching.

IV.N.3 IT Project Documentation

IT project documentation helps ensure clear communication among the project team, management, and other stakeholders. For example, strong written documentation provides a record of key decisions (e.g., project approvals, timeline, and scope changes) that the project team can refer to throughout the project.

Types of IT project documentation range from simple documents, such as meeting minutes and project status reports, to more complex documents such as feasibility studies, project plans, requirements, and PIRs. Project documentation allows management to track and monitor security, resilience, and compliance concerns throughout the project and after its completion. Entities establish documentation standards and procedures to mitigate the risk of inconsistent or incomplete documentation. When determining the amount of project documentation needed, various factors to consider include an entity’s size and complexity, the project type (e.g., development, acquisition, or maintenance), and the project’s complexity. The following subsections describe common examples of project documentation.

IV.N.3(a) *IT Project Request*

IT project requests outline proposed projects including the reason for initiating the project. Effective project requests convey proposed business objectives and project requirements between the project team, management, and other stakeholders. Project requests also identify project responsibility and accountability, including identifying the project sponsor. Project requests often include the following:

- Project sponsor.
- Customer or user.
- Other stakeholders.
- Purpose and proposed scope of the project.
- Project deliverables.
- Project constraints (e.g., time, budget, and technology).
- Business case.[131]

[131] A business case generally accompanies the project request and provides additional information to help set appropriate parameters for the project.

IV.N.3(b) *Business Case*

A business case conveys business related information to help management determine whether to fund a project. The goal is to justify the project resources (e.g., budget and staffing) to address a business need. Typically, the project's sponsor helps to develop and owns the resulting business case.[132] Business cases often include the following:

- The need or problem the project will address (e.g., systems not meeting operational requirements or loss of market share).
- Alternative solutions including estimated staffing, timeline, costs, benefits, and risks.
- The proposed solution with rationale (e.g., buy a new system or develop a new component).

Effective business cases address both the parameters for implementing and maintaining the solution. For example, both the costs to build and maintain various system options are analyzed before a proposed solution is presented.

IV.N.3(c) *Feasibility Study*

A feasibility study is an analysis of a known problem or need and the proposed solution conducted independent of the project request team. It helps stakeholders evaluate a proposed solution considering relevant factors—including economic, technical, legal, and scheduling considerations—to determine a project's probability of success. Table 2 outlines feasibility study considerations.

Table 2: Feasibility Study Considerations

Business considerations	• Entity objectives. • Effect on existing operations. • IT security risk assessment to determine security risks and mitigation needs. • Entity's IT and information security requirements addressing data confidentiality, integrity, availability, and resilience. • Expected benefits. • Potential entity changes regarding facilities or the addition or reduction of users, IT staff, key security roles, or managers. • Budget, scheduling, training, or personnel constraints (including for oversight of third-party relationships). • Licensing needs. • Potential legal or regulatory issues that could affect the project's feasibility. • Estimated completion dates of IT projects and major project milestones. • Expected benefits and risks for consumers relative to the new product or service. • Third party's due diligence throughout the supply chain.
Functional requirements	• End user needs. • Internal control and information security requirements. • Operating, database, and backup system requirements (e.g., type, capacity, performance, scalability, and resiliency). • Hardware requirements (e.g., infrastructure needs, processing chips, and credit card or point-of-sale components). • Facility needs (e.g., raised floor, cabling, and heating, ventilation, and air conditioning [HVAC]).

[132] Refer to PMI, "Is This Really Worth the Effort? The Need for a Business Case."

	• Network requirements (e.g., physical, virtual, software-defined, and hyper-converged networks; types of storage; number of users; and type, volume, and frequency of data transmission). • Connectivity and interface requirements with internal or external applications, third-party service providers, internal or external users, and customers. • Third-party hardware and software vendor requirements related to entity functional requirements. • Product development, design, and testing standards throughout the supply chain. • Supply chain considerations related to entity business considerations and functional requirements. • Entity's requirements compared to third-party products, services, and activities implemented in the supply chain.
Cost-benefit analysis	• Expected useful life of the proposed product or application. • Alternative solutions (e.g., build or buy). • Estimated costs of projects (e.g., overall and by project phase). ○ Nonrecurring project costs (e.g., hardware, software, and overhead). ○ Recurring operational costs (e.g., personnel, maintenance, telecommunications, and overhead). • Soft (or intangible) costs.[133] • Tangible benefits (e.g., increased revenues, decreased costs, and return on investment). • Intangible benefits (e.g., improved public opinion or more useful information).

IV.N.3(d) *IT Project Plans*

IT project plans describe how a project will be executed, monitored and controlled, and closed. Entities use project plans to communicate tasks, time frames, and responsible parties to project stakeholders throughout a project. The details in an IT project plan identify necessary technical steps in sequence and help the team identify any missing tasks that are needed to achieve the project's objectives. The project plan also identifies interconnectivities and interdependencies throughout the project—how segments of the project are interrelated. Generally, IT project plans consider the following:

- Approved project scope.
- Roles and responsibilities.
- Communication among the project team, stakeholders, and management.
- Schedule, including completion of key tasks and achieving milestones.
- Quality management tasks (e.g., testing both developed systems and patches for procured systems).
- Risk management tasks (e.g., periodic risk reviews).
- Task budgets.
- Implementation tasks (e.g., training and deployment).

IV.N.3(e) *Closeout Documentation*

Entities use closeout documentation to apply lessons learned from past projects to future projects. These lessons can include practices and techniques that were effective, problems that could be avoided if particular processes are used, and how high-performing project teams were

[133] Soft costs generally include indirect dollars spent in other areas (e.g., sales, general, and administrative expenses) to support a change in business model, equipment, or practices.

formed. Many lessons can be memorialized by updating existing policies, procedures, and standards. Several documents can be used to improve future projects, such as the following:

- Project plans with projected and final task completion dates.
- PIR.
- Risk reviews including suggested changes to project management-related policies, standards, and procedures.

IV.O System Development Life Cycle

Action Summary

Management should implement an SDLC to manage systems and system components throughout their life cycle and achieve the objectives of confidentiality, integrity, availability, and resilience to achieve the entity's business objectives.

Examiners should review the following:

- The documented management and control processes, including for supply chain partners.
- Responsibility and accountability assignment.
- Key stakeholder involvement level.
- Stakeholder communication and tracking of all SDLC phases and actions.

One NIST SDLC definition is that it is "the scope of activities associated with a system, encompassing the system's initiation, development and acquisition, implementation, ongoing operation and maintenance, and ultimately its disposal that instigates another system initiation."[134] The acronym SDLC is also used to refer only to software development and implementation (standing for the software development life cycle). However, this booklet uses the acronym SDLC to refer to the broader concept encompassing the entire system and its components. If an entity engages in internal development activities, management should have a documented process (e.g., SDLC) that management and personnel use to manage and control development activities. Effective SDLC processes are well-documented by organizations such as NIST.[135] Using proven processes helps to mitigate the various risks of building, maintaining, and retiring systems. Implementing the SDLC allows management to organize these activities into smaller, more manageable segments or phases. Data security should be considered throughout all phases of the SDLC to maintain confidentiality, integrity, availability, and resilience.

The SDLC discipline discussed in this section and the project management discipline discussed in the last section overlap significantly. In practice, effective system and component development teams integrate discipline use without duplicating effort.

[134] Refer to NIST Glossary.

[135] Refer to NIST SP 800-160, vol. 1, rev. 1, *Engineering Trustworthy Secure Systems*.

IV.O.1 SDLC Phases

Action Summary

During the SDLC phases, system development project teams work to achieve project objectives, mitigate the risk of lower quality development projects, and address security. These considerations are considered throughout all phases.

Examiners should review elements of each phase such as the following:

Initiation Phase

- The system's purpose, expected benefits, how it supports business objectives, and any legal and regulatory requirements.
- Initial security impact analysis and validation of the appropriate project specifications.
- Extent to which the project request is communicated to stakeholders in the supply chain.

Development or Acquisition Phase

- Design specifications, including security control design.
- Risk assessments, including using the results of security risk assessment, to supplement the baseline security controls.
- Risk mitigation strategies.
- Test, conversion, implementation, and training plans, accounting for confirmation of the functionality and controls.
- Change management documentation.
- Draft user, operator,[136] and maintenance manuals.

Implementation and Assessment Phase

- Design reviews and system tests, including any new specification testing.
- Deployment approach (e.g., phased-in, simultaneous).
- Training, including user and system support documentation.
- PIR.

Operations and Maintenance Phase

- Performance and controls monitoring.
- Change management controls, including as used to control configurations.
- ITAM inventory and associated audits.

Sunset and Disposal Phase

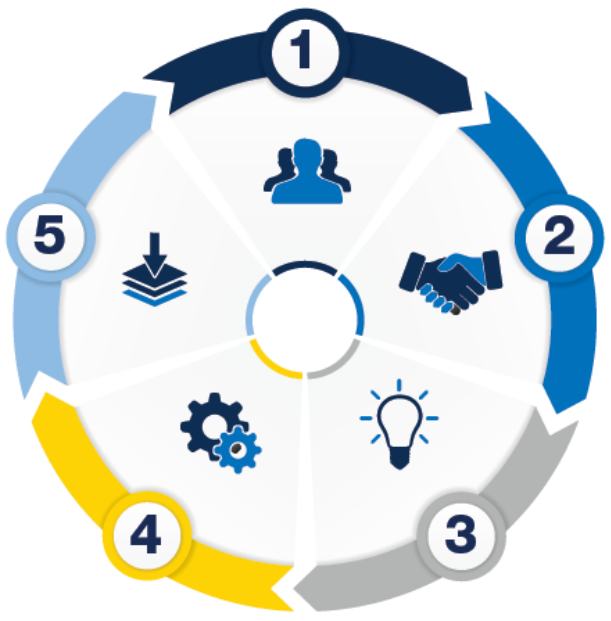
- Plans and validation measures for accessing and retrieving data from archives.
- Off-boarding procedures, including for third-party providers.
- The entity's post-disposal review, including any lessons learned documentation.

[136] Refer to NIST Glossary.

Effective management understands the SDLC phases and the actions needed in each phase. The phases may be divided differently depending on the entity, the project type and characteristics, and the SDLC used. Management should identify the actions and assign responsibility and accountability for completing those actions. Key stakeholders of the system or component being developed or modified should monitor progress in each phase including the output of each phase. This involvement mitigates the risk that the system or component does not deliver the requested functionality. Management should maintain confidentiality, integrity, availability, and resilience throughout all phases of the SDLC.

An entity that defines its SDLC phases helps ensure transparency and accountability to, and agreement and system or component acceptance by, the stakeholders. Figure 4 illustrates the five general SDLC phases (Initiation, Development or Acquisition, Implementation and Assessment, Operations and Maintenance, and Sunset and Disposal) identified by NIST.[137]

Figure 4: NIST Example of a System Development Life Cycle



1. INITIATION
The need for a system is expressed, and its purpose and high-level requirements are documented.

2. DEVELOPMENT OR ACQUISITION
The system is designed, purchased, programmed, developed, or otherwise constructed. This phase often consists of other defined phases, such as the system development phase of the acquisition phase.

3. IMPLEMENTATION AND ASSESSMENT
After initial testing, the system is installed or fielded.

4. OPERATIONAL AND MAINTENANCE
The system performs the work it was developed for.

5. SUNSET AND DISPOSAL
The system is disposed of once the transition to a new computer system is completed.

IV.O.1(a) *Initiation*

The initiation phase begins when entity personnel identify an opportunity to build, buy, or modify a system or component and formally seek approval with an IT project request. The project team should describe the IT project's purpose, identify expected benefits, and explain how the proposed system or component supports the entity's objectives. The project team should summarize the confidentiality, integrity, availability, resilience, and legal and regulatory

[137] Refer to NIST Information Technology Laboratory (ITL) Bulletin, "The System Development Life Cycle (SDLC)."

requirements. They should identify alternative solutions and justify their recommended solution. Through IT project request development, stakeholders should gain a common understanding of the project's objectives, security considerations, and risk management planning to mitigate the risk of project failure.

During the IT project request review, management can accept, reject, or request changes before it allocates resources. Management can also commission a formal feasibility study to inform its decision. If the IT project request is to modify an existing system, management should consider performing a security impact analysis[138] to identify any negative impact to existing security controls. Such a security impact analysis may be performed as part of the feasibility study.

Planning, particularly in the project's early stages, should help the project team and management coordinate development activities and manage risks effectively. Planning helps the project team and management add or clarify the project's specific activities, resources, costs, and benefits. A critical part of planning is to coordinate stakeholder discussions to identify and document as many of the entity's functional, security, and network requirements as possible.

Primary items entity personnel typically include in planning consist of the following:

- Responsibilities of third-party service providers, internal audit, information security, and IT staff.
- Entity SDLC processes.
- SDLC phase acceptance criteria including review and approval procedures to help ensure that development teams complete all SDLC phase or independent sprint requirements before moving into subsequent phases.
- Control and security features to be designed and built, or acquired, and implemented.
- Change management processes to minimize disruption to the project plan.
- Risk management processes, including project methodology selection.
- Cost tracking mechanisms, including to track overhead (e.g., office space, hardware, and software used during the project) as well as other related costs (e.g., budgeting personnel expenses, outsourced activities).

When evaluating IT project requests, effective management considers input from all stakeholders. For example, management evaluates the appropriateness of the requested functional requirements. Each function has design and development implications. Each function requires testing, documentation, and ongoing support. Therefore, the exclusion of unnecessary functions can significantly reduce the resources required to support a request. Management considers and analyzes all requests to determine whether to develop or acquire the system or component.

[138] For more information on conducting a security impact analysis, refer to NIST SP 800-128, *Guide for Security-Focused Configuration Management of Information Systems*.

IV.O.1(b) *Development or Acquisition*

During the development or acquisition phase, the system or component is designed and developed or acquired. Key system or component designs are created or reviewed before development begins or the acquisition is completed. When a project is initially proposed and reviewed, the decision to develop or acquire may be uncertain. Early stakeholder involvement in design creation or acquisition specification review mitigates the risk that the system design does not support the functional requirements. The more predictive the SDLC is, the more design work will occur earlier in the SDLC. The more adaptive the SDLC is, the more design work will be spread throughout the SDLC phases.

When an entity acquires a system or component, important design considerations include how much configuration is possible, whether the system or component can be restored to a secure configuration, and whether there are restrictions on users or services that can make configuration changes. Entity personnel can configure systems and components to avoid specific known threats and risks based on the entity's risk appetite. For example, if a product or service is used by bank customers, it may be important that the product or service's security and functionality allows the capability for the customer to change some configurations (e.g., change location restrictions, adjust monetary limits, and allow use of physical access controls).

System and component security should be a high priority throughout this phase. If an IT security risk assessment was conducted as part of the feasibility study, it can be used early in this phase to design baseline controls to address risks identified. If an IT security risk assessment was not completed as part of the feasibility study, it can be completed early in this phase. Designing security controls early in the development and acquisition phase is important to system or component confidentiality, integrity, availability, and resilience. If the system or component is being developed or acquired, it is important to address security concerns, such as those for customer information, and personnel should "identify reasonably foreseeable internal and external threats that could result in unauthorized disclosure, misuse, alteration, or destruction of customer information or customer information systems."[139] Therefore, it is important to identify security controls requirements and consider appropriate controls that might affect the entire supply chain. It is typical for system and component functionality and design to change throughout the phase, which can change security effectiveness. The security impact analysis performed during the SDLC initiation phase can be helpful to evaluate security adequacy before, during, and after system and component changes are made in this phase.[140] Early implementation

[139] Refer to GLBA and 15 USC 6801 and 6805(b), further implemented by FFIEC members as follows: FDIC: 12 CFR 364, appendix B, "Interagency Guidelines Establishing Information Security Standards," and 12 CFR 364, supplement A to appendix B, "Interagency Guidance on Response Programs for Unauthorized Access to Customer Information and Customer Notice;" FRB: Regulation H, 12 CFR 208, appendix D-2, "Interagency Guidelines Establishing Information Security Standards," and Regulation Y, 12 CFR 225, appendix F, "Interagency Guidelines Establishing Information Security Standards"; NCUA: 12 CFR 748, appendix A, "Guidelines for Safeguarding Member Information"; OCC: 12 CFR 30, appendix B, "Interagency Guidelines Establishing Information Security Standards." Additionally, refer to FTC 16 CFR 314, "Standards for Safeguarding Customer Information," for similar provisions for service providers.

[140] For more information on conducting a security impact analysis, refer to NIST SP 800-128, *Guide for Security-Focused Configuration Management of Information Systems*.

of security controls can minimize their cost, allow for efficient deployment, and reduce integration issues.

Appropriate personnel should define and create initial testing, conversion, implementation, and training plans and obtain management approval during the development or acquisition phase. Development personnel should initially test the design during development to confirm that the system's or component's functionality is meeting requirements and controls, including security controls, are functioning as intended. Testing early and throughout this phase helps a team identify functional or security flaws when remediation can be less expensive.

The potential for scope creep increases during this phase, and project stakeholders monitor and mitigate this risk. Effective entities have a process to review the appropriateness of new or modified requirements and development changes to minimize scope creep, as discussed in the "IT Project Management" section of this booklet.

Designers should document completed designs and functions, including information security designs. Design documents may include system architecture, software architecture, and data structure design. Detailed documentation enhances a developer's ability to modify systems and components after they are placed into production. The documentation should include appropriate information such as all related systems and components (including IoT product components and devices), any accreditation, certification, or evaluation results. Documentation can be used by management to compare original objectives and specifications with systems and components ready for implementation. The development team typically drafts user, operator, and maintenance manuals during this phase.

IV.O.1(c) *Implementation and Assessment*

The objective of the implementation and assessment phase is to prepare the system, operational environment, entity, and end users for system or component use, and to assess whether the system or component meets business needs and operational requirements. In this phase, the team configures and enables system or component features, tests the features' functionality, and installs or implements the system or component after obtaining appropriate approvals.

It is important to thoroughly test before implementation to help ensure that the system or component meets all entity specifications. These tests should be documented so that as the system or component is changed, previous tests are used and adjusted to validate rather than creating new tests. Building on original testing documentation also may result in subsequent tests being more comprehensive than if they were not leveraged.

The more complex and critical the development or acquisition, the more important it is to rigorously assess the product before implementation. For example, conversions of core banking systems require more detailed reviews and higher levels of approvals from management given their significance to delivery of business services, as opposed to a lower level of oversight for routine maintenance of a less critical system. These reviews would occur before implementation. In the more complex and critical implementations, approval from many stakeholders such as business continuity management, information security, and third-party risk management may be necessary. For more information on routine changes and conversions, refer to the "Implementing Changes" section of this booklet.

Primary implementation phase tasks include communicating the implementation schedule to internal and external stakeholders, training users, and installing the system or component. To be most effective, the implementation schedule takes into account the business impact (e.g., conversions are scheduled on long weekends to allow additional time before affecting weekday business). Management and other appropriate personnel perform tasks such as the following during the implementation phase:

- Execute the implementation plan, including the following:
  - Assessing risk for issues that may occur during implementation activities (e.g., rollout, conversion, or update).
  - Conducting user acceptance testing (UAT).
  - Updating the asset inventory.
  - Training (internal and external).

- Input, import, or convert data and validate the functionality of the system or component using those data.
- Confirm the confidentiality, integrity, availability, and resilience of systems and data.
- Confirm the interoperability of systems and components.
- Configure and test system and component security controls.
- Validate performance parameters (e.g., throughput, capacity, error rates, and memory usage) and ensure that the system or component operates as expected.
- Conduct a final audit of information security and adherence to policies and procedures, as well as risk tolerance levels. Determine whether the findings of the audit meet the entity's predetermined requirements for system deployment.
- Create and test back-out or rollback plans to prepare for unforeseen and unrecoverable problems that may occur during implementation. Entity personnel may accomplish this by various back-out or rollback methods (e.g., using one or more restart points or reverting to the legacy system or prior version).
- Conduct PIRs (e.g., survey users, review complaints, plan future functionality, and identify other improvements). Additionally, the reviews may identify lessons learned and process improvement information.

Implementation strategies vary and are designed to mitigate different types of implementation risk. Examples of implementation types may include the following:

- **Parallel implementation** consists of running the old and new system or component simultaneously until the new system or component is evaluated and approved. If the new system fails, operations can continue using the old system while issues are corrected. Running two systems or components simultaneously requires additional resources (e.g., costs, personnel, and time), but is a logical strategy if no system downtime is tolerable.
- **Pilot implementation** involves releasing the complete, new system or component to a limited (i.e., pilot) group of users. After successful rollout to that pilot group, management decides how to proceed with the rollout process. Advantages of using a piloting strategy

include minimizing the risks of a comprehensive rollout to the entire user population, as well as opportunities to test the system with the initial rollout and any additional rollouts. However, this strategy may also be time-consuming and expensive. A risk to this strategy is that it may allow for large numbers of insignificant enhancements instead of focusing on critical system issues.

- **Phased implementation** involves deploying a new system or component in phases (i.e., rolling out modules, features, or functionality incrementally over time) generally according to a predetermined schedule. This strategy helps mitigate the risk that a foundational flaw is discovered at the end of the project rather than earlier when it would be easier to correct. This strategy potentially requires less training and fewer resources spent at one time, and potentially fewer errors on which to focus during each phase. This strategy may lengthen the implementation time frame. In addition, there may be integration and interoperability concerns, such as reverse compatibility issues with legacy components while the old system is being replaced in phases.

Training is critical to the success of the implementation and assessment phase as are training plans and supporting materials for operating, using, and maintaining the system. Training is often the first exposure to the system for most users and should be provided before system or component deployment to the production environment. Training enables users to familiarize themselves with the new or updated system or component. A positive training experience typically improves user acceptance. During this phase, appropriate personnel should coordinate training logistics, including who should be trained, what the training involves, and when training should be conducted. Effective management organizes a training and awareness campaign and notifies users of any implementation and training responsibilities. This helps establish user expectations regarding system and component capabilities. Those responsible for supporting the system or component in the entity should have a combination of technical documentation, training, and hands-on assistance enabling support personnel to provide operational support to the users. Once the product is implemented, management should perform a PIR (see the "Post-Implementation Review" section of this booklet for more information).

IV.O.1(d) *Operations and Maintenance*

In the operations and maintenance phase, systems and components are in production and operating; enhancements and other modifications are developed and tested; and hardware and software components are changed. Entity personnel should continuously monitor system and component performance and adjust the system or component so that it operates consistent with pre-established user, security, and other entity requirements. Entity personnel should manage and document configuration changes whether baseline or unique. Documenting system and component changes and assessing the potential impact of these changes on a system's or component's security, functionality, performance, and resilience are essential activities in this phase.

Maintenance can occur periodically or regularly, such as annually, semiannually, quarterly, or weekly. In agile-based methodologies, maintenance may occur continuously. The maintenance frequency varies across systems and components, and the timing is prompted by business and operational need.

An entity’s change control processes play an important role in the operations and maintenance phase. Changes to a system or component can affect baseline configurations and controls, and this impact should be analyzed. To mitigate the risk of an enhancement or other modification disrupting or degrading operations, effective entities follow established change management policies, standards, and procedures. For more information, refer to the “Maintenance” section of this booklet and the *FFIEC IT Handbook’s* “Architecture, Infrastructure, and Operations” and “Information Security” booklets.

Examples of commonly performed tasks in this phase are the following:

- Ensuring that systems and components are operational and available during the defined hours of operation.
- Implementing nonemergency change requests during scheduled outages.
- Ensuring that all processes, manual and automated, are documented.
- Ensuring adequate resource availability for operations, maintenance, and resilience.
- Performing and testing system, component, and data backups.
- Validating the entity’s physical security measures.
- Ensuring that contingency plans for resilience are updated, tested, and funded.
- Ensuring that all operations and maintenance personnel take appropriate, ongoing training.
- Maintaining and monitoring system and component performance measurements, statistics, and system logs.
- Ensuring that third-party SLAs are being monitored and met.
- Performing configuration, security, and design assessments to ensure that system and component parameters and configurations are correct.
- Patching software for systems and components.
- Managing and controlling configurations and changes to the system. This includes installing, configuring, upgrading, and maintaining systems, components, and data, as well as related documentation.

Management is responsible for operations and maintenance phase task performance regardless of whether tasks are performed by entity or third-party personnel. For more information, refer to the *FFIEC IT Handbook’s* “Outsourcing Technology Services,” “Business Continuity Management,” “Architecture, Infrastructure, and Operations,” and “Information Security” booklets.

IV.O.1(e) *Sunset and Disposal*

In the sunset and disposal phase, plans are developed for the orderly termination of systems or components and the preservation of data that reside in them during the transition to a new system or component. Data to be preserved include entity data and client and consumer data. The systems, components, and data may be migrated to another system, archived, reassigned, or destroyed. There are several risks in the disposal phase such as the risk of sensitive data disclosure to unauthorized individuals, business functionality impairment, intellectual property loss, and reputation damage. Archive solutions may address the frequency and speed of any future data retrieval. Additionally, effective entities have a process in place to periodically validate the accessibility of archived data.

Specific tasks and activities occurring during this phase depend on the risk and complexity of the systems, components, or services. Considerations in system retirement and disposal include the following:

- Disposal or transition plan involving all applicable parties that identifies critical steps, decisions, and milestones to properly terminate, transition, and dispose of a system, component, service, and data.
- Hardware and software components to preserve, including for archive use.
- Data archive and migration—data could be migrated to the new system, archived, or a combination of the two.
- System and component documentation preservation.
- Advanced notification to all end users and stakeholders of the system or component sunset plan and the planned termination date.
- Plans for end-user migration to a replacement system or component.
- Plans for permanently erasing (i.e., sanitizing) data from the terminated system or component. Methods for sanitization include deleting, overwriting, and destroying data.[141]

If the system or component requiring disposal is managed by or the related data are stored by a third party, comprehensive off-boarding procedures help mitigate risks related to unauthorized access to sensitive information. Procedures should address items such as access, security, data governance and storage, and recordkeeping requirements (e.g., legal, regulatory, compliance, and audit). Effective management has a process to validate that its third party appropriately performs all necessary steps associated with system, component, data, and service sunset and disposal. The third party should maintain and make available to management evidence (e.g., certification) of disposal and archival activities to demonstrate completion.

Effective entities perform a post-sunset and disposal review that confirms the necessary processes were completed and documents lessons learned from shutting down and archiving the terminated system or component. The review should identify archived data and documentation locations. The review may be conducted again within six months of system or component disposal to identify any missed issues that only surface after an extended period of the system or component not being available.

For more information on sunset and disposal, refer to the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations," "Information Security," and "Outsourcing Technology Services" booklets.

[141] For more information on data sanitization processes, refer to the *FFIEC IT Handbook's* "Information Security" booklet.

IV.P Third-Party Relationship Risk Management

Entities can realize significant benefits from working with third parties such as gaining access to new technologies, human capital, delivery channels, products, services, and markets. However, the use of third parties can reduce an entity's direct control over activities and may introduce new risks or increase existing risks[142] in development, acquisition, and maintenance activities. Increased risk often arises from greater operational or technological complexity, newer or different types of relationships, or potentially inferior performance by the third party. As part of sound risk management, entities engage in more comprehensive and rigorous planning, due diligence, oversight, and management of third-party relationships that support higher-risk development, acquisition, and maintenance activities, including critical activities. Considerations discussed are relevant to each entity's development, acquisition, and maintenance activities, based on specific facts and circumstances, but may not apply to all of an entity's third-party relationships.

IV.P.1 Planning

As part of sound risk management, effective planning allows an entity to evaluate and consider how to manage risks before entering into a third-party relationship. Higher risk and greater complexity development, acquisition, and maintenance services warrant more careful planning and other considerations to mitigate the risk that the relationship will result in material negative impacts on the entity. Refer to the *Interagency Guidance on Third-Party Relationships: Risk Management* and *FFIEC IT Handbook's* "Outsourcing Technology Services" booklet for more information on planning concepts.

IV.P.2 Due Diligence and Third-Party Selection

Conducting third-party due diligence[143] before selecting and entering into relationships is an important part of sound risk management when engaging third parties in development, acquisition, and maintenance activities. Due diligence gives management the information needed to determine whether a relationship would help achieve a banking organization's strategic and financial goals. Third parties that would support an entity's higher risk or more complex development, acquisition, and maintenance activities typically warrant a greater degree of due diligence.

In some instances, the appropriate entity personnel may not be able to obtain the desired initial due diligence information from a third party. For example, the third party may not have a long operational history, may not allow onsite visits, or may not share (or be permitted to share) information that is requested. While the methods and scope of due diligence may differ, the

[142] Refer to the *Interagency Guidance on Third-Party Relationships: Risk Management*.

[143] *Merriam-Webster* defines "due diligence" as "the care that a prudent person might be expected to exercise in the examination and evaluation of risks affecting a business transaction." Due diligence includes assessing the third party's ability to perform the activity as expected, adhere to an entity's policies related to the activity, comply with all applicable laws and regulations, and conduct the activity appropriately.

entity's project management process should identify and document any limitations, understand the risks of such limitations, and consider potential alternatives.[144] Management should evaluate the conclusions from such supplemental efforts based on the entity's own specific circumstances and performance criteria for the activity. Refer to the *Interagency Guidance on Third-Party Relationships: Risk Management* and *FFIEC IT Handbook's* "Outsourcing Technology Services" booklet for more information on due diligence concepts.

IV.P.3 Contract Negotiation

When evaluating whether to enter into a relationship with a third party, an entity's management typically determines whether a written contract is needed; and, if the proposed contract can meet the entity's business goals and risk management needs, then entity management typically negotiates appropriate contract provisions[145] for development, acquisition, and maintenance activities. In certain circumstances, it may be advantageous to negotiate initial development, acquisition, and maintenance contracts with other organizations (e.g., banking associations or user groups). As part of its oversight responsibilities, the board should be aware of—and, as appropriate, may approve or delegate approval of—contracts involving higher-risk development, acquisition, and maintenance activities. Refer to the *Interagency Guidance on Third-Party Relationships: Risk Management* and *FFIEC IT Handbook's* "Outsourcing Technology Services" booklet for more information on contract negotiation concepts.

IV.Q Supply Chain Considerations

Action Summary

Management should implement SCRM policies and procedures consistent with the entity's size, criticality to the financial system, and complexity.

Examiners should review the following:

- Relevant policies, standards, procedures, project plans, and any SDLC documentation to determine how the examined entity has incorporated SCRM into its enterprise-wide risk management practices.
- System and component control configurations and reports used by management to monitor, control, and protect communications at the key access points of information systems that inform SCRM activities.
- Entity's architectural designs, system and component development techniques, and systems engineering principles as well as those applicable to the entity's third parties to maintain effective information security throughout the supply chain.

[144] Refer to the *Interagency Guidance on Third-Party Relationships: Risk Management*.

[145] Ibid.

- System and service inventories to determine how well the inventories identify all systems and services, how well they categorize each system and service's criticality, and how well they identify the entire supply chain needed to provide the system or service.
- Entity's methods for initially assessing and monitoring after implementation for provenance of systems, components, and data.
- Communication processes and documented communication of threat intelligence and vulnerability identification with supply chain partners.
- Entity management's assessment of supply chain partners' programs to address inauthentic or unapproved systems or components (e.g., counterfeit or shadow IT).
- SCRM controls in maintenance-related situations including monitoring for unauthorized modifications, communicating changes, and monitoring for EOL.

NIST defines "supply chain" as "a system of organizations, people, activities, information, and resources, possibly international in scope, which provides products or services to consumers."[146] Supply chains evolve continuously through mergers and acquisitions, joint ventures, and other partnership agreements. Supply chains can be dispersed around the world.

Entities interact with a supply chain when they develop, acquire, and maintain logical or physical systems or components (e.g., computers, servers, and other hardware components, software, or services), and when they engage with a third party for a service. Even when an entity develops its own systems and components and manages its own infrastructure, it will still periodically interact with one or more supply chains when purchasing additional external hardware, software, raw materials, and services. As entities increase their reliance on third parties for systems, components, and services, supply chains are more complex, diverse, and interconnected.

If the entity incorporates systems and components from third parties in their development and maintenance activities, consideration of SCRM in the SDLC is appropriate. Awareness of risks arising from integration with suppliers' systems and components helps with management of associated risks such as compatibility issues, backdoors, or unauthorized access points. Whether developed in-house or by a supply chain partner, additional scrutiny should be applied when custom development of systems and components occurs to ensure proper configuration to meet business and information security needs.

Supply chain risk is defined as "the potential for harm or compromise that arises as a result of security risks from suppliers, their supply chains, and their products or services. Supply chain risks include exposures, threats, and vulnerabilities associated with the products and services traversing the supply chain as well as the exposures, threats, and vulnerabilities to the supply chain."[147] Examples of supply chain risks include counterfeit goods (e.g., unauthorized systems or components sold as authentic goods), substandard materials, loss of functionality, theft of data

[146] Refer to NIST SP 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*.

[147] Refer to NIST Glossary.

or intellectual property (e.g., customer data and logos), and malware inserted into authentic systems and components.

Another supply chain risk is the potential to contract with a third party on the U.S. Department of the Treasury's Office of Foreign Assets Control (OFAC) Sanctions List or located in countries that are on the list. Management should ensure that the entity's supply chain partners are not on the OFAC list.[148] This may include evaluating the third party's ownership structure (including identifying any beneficial ownership whether public or private, foreign or domestic).[149] To accurately determine whether a third party is on the list or affiliated with companies on the list, it is important to consider any nested relationships through common ownership or any subsidiary or affiliate relationships. It is important to understand risks related to supply chains by tracking interdependencies between all parties involved (e.g., owners and third-party service providers, including vendors) in the supply chain.

Supply chain risks increase when supply chain partners are unstable financially or during times of economic, cyber, or logistical disruption to the operations and continuity of key supply chain elements.[150] These partners include third parties and their supply chain partners (also referred to as fourth parties). Some supply chain attackers use indirect paths and trusted relationships to gain access to the entity for disruptive purposes, financial gain, or to obtain intelligence. Attackers often exploit a third-party service provider's supply chain to access multiple other victim businesses (i.e., the service provider's customers) for subsequent attacks.

IV.Q.1 Supply Chain Risk Management

SCRM is the systematic process for managing supply chain risk by identifying susceptibilities, vulnerabilities, and threats throughout the supply chain and developing mitigation strategies to combat those threats whether presented by the supplier, the supplier's product and its subcomponents, or the supply chain (e.g., initial production, packaging, handling, storage, transport, mission operation, and disposal).[151] SCRM builds on traditional acquisition and procurement practices by adding processes to identify, measure, monitor, and control risks throughout the supply chain. SCRM processes provide entity management with information to help supplement third-party risk management activities (e.g., due diligence and contract negotiation) and procurement of systems or components by presenting all elements in the supply chain for evaluation.

[148] Per the U.S. Department of the Treasury's Office of Foreign Assets Control's (OFAC) "Basic Information on OFAC and Sanctions," "U.S. persons must comply with OFAC regulations, including all U.S. citizens and permanent resident aliens regardless of where they are located, all persons and entities within the United States, all U.S. incorporated entities and their foreign branches. In the cases of certain programs, foreign subsidiaries owned or controlled by U.S. companies also must comply. Certain programs also require foreign persons in possession of U.S.-origin goods to comply."

[149] Refer to *Interagency Guidance on Third-Party Relationships: Risk Management*.

[150] Refer to NIST IR 8419, *Blockchain and Related Technologies to Support Manufacturing Supply Chain Traceability: Needs and Industry Perspectives*.

[151] Refer to NIST Glossary.

Management should monitor, control, and protect communications (i.e., information transmitted or received) applicable to supply chain-related activities at key access points (e.g., external boundaries and key internal boundaries). Effective management uses architectural designs, software development techniques, and systems engineering principles to promote effective information security in the supply chain by identifying and mitigating potential SCRM-related risk.

An attempt should be made to identify potential single points of failure among all entities in the entity's supply chain due to the numerous interconnectivities and risks from all partners. Review of documents, such as system and network topologies and process flow diagrams, may help with identification of these single points of failure.

NIST identifies SCRM high-level controls in various documents.[152] Management should consider the following controls in all development, acquisition, and maintenance activities.

- **SCRM policies, standards, and procedures** that
  - Align with other internal and external policies, standards, and procedures (e.g., information security and business continuity management).
  - Instruct staff on how to choose hardware, software, and third-party service providers from an SCRM perspective.
  - Address system and component supply chain traceability.
  - Specify supply chain due diligence for new vendors, third-party service providers, and ongoing supply chain monitoring. The initial assessment should occur during the initiation phase of a project, before new products or services are provided, or new contracts are signed. Due diligence activities should include at least identification and understanding of the criticality of supported function(s), its critical components, and the sensitivity of the information that may be accessible by the supplied system, component, or third party.
  - Identify security standards (e.g., percentage of open-source components allowed and access requirements) for purchased products and supply chain partners. It is important for management to consider starting with applicable, established national and international standards as a baseline for security requirements for the entity's supply chain.
  - Specify information protection processes including what data may be shared, the sharing method, and to whom information is shared (the specific roles). Processes should account for appropriate privacy protections and any dissemination prohibitions, safekeeping, and clearance (e.g., security and access control) requirements.
  - Detail information retention and disposal requirements, especially when sensitive and proprietary information of a supply chain partner is concerned.
- **SCRM plan** contains steps to assess, monitor, and respond to risks associated with supply chain interaction and procurement activities. The plan describes entity and supply chain implementations, requirements, constraints, and implications at the system level. The SCRM

[152] Refer to NIST SP 800-53, rev. 5, *Security and Privacy Controls for Information Systems and Organizations*, and NIST SP 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*.

plan can be stand-alone but is often part of an entity's information security program and third-party risk management program. The plan should address supply chain risks associated with geographic location (e.g., defining acceptable locations). The entity should define roles and responsibilities for personnel (e.g., development, acquisition, and maintenance) to address various risks associated with supply chain activities.

- **General controls and processes** are based on the SCRM plan and entity risk assessments. Management analyzes the risks and designs controls to mitigate risks to an acceptable level as defined by entity senior management and the board. To promote confidentiality, integrity, availability, and resilience throughout the supply chain, controls and processes should include development, information security (e.g., access control and other relevant laws and regulations),[153] procurement, supplier diversity,[154] and contractual provisions (e.g., clauses stating that contractual requirements apply to any subcontractors). Management should provide the following:
  - Appropriate protections in the information systems, components, and networks for data in transit and at rest for the supply chain partners' information, such as source code testing data, blueprints, and intellectual property.
  - Tamper-resistance and detection controls for critical components.[155]
  - Appropriate logging, disparate data correlation, and alerting controls (e.g., access logs, usage pattern reports, and time of day access reports) to obtain evidence in the event of a supply chain compromise.
  - Appropriately segregated and isolated environments between the entity and its supply chain partners to minimize the effect of information leakage across IT environments (e.g., development, testing, and production).
  - Controls over the use of security assessment and monitoring tools within and between IT environments.
  - Logical and physical access controls, such as identification and authentication (e.g., out-of-band validation) for user access to systems and components throughout the supply chain, providing for timely update and periodic review.
  - Current access and authentication lists and appropriate controls to minimize the potential for insider threats as entities develop and acquire systems and components.
  - Formal process for assigning identifiers (e.g., names, numbers, and blockchain) to systems and components (e.g., digital and physical) for management and control throughout the supply chain.
  - Provisions for physical and logical resilience and appropriate implementation of controls in the entity's agreements with supply chain partners.
  - Appropriate monitoring tools and processes at the boundaries between the entity's systems and supply chain partners' systems.

[153] Refer to FTC 16 CFR 314, "Standards for Safeguarding Customer Information."

[154] Management should consider supplier diversity for resilience purposes. If only one supplier is relied upon for products or services and is unable to provide them, business could be interrupted; therefore, additional suppliers should be considered. Conversely, too many suppliers may be difficult to manage, and confidentiality, availability, and integrity or quality of service may be compromised.

[155] Criticality analysis can help determine which components are critical.

- **Inventory** is a foundational building block for supply chain management. The entity may rely on third parties to support its strategy and operational functions. Management should develop, document, and maintain an accurate inventory of third parties that reflects the entity's key supply chain partners. Management determines the third party's criticality based on the systems and components the supplier's products and services support and the entity's level of dependency on the supplier for business function. The inventory should be detailed enough for identifying criticality and supply chain risk for tracking and reporting. The inventory should be periodically reviewed and updated, including
  - Description of the supplied products and services (e.g., model or software version).
  - Programs, projects, systems, and components that use the supplier's products and services (i.e., interconnectivities).
  - Assigned criticality level that aligns to the criticality of the program, project, system, component, or service.
- **Configuration management** allows management to track changes made throughout the SDLC to systems, components, and documentation, which is important for tracking details of changes made (e.g., what changes were made, who made them, and who authorized them). Configuration management processes help identify evidence (e.g., baselines and authorizations) used for investigations of potential supply chain cybersecurity compromise. Configuration management is critical to the entity's ability to establish the provenance of components, including tracking and tracing them through the SDLC and the supply chain. It is important to consider configuration management minimum security requirements for the supply chain. Management should apply appropriate configuration management controls to its own systems and encourage or require the use of comparable controls by all parties in the entity's supply chain through contracts. For more information, refer to the *FFIEC IT Handbook's* "Information Security" booklet.
- **Resilience** should be considered from the beginning of the SDLC and maintained throughout the useful life of the system or component. This aids in cost reduction and efficiency of implementation and includes planning for alternative third-party service providers and vendors of systems, components, and services. Another resilience consideration is alternative delivery routes if the primary one is unavailable, especially when it applies to a critical provider. Management can improve resilience by using platform-agnostic systems and components that allow for portability that will improve resilience throughout the supply chain. Having multiple third-party sources (i.e., heterogeneity)[156] for supply of systems and components can result in alternatives for availability and reduce the potential impact of a supply chain compromise. In those situations, an alternative source of supply allows an entity to more rapidly switch to an alternative system or component that may not be affected. Additionally, having heterogeneous components may help decrease the attack surface and impact of a compromise. This is because all systems and components are not alike; therefore, while they may be exposed to similar risks, they may not be exposed to identical vulnerabilities. Additionally, many risk mitigation solutions used for contingency planning (e.g., alternative storage and processing sites and telecommunication pathways) may have their own supply chains with additional risks. Entity management should understand and mitigate all relevant risks associated with interdependencies throughout the various supply

[156] Heterogeneity techniques include the use of distributed storage and processing, different Oss, virtualization techniques, and multiple sources of supply.

chains potentially affecting the entity. Management should plan for certain scenarios, including

- Unplanned system or component failure and subsequent replacement.
- Planned replacement related to feature improvements, maintenance, upgrades, and modernization.
- Product or service disruption (e.g., loss or degradation of data or operations).

Potential supply chain disruptions affect the entity's operations (e.g., transportation issues, system and component availability problems, or financial difficulties of entities in the supply chain). Management should consider provisions for excess capacity, bandwidth, and redundancy in agreements with supply chain partners and take appropriate mitigation steps, as necessary. For more information, refer to the *FFIEC IT Handbook's* "Business Continuity Management" booklet.

- **Provenance** involves the chronology of the origin, development, ownership, location, and changes to a system or system component and associated data. Provenance may include personnel and processes used to interact with or make modifications to the system, component, or associated data.[157] Effective management documents provenance for systems, components, and data, and monitors for changes in the chain of custody that may increase risk to the entity throughout the SDLC. Management should consider producing software bills of material (SBOM)[158] for applicable and appropriate classes of software (e.g., purchased, open-source, and in-house developed software). SBOMs should be digitally signed using a verifiable and trusted key, enabling entities to establish provenance. For more information on SBOMs, refer to the "Software Bill of Material" section of this booklet.
- **Supply chain partner assessments and reviews** should be part of the entity's third-party and supply chain risk management processes, as appropriate. SCRM assessments should include the supply chain infrastructure (e.g., development and testing environments and delivery systems) and the information systems or components traversing the supply chain and should align with enterprise risk management processes and governance. As part of the risk assessment, the entity should have an accurate inventory of suppliers that identifies their criticality to the business. It is important for entity personnel to consider any information pertinent to the security, integrity, resilience, quality, trustworthiness (e.g., not on the OFAC list), or authenticity of their supply chain partners and products. Management should consistently evaluate supply chain partners consistent with the specific context and purpose for which the assessment is being conducted, selecting additional factors for consideration based on supply chain risk. The quality of information (e.g., its relevance, completeness, and accuracy) relied on for an assessment is an important consideration; therefore, management should document the reference sources.

  If an entity has a PMO, it can help define requirements, methods, and tools for supply chain partner assessments. Irrespective of the entity's project management methodology, effective project management processes can help identify critical components, especially those that are

[157] Refer to NIST Glossary.

[158] NIST Glossary defines "SBOM" as a "formal record containing the details and supply chain relationships of various components used in building software. Software developers and vendors often create products by assembling existing open-source and commercial software components. The SBOM [lists] these components [for a given] product."

used by multiple business lines, functions, systems, and components. For example, effective management considers the following practices:

- Determine whether there is potential foreign ownership or influence and whether the supply chain partner may have relationships with OFAC-sanctioned individuals, organizations, or countries.
- Evaluate supply chain partner oversight of its subcontractors (i.e., fourth parties) or developers.
- Identify the level of open-source systems and components used by the entity, and determine how the supply chain partner demonstrates its compliance with applicable open-source licensing agreements.
- Conduct research on COTS systems and components (e.g., via publicly available resources) or request proof to determine whether the supply chain partner (e.g., original equipment manufacturer [OEM]) has performed testing as part of their quality or security processes.
- Use authorized resellers or distributors with an ongoing relationship with the supply chain partner for systems and components not directly acquired from an OEM entity.
- Acquire directly from vetted OEMs or their authorized distributors and resellers when obtaining alternative sources for continued support. Decisions about using alternative sources (i.e., other than OEMs or authorized resellers and servicers) should consider input from all stakeholders (e.g., business lines, IT, CISO, and compliance).
- Track chain of custody of systems, components, and underlying code as they move throughout the supply chain to minimize the potential for counterfeit or altered products. Use of mechanisms, such as radio frequency identification (RFID), digital signatures, bar code scanning, or blockchain to employ supply chain protection techniques in the acquisition, development, and use of code may be deployed throughout the entity to facilitate tracking chain of custody. These techniques include ensuring that[159]
  - Code (including mobile code) originates from vetted sources when acquired.
  - Vetted system and component integrators are used for the development of custom code before installing.
  - Verification processes are in place for acceptance criteria before installation to verify the source and integrity of code.

- **Audit and assurance (internal)** to validate management’s assessment of risks in the supply chain, including activities to assess an entity’s supply chain partners. Internal audit and assurance activities may include the following:
  - Review of system or component documentation.
  - Review of processes for tracking reports to measure adherence to SLAs.
  - Use of service provider system and organization control (SOC) reports and other independent review reports to validate that supply chain partners meet the entity’s minimum standards and implement timely corrective actions for identified deficiencies.

[159] Refer to NIST SP 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*.

  - Verification of suppliers' claims of conformance to contractual security and compliance requirements.
  - Validation of system or component integrity.
  - Use of validation tools and techniques (e.g., scanning) for detecting malware or counterfeit goods (e.g., unauthorized systems or components sold as authentic goods). Additionally, management may use manual inspection techniques for identifying genuine components.
  - Verification of ongoing training to ensure that appropriate personnel are aware of emerging supply chain risks.
- **Supply chain operational security** is important because of the interconnectedness of numerous relationships with supply chain partners and the difficulty in coordinating multiple operational security efforts, which creates the opportunity for a potential breach. Effective management communicates with its supply chain partners to promote awareness of threat intelligence and relevant vulnerabilities in the entity's supply chain to inform operational security processes. In some situations, entity management may choose to withhold information from supply chain partners to prevent parties with malicious intent from using the information to compromise the entity. To protect against compromise,
  - Management should consider employing techniques to introduce randomness[160] into entity operations and assets in the entity's systems or networks.
  - Management should consider concealment techniques, such as masking metadata that may be accessible in downloads or deliveries of systems and components, whether developed or acquired.
  - If management considers advanced security protection techniques (e.g., misdirection, honeypots) beyond standard industry techniques (e.g., basic randomness and concealment), it should discuss these techniques with the entity's legal counsel and board, as appropriate, before implementation, as they may present liability and additional risk to the entity.
- **Agreements**[161] between the entity and its supply chain partners should include provisions addressing common supply chain risks, including their applicability to any subcontractors. Agreements should address risks, including the following:
  - Personnel[162] security controls (e.g., background checks and screening, termination and transfer procedures, and consequences of insider threats) for individuals with access to the entity's supply chain.

[160] For purposes of this discussion, randomness includes randomly switching among several delivery enterprises or routes or changing the time and date of receiving supplier software updates if previously predictably scheduled. For more information, refer to NIST SP 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*.

[161] Agreements may include contracts, contract addendums, service level agreements (SLA), and operating agreements.

[162] Supply chain-related personnel, whether internal or third-party, may include acquisition and contracting professionals, program managers, supply chain and logistics professionals, shipping and receiving staff, IT professionals, business owners, system owners, and information security engineers.

    - Off-boarding procedures to maintain security of entity or supply chain partner information at the close of a contract or end of service to prevent the misuse of sensitive entity or customer information. Agreements should address relationship termination to facilitate a secure off-boarding process (e.g., removing data from cloud environments).
    - Most recent set of applicable security and functionality requirements as a baseline, with provisions for future enhancement as risk mitigation needs dictate.
    - Processes to monitor and track the vulnerability of the supply chain, based on the criticality or risk profile of the supply chain partner, system, component, or service. A system, component, and service inventory may help management to identify assets that may be subject to known vulnerabilities. This may provide a starting point for determining the level of vulnerability monitoring needed for systems, components, and services throughout the entity's supply chain.
    - Information sharing (e.g., collection, analysis, and distribution) of threat intelligence, incidents, and other key risk indicators throughout the supply chain. The intelligence gathered enables management to proactively identify and respond to threats in the entity's supply chain. Therefore, management should consider information-sharing clauses in agreements and contracts.
    - Procedures and responsibilities of supply chain partners when actionable (i.e., management needs to take action to mitigate the risk in some way) information is learned, indicating that a supply chain partner or a system, component, or service is a target of a specific threat.
    - Provisions for boundary (e.g., network perimeters) protections, which should be incorporated into agreements with supply chain partners, when applicable. This should address initial implementation of boundary control requirements for interconnected networks in the supply chain.
    - Inclusion of protections (e.g., encryption) for sensitive information at rest or in transit, applicable throughout the supply chain. This is a consideration for assessing compliance with Gramm–Leach–Bliley Act (GLBA) provisions.
- **Incident response** agreements should include the entity's notification requirements for imminent threats (e.g., system, component, service, or supply chain partner may be the target of an attack), incidents, and compromises that may affect the entity.[163] It is important for management to consider addressing the following incident response-related information in its agreements with its supply chain partners:
    - Definition and triggers of an incident.

[163] Refer to *Computer-Security Incident Notification Requirements for Banking Organizations and Their Bank Service Providers*, referenced in agency guidance FDIC FIL-74-2021, "Computer-Security Incident Notification Final Rule," and OCC Bulletin 2021-55, "Computer-Security Incident Notification: Final Rule." Implementation is further discussed in FDIC FIL-12-2022, "Computer-Security Incident Notification Implementation"; FRB SR Letter 22-4/Consumer Affairs (CA) 22-3, "Contact Information in Relation to Computer-Security Incident Notification Requirements"; and OCC Bulletin 2022-8, "Information Technology: OCC Points of Contact for Banks' Computer-Security Incident Notifications."

- Roles and responsibilities, including clear boundaries of responsibility, for incident response processes, including third-party expertise.[164]
- Methods and timing of incident communication with regulators and supply chain partners. These should be defined in agreements with supply chain partners to ensure an efficient, coordinated incident response effort (e.g., speed of communication, response, corrective actions, and other related activities).
- Incident response training (e.g., threat briefing or incident response exercises) when appropriate.
- Incident response testing with critical supply chain partners, as appropriate.
- Review of the agreement based on lessons learned after incidents and any new intelligence.

- **Tamper resistance and detection** processes allow entity management to recognize when unauthorized changes to systems and components are made in transit between the supply chain partner and the entity. Entity or supply chain partner personnel should inspect critical systems and components to ensure that tamper-resistant controls are in place and to determine whether there is evidence of tampering. Systems and components should be reviewed for tampering before use and periodically thereafter. Provisions for this type of review should be included in contracts with supply chain partners.
- **System or component authenticity and integrity** is critical for managing cybersecurity risks throughout the supply chain. Without effective SCRM and ITAM processes, it may be possible for entity personnel or third parties to use counterfeit systems and components. This may allow for potential security, functionality, quality, and legal weaknesses (e.g., counterfeit components may void contracts, warranties, or insurance coverage), leading to compromise of confidentiality, integrity, and availability of systems, components, services, and data throughout the supply chain. By deploying effective system and component integrity controls, management can mitigate cybersecurity risks, such as insertion of malicious code and use of counterfeits, throughout the supply chain. Effective controls include anti-counterfeit policies and procedures that protect against introducing counterfeit systems and components into the entity's infrastructure (e.g., ATM processor boards with extra communication chips built into the circuitry, noncertified hardware developers). System and component integrity controls may include the following:
  - System and component integrity policy and procedures, including the use of various integrity verification tools and techniques. These tools and techniques may include the following:
    - Digital signature or checksum verification.
    - Acceptance testing for physical systems and components.
    - Verification testing confined to limited-privilege environments, such as sandboxes.
    - Code execution restrictions in limited-privilege environments before implementation in production.
    - Validation that binary or machine-executable code is obtained directly from the OEM or a verified supplier or distributor.

[164] Roles and responsibilities include identification, determination, escalation, notification, handling, analysis, monitoring and tracking, and provisions for third-party expertise (e.g., forensics).

    - Process to communicate with supply chain partners to identify, report, and correct system and component flaws in a timely manner.
    - Protection from code threats originating from an unauthorized or malicious source in the supply chain. These protections should be applied throughout the supply chain.
    - Monitoring for supply chain system and component security and threat intelligence alerts and advisories from supply chain partners. Effective management takes appropriate actions in response. Monitoring processes should include monitoring that lessons learned from prior compromises are appropriately acted upon. For example, if malicious code implanted during software development leads to a compromise, effective controls are implemented to further ensure code legitimacy in the future. It is important for management to consider
        - Correlation of available threat intelligence information (e.g., internally produced and from supply chain partners) to identify potential threats or vulnerabilities requiring mitigation.
        - Employment of enhanced monitoring over activities performed by higher-risk personnel (e.g., users with elevated authority or privileges).
    - Determining periodically whether any counterfeit or shadow IT systems and components exist in the supply chain. Typically, management addresses this issue through contractual provisions with supply chain partners regarding their mitigation program for shadow IT risk. If any are discovered, management should assess the impact of removal, then remove any suspect systems or components from the entity's infrastructure as appropriate.[165]
- **Maintenance** includes performing system and component updates and replacements and may be performed by a third party. When performed by a third party, maintenance becomes part of the supply chain and SCRM principles should be applied. Management should perform the following when implementing maintenance processes throughout the supply chain:
    - Define agreements that identify roles, responsibilities, and practices that may be used for maintenance activities, especially when third parties are responsible for maintaining the entity's systems or components.
    - Monitor for unauthorized modification or removal of the entity's systems or components (e.g., use of counterfeit systems, counterfeit components, and malware) in the supply chain. Agreements with supply chain partners should address monitoring and communication to affected parties throughout the supply chain.
    - Monitor systems and components for planning for EOL to prepare for replacement or upgrade to the systems and components. Effective management considers any potential availability issues in the supply chain for systems and components or has agreements to continue support for legacy systems or components until a change can be made without significant disruption to operations.
- **System or component EOL and disposal processes** outline the steps personnel should take when components are nearing EOL and need to be decommissioned and removed from the entity's infrastructure. Standard EOL processes should be followed. Additional processes related to supply chain activities may involve the following:

[165] For more information on shadow IT and IT asset management, refer to the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations" booklet.

- Inclusion of data disposal processes to allow entity management to mitigate the risk that entity data maintained by supply chain partners may be retained or compromised.
- Consideration of supply chain partner interdependencies associated with the removal of a system or component from the entity’s environment, as well as throughout the supply chain. For example, the removal of a supply chain partner’s system or component that may disrupt the entity’s ability to provide continuous service to customers.
- Sanitization or destruction of information from supply chain partner systems, components, and media before disposal or release for reuse. Agreements should address this for all locations and providers in the supply chain. Off-boarding procedures should be addressed in agreements to maintain data confidentiality, integrity, and availability until the relationship with that supply chain partner is officially terminated.
- Application of data protection controls (e.g., tracking chain of custody) throughout off-boarding processes for an entity’s data and for the data received from supply chain partners.

Additionally, as a part of SCRM, entity management should consider interconnectivity risks throughout the supply chain. What may initially be assessed as a low-risk activity or asset may be considered a high-risk activity or asset based on the interconnectivity or reliance on that system or component. For example, a system or component (e.g., HVAC, smart TVs, and IoT devices) may not appear high-risk, yet the system or component may be connected to critical systems and components throughout the supply chain, and subject to remote access risks. Therefore, management should have a process to validate the effectiveness of the security of systems and components throughout the supply chain when performing development, acquisition, and maintenance activities, and make appropriate adjustments to risk assessments to account for interconnectivity risk.

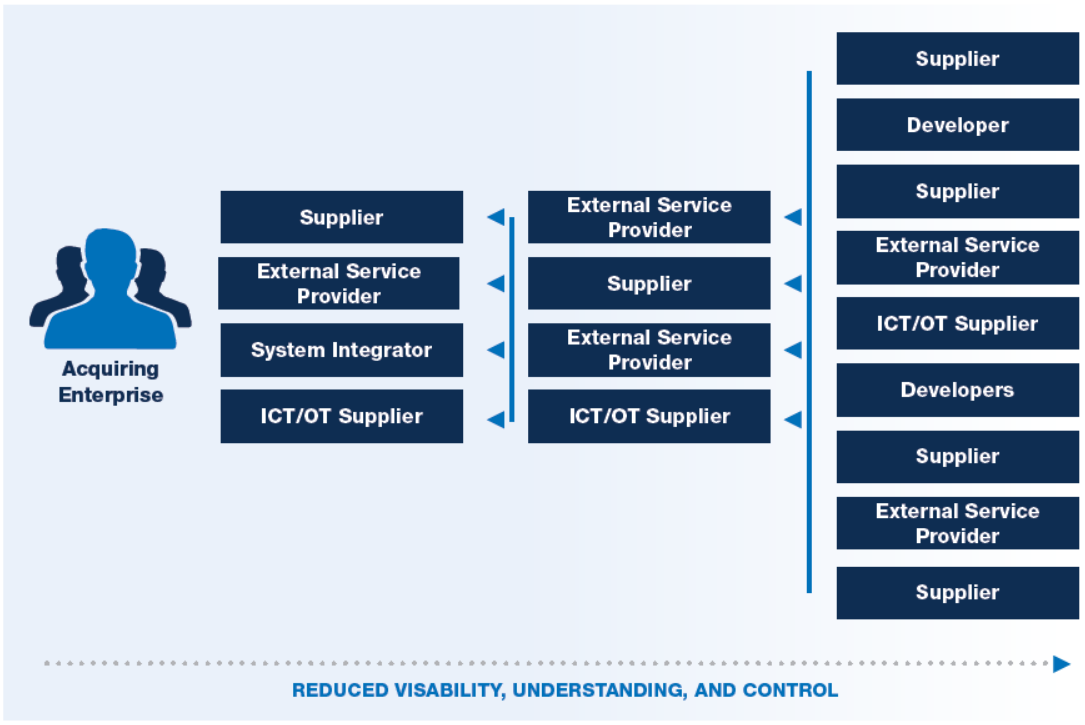
To effectively manage supply chain risks, management should have a clear understanding of interconnectivity in the entity’s supply chain. To facilitate this understanding, management should consider using available information, such as that provided by third-party user groups and associations,[166] which can augment ongoing monitoring and due diligence, threat intelligence, and security throughout the supply chain. A vulnerability to one system, component, or supply chain partner may pose a vulnerability to the entire supply chain. Figure 5 depicts diverse supply chain relationships that affect an entity’s visibility and control of the supply chain. Entities depend on the supply chain to provide a variety of products and services to enable the enterprise to achieve its strategic and operational objectives. As NIST states, acquirers often lack visibility and understanding of how acquired technology is developed, integrated, and deployed and how the services that they acquire are delivered. Additionally, acquirers with inadequate or absent SCRM processes and practices may experience increased exposure to cybersecurity risks throughout the supply chain. The level of exposure to cybersecurity risks throughout the supply chain depends largely on the relationship between the products and services provided and the

[166] Associations may include standards-setting organizations, such as the National Automated Clearinghouse Association (Nacha), and PCI Security Standards Council (for the payment card industry) and information-sharing organizations (e.g., Financial Services Information Sharing and Analysis Center [FS-ISAC], and other information-sharing and analysis centers [ISAC]).

criticality of the missions, business processes, and systems they support.[167] Entities have a variety of relationships with their suppliers, developers, system integrators, external system service providers, and other information communication and technology/operational technology-related[168] service providers.[169]

Effective management considers risk management factors over the entire life cycle of a third-party relationship, including planning, due diligence, contract negotiation, ongoing monitoring, and termination. For more information on these factors, refer to the *FFIEC IT Handbook's* "Outsourcing Technology Services" and "Management" booklets. Additionally, for more information on supply chain relationships, refer to NIST SP 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*.

Figure 5: Supply Chain Relationships


Acquiring Enterprise
Supplier
External Service Provider
System Integrator
ICT/OT Supplier
External Service Provider
Supplier
External Service Provider
ICT/OT Supplier
Supplier
Developer
Supplier
External Service Provider
ICT/OT Supplier
Developers
Supplier
External Service Provider
Supplier
REDUCED VISABILITY, UNDERSTANDING, AND CONTROL

[167] Refer to NIST SP 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*, for more information.

[168] NIST Glossary defines "information and communication technology" as "encompassing the capture, storage, retrieval, processing, display, representation, presentation, organization, management, security, transfer, and interchange of data and information." It defines "operational technology" as "programmable systems or devices that interact with the physical environment (or manage devices that interact with the physical environment)."

[169] Refer to NIST SP 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*.

IV.Q.2 Software Bill of Material

An SBOM is a formal record containing the details and supply chain relationships of various components used in building software.[170] "A primary purpose of an SBOM is to identify components and their relationships to one another, using some combination of baseline component information."[171] There are benefits to supply chain partners including developers, purchasers, and operators of software. An SBOM can help developers of systems and components be aware of any software they might use in development of entity solutions. For purchasers, an SBOM helps identify software supporting a product potentially subject to compatibility issues. An SBOM enables an entity and its operators to determine in a relatively short time whether the entity is affected by a vulnerability and where in the supply chain the entity may have been affected[172] because management can match vulnerabilities to elements in the SBOM.

Management may use an SBOM to identify what systems and components fall outside the entity's risk tolerance thresholds (e.g., has a significant vulnerability, inappropriate use of open-source components, or is linked to an untrusted provider). An SBOM may help with the management of OEM licensing and compliance and enable management to quickly identify interdependencies and other supply chain risks. An SBOM gives management visibility into systems and components used in the entity's infrastructure, including open-source systems and components and IoT products.[173] This visibility allows management to perform activities such as the following:

- Determine whether any of the software or components used in the entity's infrastructure are subject to specific vulnerabilities or potential risks associated with publicized breaches, as determined through entity threat intelligence processes.
- Uniquely identify systems and components used by customers and authorized entities.
- Trace users and the use of licenses, allowing management to determine whether access control information is accurate and whether its use is appropriate.

Tools used to read an SBOM should "uniquely identify individual components in a standard format,"[174] so that all references are uniform throughout the supply chain. Additionally, an SBOM should be updated whenever software elements are changed or software is updated.

[170] Refer to NIST Glossary.

[171] Refer to NIST SP 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*.

[172] Refer to National Telecommunications and Information Administration (NTIA), *Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM)* and "SBOM Myths vs. Facts."

[173] For more information, refer to NIST IR 8425, *Profile of the IoT Core Baseline for Consumer IoT Products*, and NIST IR 8259A, *IoT Device Cybersecurity Capability Core Baseline*.

[174] Refer to NTIA's *Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM)* and "SBOM at a Glance."

SBOMs typically do not identify intellectual property, patents, algorithms, or code; if they do, management should implement appropriate security for this proprietary information.

NIST states that as SBOMs mature, management should help ensure that they do not deprioritize existing SCRM capabilities (e.g., vulnerability management practices and vendor risk assessments) under the mistaken assumption that SBOMs replace these activities. Accurate SBOMs are meant to provide improved transparency, which is complementary to an entity's risk management practices, not a substitute for them. "Entities that are unable to appropriately ingest (i.e., recognize and understand), analyze, and act on the data that SBOMs provide likely will not improve their overall SCRM posture."[175]

IV.Q.3 Enterprise Risk Management and Supply Chain Risks

SCRM touches on risk management activities across the entity and should be incorporated into enterprise and technology risk management activities. A few examples of security elements tied to SCRM include the following:

- **Confidentiality:** Supply chain partners may receive sensitive entity and customer information.
- **Integrity:** Systems or components may not operate as designed, leading to a data integrity issue, or third parties may intentionally or unintentionally change entity, partner, customer, or consumer data.
- **Availability and resilience:** Systems, components, or services may not be available or recoverable when needed.

According to NIST, "Managing supply chain risk is a complex, multifaceted undertaking that requires a coordinated effort across an organization to build trust relationships and communicate with internal and external stakeholders."[176] Supply chain activities occur throughout the SDLC; therefore, SCRM should be embedded throughout the SDLC.[177] Additionally, when applicable, management should incorporate third parties into business continuity and resilience activities throughout the supply chain.

[175] Refer to NIST SP 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*.
[176] Refer to NIST 800-53, rev. 5, *Security and Privacy Controls for Information Systems and Operations*.

[177] Refer to NIST SP 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*.

V DEVELOPMENT

Action Summary

Implementation of sound processes for the development of systems and components supporting the entity's business needs and operations is important.

Examiners should review the following:

- Documentation of training in secure design and coding techniques for those entity personnel who are responsible for development.
- Development standards, including procedures for managing changes and a back-out plan.
- Evidence of communication throughout the supply chain regarding identification of critical systems and components, potential threats and vulnerabilities, configurations, and responsibilities for selection and maintenance of system and network components.
- Entity coding standards and list of entity-approved software development languages.
- Documentation of security certification for completed systems and component-related code.
- Development designs used to validate security and functionality throughout the supply chain.
- Risk assessment documentation related to the use of or switch to a development operations (DevOps) approach, or extending to a development security operations (DevSecOps) approach, and effectiveness of controls.

Development is the systematic application of knowledge toward the production of useful materials, devices, and systems. In this context, it involves the processes of defining, designing, developing, testing, and implementing systems or components. Development includes validation and demonstration of a chosen technology, use of test and production environments, improvement of developed prototypes, integration into systems and subsystems, and hardware builds. Understanding development concepts is important to all entities whether development is performed in-house or by a third party on the entity's behalf. Additionally, development can be a complex topic, which requires the entity's board, senior management, and business line management to understand risks and communicate effectively. For entities that develop or modify their own systems and software, effective management requires training in secure design and coding techniques of those responsible for development. If formal standards are not in place and development processes are not controlled, the following can occur, affecting confidentiality, integrity, availability, and resilience:

- Corruption (e.g., malware or sabotage) of the development process.
- Unintentional creation of vulnerabilities (e.g., backdoors or open ports).
- Omission or errors in process or code (e.g., intended processes that were forgotten or calculations that are incorrect) resulting in production issues (e.g., disruptions or delays).

Entity personnel should maintain and follow documented development policies, standards, and procedures that result in systems and components that meet the entity's requirements for confidentiality, integrity, availability, and resilience. Standards and procedures should identify approved development tools and tool configurations in the development process or in a specific development methodology.

When engaged in development, management may consider using design and coding techniques, appropriate tools (e.g., computer-aided design), and prototypes. When development personnel use design and coding techniques, they should appropriately manage risks (e.g., eliminating inaccurate techniques or outdated tools and prototypes). Prototypes may help illustrate a system's or component's functionality to stakeholders and help management identify, evaluate, and highlight design risks not apparent during the impact analysis or before production. Then the design risks may be resolved more efficiently. There are alternatives to prototyping, such as defining and building a system or component with only enough features to accomplish the essential user objectives of the system.

As part of the SDLC, effective management works toward developing a trustworthy system. From a security perspective, a trustworthy system[178] meets specific security requirements and other critical requirements defined and set by entity management. To help develop a trustworthy system, management should employ secure program coding practices. Secure coding practices help developers minimize the potential for known source code-related vulnerabilities in the development phase.

Examples of secure coding practices include the following:[179]

- Validate all inputs and validate and properly encode all outputs.
- Avoid using unsafe functions.
- Detect and correct errors effectively.
- Provide logging and tracing capabilities.
- Consider use of automated features that encourage secure coding practices in development environments, when appropriate.
- Establish procedures for manually applying secure coding practices when automated methods are insufficient or unavailable.
- Use tools (e.g., linters[180] or formatters) to analyze code, identify common mistakes and bad coding style, and standardize the style and formatting of the source code.
- Check for common system, code, and development environment vulnerabilities.
- Assign developers to review code to complement (not replace) independent code review and verify compliance with security requirements.

[178] Refer to NIST SP 800-160, vo1. 1, rev. 1, *Engineering Trustworthy Secure Systems*.

[179] Refer to NIST SP 800-218, *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities*.

[180] For purposes of this booklet, a "linter" is a type of tool used to perform code analysis to flag issues, such as programming errors, bugs, and stylistic errors.

- Identify and correct vulnerabilities before release of software to production to prevent exploitation.

Developers should communicate throughout the supply chain to ensure that partners are aware of potential threats and vulnerabilities when developing, testing, and maintaining systems and components. Additionally, the individuals responsible for selecting system and network components should incorporate supply chain processes when choosing such components. Developer training should include secure coding and the use of tools to find vulnerabilities in systems and components across the supply chain. For more information, refer to the "Supply Chain Considerations" section of this booklet.

V.A Development Standards and Controls

Management should establish development standards, including procedures for controlling changes during the development process that address the following:

- **System controls**, including a system's or component's functionality, security, automated control features, and coding standards.
- **Quality management** (also referred to as quality assurance and quality control [QA/QC]), including the development and implementation of a system or component so it meets predefined specifications and management's expectations.
- **Release management**, including migration of the final release of systems or components from a development or development testing environment into the production environment for deployment. This may include multiple steps. A release can be completed manually, or the release process could be automated. Established rollback instructions, or back-out plans, should be developed and available in the event of unforeseen deployment problems.
- **Documentation** to facilitate stakeholders' understanding of the following:
  - Development process.
  - Error correction and program modification processes.
  - System or component business functionality.

- **Reporting**, including processes for generating and managing reports (e.g., types of reports, timing, and stakeholders who should receive them) during development to help management make important budgeting and other data-driven decisions. Reports could include project updates, technical issue logs, and change management logs.

Development standards should articulate the entity's minimum development requirements. For example, development standards may address the selection of programming styles, languages, and tools; layout or format of scripted code; naming conventions; and program library requirements. Effective management communicates in policy its reasons for using specific programming styles and languages on a project or service in project documentation. This is done to identify decisions made by management so developers can understand the reasons certain programming styles and languages were used and whether they remain appropriate for current functionality and stakeholder needs. Examples are included in table 3.

Table 3: Descriptions and Reasons for Using Programming Styles and Example Languages

Programming style	Description	Why	Example languages[181]
Compiled	Source code converted into machine language before the code can operate	• Useful when resources are limited and execution speed (for transactions) is essential. • Less portable than hybrid or interpreted languages due to the need to recompile for each new system.	C, C++, GO, Swift, Objective-C, Visual Basic
Hybrid (compiled and interpreted)	Uses a combination of compiled and interpreted languages programming style	• Portable and slower than compiled. • Requires more resources (e.g., CPU and random access memory [RAM]) than compiled, but less than interpreted language.	Java, C#, Clojure, Scala, Spark, Prolog
Interpreted[182]	Lines of code that can operate without first being compiled	• Less complex than compiled or hybrid programming styles, but requires more resources (e.g., CPU and RAM). • Easier to find trained developers, as less technical knowledge is needed to implement the code.	Python, JavaScript, PowerShell, PHP, Structured Query Language (SQL), Ruby, Perl, Lua, Delphi
Assembler	Converts assembly instructions into machine language	• Useful when you need to minimize resources (e.g., CPU and RAM) needed to implement the code. • Often used for firmware.	Assembly
Markup language	Language with rules and instructions accompanying the code, which facilitates use by humans and programs	• Useful when text needs to be formatted in a specific way (e.g., font, color, and size).	HTML, XML, SGML
Legacy	Older programming languages that potentially are unsupported, general use is declining	• Personnel with knowledge of code development and use is declining, making it difficult to maintain and upgrade. • Concerns related to programming style in sunset phase of SDLC.	FORTRAN, COBOL, APL, ALGOL, LISP, PASCAL, Assembly, REXX, JCL

The use of naming conventions[183] for a software program's modules and any related subroutines, databases, or programs that interact with an application help a team of developers work more efficiently together. For example, naming convention use promotes consistency so programmers can link subroutines into a unified program more efficiently. Naming conventions also facilitate understanding so IT staff not involved in the original development can later modify systems

[181] Languages may be applicable in several different language styles; these are just examples.

[182] An interpreted programming style refers to processing a script or other program expressions line by line and in accordance with the language requirements, without first compiling a program into machine instructions.

[183] NIST defines a "naming convention" as "a collection of rules, which when applied to data, results in a set of data elements named in a logical and standardized way. These names inform the user about the contents of the data value domain, and the usage of the data element, in a concise manner."

efficiently. Finally, use of naming conventions often facilitates software code reuse and repurposing.

To establish development standards and controls, prudent practices include thorough development testing and validation of systems and component-related code. Effective management keeps completed systems and component-related code that have passed security certification in program libraries as discussed in the "Additional Control Considerations in Change Management" section of this booklet.

Due to supply chain risk in development, for both in-house and supply chain partners, it is important to consider using the following standards and controls when developing information systems and components:

- Design security controls to be difficult to disable (e.g., tamper-proofing techniques), and, if they are disabled, trigger notification methods such as audit trails, tamper evidence, or alarms.
- Design delivery mechanisms (e.g., downloads for software) to avoid unnecessary exposure or access to the supply chain and systems or components traversing the supply chain.
- Design relevant validation mechanisms to be used during implementation and operation.

Developer configuration management is critical for reducing cybersecurity risks, both in-house and throughout the supply chain. Developers should manage the configurations of the development environments in which they are working. They should create configuration guides for the systems and components they are developing for distribution with the system or component. By implementing configuration management standards and controls, developers reduce the occurrence and likelihood of flaws while increasing accountability and ownership for the changes. Developer configuration management should be performed by developers, including any third parties involved in the process.

If management decides, based on assessments of risk throughout the supply chain, that customized development of certain critical systems and components is necessary, then it should have agreement on standards and controls used in customized system and component development throughout the supply chain. Whether developed in-house or by a supply chain partner, additional scrutiny should be applied when custom development of systems and components occurs to ensure proper configuration to meet business and information security needs. Management should work with suppliers and partners to ensure that critical systems and components are identified. Effective management helps ensure that suppliers or the entity itself has a continued ability to maintain customized systems and components that are critical to the entity's operations. For example, having the source code, build scripts, and tests for a software component could enable an entity to have a third party maintain the system or component if necessary. For more information, refer to the "Escrowed Source Code Agreements and Documentation" section of this booklet.

V.B Testing

Action Summary

Management should maintain testing policies, standards, procedures, as well as other process controls that are effectively used to mitigate risks in internally and externally developed systems and components.

Examiners should review the following:

- Testing policies, standards, and procedures, as well as other process controls, to evaluate how well entity personnel use testing to confirm that systems and components meet the entity’s requirements.
- Testing scope documentation, including for application interoperability and vulnerability discovery.
- Documentation regarding controls over the use of production data in testing.
- Documentation of the type of testing, testing results, corrective actions, and testing completion.
- Documentation of testing for any new controls added to systems and components to avoid conflicts (e.g., integration and interface, security) resulting in failed business processes.
- Any updates to manuals and training plans after testing.

Testing is essential to the development process to promote confidentiality, integrity, availability, and resilience. Whether developed internally or by a third party, appropriate personnel should test systems and components to identify and correct defects before deployment, including security-related defects.

Testing should be sufficient to confirm that systems and components meet the entity’s architectural, functional, information security, and legal/regulatory requirements. To help mitigate the risk that an integration point contains an undiscovered defect, entities test the system or component being developed, including how well that system or component interoperates with other systems and components. This scope is commonly achieved through a combination of code-level, system-level, and code vulnerability testing.

Starting test plans no later than the development and acquisition phase helps teams test more comprehensively than if they wait to create test plans later in the SDLC. Additionally, the level of detail in the test plan and script should match the level of risk the system or component presents to the examined entity’s business. The more detailed test plans and scripts are, the higher the likelihood that testers will identify defects before system or component implementation.

Using production data in testing may provide assurance that the system or component functions as expected; however, there are significant risks (e.g., unauthorized access to or modification of sensitive entity or customer information). Management should determine the need for using production data in testing and employ appropriate controls if its use is deemed necessary. When possible, testers should use simulated synthetic data or sanitized[184] production data during system or component development and testing[185] to protect sensitive customer and entity data. Data used for testing should not include sensitive customer information unless the data are sanitized[186] before testing, and this should be addressed in policy. Due to the risks involved with actual customer data, when sanitizing data is not feasible, management should document an exception to board-approved policy for its use. Implementation and maintenance of controls similar to those used in the production environment should be used to appropriately protect the data for compliance with legal and regulatory requirements. The Information Security Standards require safeguarding of sensitive customer information regardless of the environment (e.g., testing, QA, or production) where the information resides.[187] For more information, refer to the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations" and "Information Security" booklets.

The types of testing[188] performed should be appropriate for the development activity's inherent risk. Effective management has a methodical process to define and conduct testing necessary to demonstrate the effectiveness of a developed system or component. Typically, testing is performed in stages, and knowledgeable testing specialists and relevant stakeholders should be involved in testing. Regardless of the process, various tests may be used individually or in combination at different points in the development and testing phases. During system or component development, developers perform code reviews, which typically occur as part of the testing process. Appropriate personnel can perform testing manually, with automated tools, or a combination of both.

---

[184] Data sanitization disguises (i.e., anonymizes) sensitive information in test and development databases by overwriting the information with realistic looking but false data of a similar type. Some forms of sanitization include NULLing out, masking data, substitution, shuffling records, and encryption. For more information, refer to NIST SP 800-160, vol. 1, rev. 1, *Engineering Trustworthy Secure Systems*.

[185] Refer to NIST SP 800-160, vol. 1, rev. 1, *Engineering Trustworthy Secure Systems*.

[186] Data sanitization can help minimize risks related to unauthorized access of that data. Unsanitized data passed to other systems or subsystems (e.g., command shells, relational databases, and commercial off-the-shelf [COTS] components) may be vulnerable. Attackers may be able to exploit unused functionality to access, manipulate, or exfiltrate the data causing legal, reputation, and operational risk to the entity. Additionally, that data may provide information to attackers, allowing for unauthorized lateral movement throughout the network, systems, and components.

[187] Refer to GLBA, further implemented by FFIEC members as follows: FDIC: 12 CFR 364, appendix B, "Interagency Guidelines Establishing Information Security Standards," and 12 CFR 364, supplement A to appendix B, "Interagency Guidance on Response Programs for Unauthorized Access to Customer Information and Customer Notice"; FRB: Regulation H, 12 CFR 208, appendix D-2, "Interagency Guidelines Establishing Information Security Standards," and Regulation Y, 12 CFR 225, appendix F, "Interagency Guidelines Establishing Information Security Standards"; NCUA: 12 CFR 748, appendix A, "Guidelines for Safeguarding Member Information;" OCC: 12 CFR 30, appendix B, "Interagency Guidelines Establishing Information Security Standards." Additionally, refer to FTC 16 CFR 314, "Standards for Safeguarding Customer Information," for similar provisions for service providers.

[188] For purposes of this section, NIST IR 8397, *Guidelines on Minimum Standards for Developer Verification of Software*, defines "testing" as "any technique or procedure performed on the software itself to gain assurance that the software will perform as desired, has the necessary properties, and has no important vulnerabilities."

While the testing types, such as dynamic testing, may be performed during multiple phases, commonly performed testing in the development and acquisition phase includes the following:

- **Static analysis:** Personnel detect software vulnerabilities by examining the application source code and binary code and attempting to analyze and identify all possible behaviors that might arise at runtime.[189] An example of static analysis is software application security testing, which analyzes the quality of the code.[190]
- **Unit testing:**[191] Developers perform unit tests to assess the functionality of small modules of code.
- **Security testing:** Developmental testing helps ensure that the technical and security features and functions of the system perform as intended.[192] This may include initial penetration testing of the code.
- **Dynamic analysis:** Personnel use testing that operates by executing a program using a set of input use cases and analyzing the program's runtime behavior. Testing may be extensive (i.e., time-consuming) depending on the number of scenarios analyzed. An example of dynamic, execution-based scans is dynamic application security testing, which is intended to simulate real-world attacks.[193]

The development team should document the results of testing performed. Documentation should include the type of testing performed, scope of the testing, test results, further action based on results of the test (e.g., error correction, patch, or retest), and whether the testing is considered complete. Appropriate personnel can provide a report communicating the test results to management as appropriate.

Commonly performed testing in the implementation and assessment phase includes the following:

- **System testing:** Testing is performed on a complete system to evaluate its compliance with specified requirements.
- **Additional security testing:** If new controls are added to the system, additional acceptance tests of those new controls should be performed to help ensure that new controls meet security specifications and do not conflict with or invalidate existing controls or functionality.
- **Integration testing:** Personnel perform an orderly progression of testing in which software elements, hardware elements, or both are combined and tested to evaluate their interactions,

[189] Refer to NIST 800-163, rev. 1, *Vetting the Security of Mobile Applications*.

[190] For more information, refer to *OWASP Vulnerability Management Guide (OVMG)*.

[191] Information Systems Audit and Control Association (ISACA) Glossary defines "unit testing" as "a testing technique that is used to test program logic in a particular program or module. The purpose of the test is to ensure that the internal operation of the program performs according to specification. It uses a set of test cases that focus on the control structure of the procedural design."

[192] Refer to NIST ITL Bulletin, "The System Development Life Cycle."

[193] For more information, refer to *OWASP Vulnerability Management Guide (OVMG)*.

until the entire system has been integrated.[194] A primary goal of integration testing is to ensure that systems and components function appropriately together. This is especially difficult when software uses components developed by different vendors, in different languages, and the implementation sources are not all available.[195]

- **UAT:** Personnel perform testing that involves taking use cases or procedures for how the system was designed to perform and ensuring that someone who follows the procedure gets the intended result; however, it does not necessarily demonstrate how well the system supports users in performing those functions.[196] Therefore, UAT is not used to determine usability. UAT generally occurs toward the end of development testing activities. UAT may be useful for validating procedures and user training on the operation of new or significantly changed systems and components.
- **Regression testing:** Personnel execute testing that involves the performance of the same tests many times to determine whether to return to the prior state. A retest of a system or component helps to assess functionality after programmers make code changes following previous tests. Regression testing generally includes systems that may not have changed but may be affected by a new or changed system or component.
- **Stress testing:** For purposes of this discussion, this refers to testing performed to determine the limitations of a system or component. IT personnel periodically test to determine whether the entity’s system or component can handle the anticipated maximum volume or duration of activity while maintaining required functionality. This may be performed using automated testing tools that generate simulated scenarios.

Regardless of the testing methods used, management should implement practices to document, report, and address identified issues, including security-related issues, in a timely manner. Tracking corrections and modifications resulting from testing facilitates the completeness of the overall program documentation. During the implementation and assessment phase, effective management reviews and finalizes all supporting documentation, such as user, operator, and maintenance manuals as well as any conversion, implementation, and training plans associated with a new release or significant update.

V.C DevOps and DevSecOps

DevOps builds on agile principles and is a software engineering culture and set of practices designed to unify software development and operations. DevSecOps takes DevOps one step further by integrating security (“Sec” stands for “security”). The following subsections of this booklet provide more information on DevOps and DevSecOps.

[194] Refer to ISACA Glossary.

[195] For more information, refer to Leonard J. Gallagher and Jeff Offutt, *Test Sequence Generation for Integration Testing of Component Software*.

[196] Refer to NIST IR 7741, *NIST Guide to the Processes Approach for Improving the Usability of Electronic Health Records*.

V.C.1 DevOps

DevOps is a "set of practices for automating the processes between software development and information technology operations teams so that they can build, test, and release software faster and more reliably. The goal is to shorten the SDLC and improve reliability while delivering features, fixes, and updates frequently in close alignment with business objectives."[197] DevOps is used to promote coordination and communication between development and operations teams (e.g., developers, testers, or operations personnel). For example, a group of developers can be embedded with the operations team to acquire domain-specific expertise, allowing them to better understand applicable business processes in a given area of the entity's systems. The operations team benefits from having developers readily accessible to help operations personnel better understand the entity's developed applications and IT infrastructure.

Using DevOps practices, personnel can automate tasks that were previously performed manually. For example, a DevOps team can automate configuration management processes (e.g., firewall rules, server hardening, and mobile device management). DevOps practices often include containerization, which allows applications to be implemented as microservices. For more information on containers and microservices, refer to the "Common Development, Acquisition, and Maintenance Risk Topics" section of this booklet.

Automation and orchestration give the DevOps approach several benefits including shorter development cycles, increased deployment frequency, faster time to market, and more stable operating environments. In the past, entities may have had changes that took place on a weekly basis; however, in a DevOps environment, the same number and type of changes can be made daily or even hourly. However, automation of previously manual activities, combined with frequent and rapid release cycles, may result in perpetuation of errors or invalid code. Management should assess the risks involved in using a DevOps approach and implement appropriate controls. Potential risks include the following:

- Misconfigured process automation (e.g., quality and release management).
- Unclear governance practices, such as inadequate development standards and procedures and a lack of validation of effectiveness.
- Inappropriate metrics or tracking measures.
- Unauthorized code access and modification via unsecured code repositories (e.g., GitHub).
- Lack of cyber hygiene (e.g., inappropriate configurations or unscanned and vulnerable systems are released).
- Developers being able to use uncontrolled and insecure system images.
- Inadequate controls over code implemented in production (e.g., bypassing the entity's coding standards).

V.C.2 DevSecOps

DevSecOps is the evolutionary progression from DevOps that integrates security objectives and controls (e.g., metrics and compliance reports) into the development process. The main

[197] Refer to NIST SP 1800-16, *Securing Web Transactions: TLS Server Certificate Management*.

characteristics of this practice are to automate, monitor, and apply security at all phases of the SDLC. Automation of the workflows and integrated security testing is a key attribute of DevSecOps. Additionally, automation of testing (e.g., unit testing, code analysis, and image scanning) provides timely feedback to DevSecOps teams while not obstructing productivity. Cybersecurity and design have the same priority as they would in traditional system and component development. Security resources are integrated at the outset of the development process. DevSecOps is a key approach used in cloud development environments. Cloud-native applications require updates and deployment techniques to be flexible and secure for business reasons, as well as resilient to respond to cybersecurity events.[198]

Coding services and tools can be used to automate and implement security faster in the development process, allowing for quicker implementation. NIST notes that DevSecOps can be used with the following types of coding and coding services:[199]

- **Application code:** Code used for one or more business functions, made up of code describing the business transactions and any database access.
- **Application services code (e.g., service mesh code):** Provides various services for the application, such as service discovery, establishing network routes, network resiliency services (e.g., load balancing, retries), and security services (e.g., enforcing authentication and authorization based on policies).
- **Infrastructure as code:** Expresses the computing, networking, and storage resources needed to run the application in the form of declarative code.
- **Policy as code:** Contains declarative code for generating the rules and configuration parameters for realizing security objectives through security controls (e.g., authentication and authorization) during runtime.
- **Observability as code:** Code that triggers automation of software related to logging all transactions, tracing communication pathways involved in executing application requests, and monitoring applications during runtime.

As infrastructure and development activities have moved to a cloud-based environment, the use of the term "as code" is often added to the type of technique. Further, "when using 'as code' techniques, the code that is written (e.g., for provisioning a resource) is managed like application source code. This implies that it is versioned, documented, and has access controls defined similarly to what is done for an application source code repository."[200] In addition, some on-premises hardware components may be replaced and deployed as software in cloud-based environments.

[198] Refer to NIST SP 800-204C, *Implementation of DevSecOps for a Microservices-Based Application With Service Mesh*.

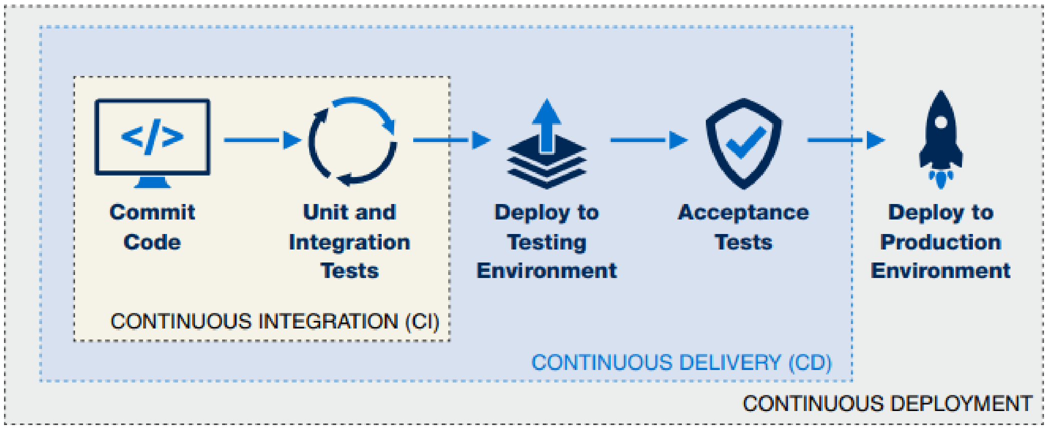
[199] Ibid.

[200] Ibid.

DevSecOps makes use of the continuous integration (CI)[201] and continuous delivery (CD)[202] or continuous deployment[203] pipeline (see figure 6) during the SDLC, depending on whether the entity uses automation to move code into production. The U.S. Department of Defense (DOD) defines this CI/CD pipeline as "the set of tools and the associated process workflows to achieve CI/CD or continuous deployment with build, test, security, and release delivery activities, which are steered by a CI/CD orchestrator and automated as much as practice allows."[204]

Developers digitally sign code in the CI/CD pipeline process. This helps with determining code integrity. For example, downstream evaluations can compare the expected code author's name to the name of the person who last changed the code and take appropriate action if they do not match.

Figure 6: CI/CD Pipeline Process Workflow


Commit Code
Unit and Integration Tests
Deploy to Testing Environment
Acceptance Tests
Deploy to Production Environment
CONTINUOUS INTEGRATION (CI)
CONTINUOUS DELIVERY (CD)
CONTINUOUS DEPLOYMENT

During CI (as illustrated above), developers frequently merge code changes into a central repository where automated builds and tests run.[205] CI activities include committing code (i.e., establishing

[201] Continuous integration (CI) involves developers frequently merging code changes into a central repository where automated builds and tests run. Build is the process of converting the source code to executable code for the platform on which it is intended to run. When developers use CI to deploy new software, the developer's changes are validated by creating a build and running automated tests against the build. This process avoids the integration challenges that can happen when waiting for release day to merge changes into the release branch. For more information, refer to NIST SP 800-204C, *Implementation of DevSecOps for a Microservices-Based Application With Service Mesh*.

[202] NIST defines "continuous delivery" as "the stage after CI when code changes are deployed to a testing or staging environment after the build stage." For more information, refer to NIST SP 800-204C, *Implementation of DevSecOps for a Microservices-Based Application With Service Mesh*.

[203] NIST defines "continuous deployment" as "similar to continuous delivery except that the releases happen automatically, and changes to code are available to customers immediately after they are made." For more information, refer to NIST SP 800-204C, *Implementation of DevSecOps for a Microservices-Based Application With Service Mesh*.

[204] Refer to DOD's *DoD Enterprise DevSecOps Reference Design: Version 1.0*.

[205] For a definition of CI, refer to NIST SP 800-204C, *Implementation of DevSecOps for a Microservices-Based Application With Service Mesh*.

version control, in which a snapshot is taken of the current code changes and saved to a repository), developers packaging up code (i.e., bundling of related code) or revisions to code, and automated initial code testing (such as unit and integration testing).

Management often uses CI processes to put the entity's apps into containers directly in the build process itself. Also, during CI, code is uploaded into a central repository where automated tools validate and initially test the code. The purpose is to avoid inconsistency from the very beginning of an app's life cycle to help ensure consistency in the app's runtime environment (i.e., the environment in which a program is executed). Many organizations may see developers build in one environment, test in another, and deploy in a third, so having consistency in assessment and enforcement across these environments is key.[206]

Once the code is initially validated and tested, CD automatically deploys code to the formal testing environment for acceptance testing. Within the CD segment illustrated above, code is deployed to production manually, using scripts to assist in the configuration. In continuous deployment, rather than manually, deployment occurs when the entity automates the deployment process, and code is available to users immediately.

NIST documents the following resources and tools used in the CI/CD pipeline process: [207]

- **Pipeline software.**
  - CI software: Pulls code from a code repository for building the software, facilitates initial testing tools, and saves tested artifacts to an image registry (i.e., repository for copies of code to maintain coding integrity).
  - CD or continuous deployment software: Pulls artifacts from the repository and deploys the package.
- **Development software.**
  - Build tools: Used for constructing software (e.g., for integrated development environments).
  - Testing tools (e.g., software application security testing, dynamic application security testing, or software composition analysis).
- **Repositories.**
  - Source code repositories (e.g., GitHub).
  - Container image repositories or registries.
- **Observability or monitoring tools.**
  - Logging and log aggregation tools.
  - Tools that generate metrics.
  - Tracing tools (e.g., sequence of application calls).

[206] Refer to NIST SP 800-190, *Application Container Security Guide*.

[207] For more information, refer to NIST 800-204C, *Implementation of DevSecOps for a Microservices-Based Application With Service Mesh*.

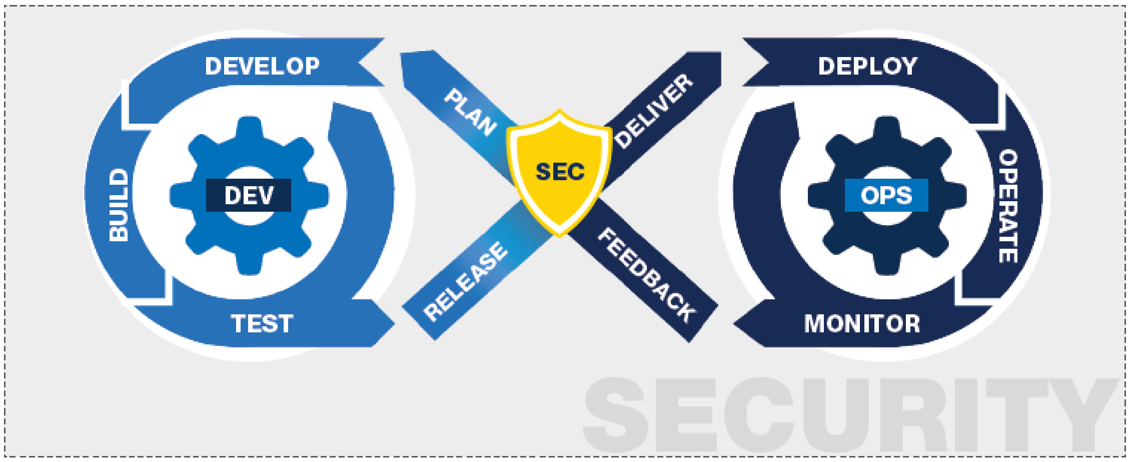
  - Visualization tools (e.g., used to combine data from all development activities to generate dashboard/alerts).

NIST recommends the following controls to secure the CI/CD pipeline process:[208]

- Harden servers hosting code and artifact repositories.
- Secure credentials (e.g., authorization tokens) used for accessing repositories.
- Implement controls on who can check in and check out artifacts in container image registries. These registries store artifacts produced by the CI pipeline and serve as bridges between the CI and CD pipelines.
- Log all code and build update activities.
- Send build reports to developers and stop further pipeline tasks when a build or test fails in the CI pipeline. Code repositories should be configured to automatically block all subsequent pull requests from the continuous deployment stage of the pipeline until issues are resolved.
- Send build reports to the security team and stop further pipeline tasks when an audit fails.
- Ensure that developers can only access the application code.
- Digitally sign (preferably multiparty digital signing) the release artifact during each required CI/CD stage during the build and release process.
- Verify that all required digital signatures are present during production release to ensure that no one bypasses the pipeline.

Figure 7 provides an overview of the CI/CD pipeline. Automated security testing is performed throughout development phases and throughout production.

Figure 7: CI/CD Pipeline Architecture[209]



[208] Refer to NIST SP 800-204C, *Implementation of DevSecOps for a Microservices-Based Application With Service Mesh*.

[209] Refer to DOD's *DoD Enterprise DevSecOps Reference Design: Version 1.0*.

V.D Functional Development Types

Functional development types include hardware, software, front end, and backend. Examples of hardware developed for the financial sector are point-of-sale terminals, mobile card readers, credit cards, and managed security devices. Examples of software developed for the financial sector are mobile banking apps, online banking systems, loan estimators, core banking systems, transaction monitoring systems, and imaging systems. Development of either hardware or software may include a combination of backend and front-end development types. Backend refers to the parts of systems that are not seen by the end user. Front end refers to the parts of systems that the end user sees and uses. Understanding that there are different functional development types is important for development teams as each development type may have unique procedures (e.g., steps and timing), include various reporting lines (e.g., internal and external stakeholders, including management), and address different risks (e.g., physical and virtual) depending on the project.

Functional development types can include the following:

- **Hardware development:** Hardware development includes integrating hardware with software to provide a user experience that supports the business use cases. This includes designing and building the physical aspects of a hardware component. Hardware may include servers, circuit boards, credit cards, ATMs, point-of-sale terminals, and mobile reader devices. The hardware developer designs the system, selects components, builds prototypes, and coordinates testing. The developer generally has in-depth knowledge of software development because developing hardware requires using complex design tools and programming skills. Hardware development includes incorporating firmware and producing circuitry concepts and circuit diagrams, which are then put into operation. The developer designs circuit board layouts and programs microchips and microcontrollers. They develop and simulate digital signal and image processing. They are responsible for maintenance and testing as well as preparing manufacturing documents and product documentation. Hardware developers are sometimes referred to as electronics developers or hardware designers.
- **Front-end (or client-side) development:** Front-end[210] development can be facilitated through hardware or software-based solutions. A front-end developer works toward providing an effective end-user experience. This includes anything (e.g., text and layout, touchscreen, and icons) the user interacts with on their devices (e.g., PCs, tablets, mobile phones, and ATMs). Developers combine design, technology, and programming to create user interfaces that provide clear and understandable products with end-user appeal. Another front-end consideration is the importance of integrating the business line product rules (e.g., rules for negative interest rate changes or loan-to-value limitations) into the user interface. Front-end development may include web development, desktop development, mobile development, and graphics development.

[210] *Merriam-Webster* defines "front end" as "a software interface (such as a graphical user interface) designed to enable user-friendly interaction with a computer."

- **Backend development (or server-side development):** Backend[211] development can be facilitated through hardware or software-based solutions. Backend development includes developing databases, internal APIs, business logic, microservices, web integration, servers, and server-based utilities. Users interact directly with the front end to communicate and engage with the backend, which is not visible to end users. Backend developers should understand front-end development and vice versa. Without that understanding, the risk of noncompliance with laws and regulation increases. For example, financial institution customers rely on loan calculators to provide accurate results relevant to their input scenario. Visible and correct results are only possible if the user interface is clear (front-end development) and the calculations are accurate (backend development).
- **Big data development:** Big data development focuses on using, organizing, and maintaining large amounts of data that may originate from many sources. Big data development includes building data mining, data visualization, and quantitative analysis tools. In addition to developing specific tools, big data developers work with analysts to help them decide whether a set of data are appropriate for their analytics. Big data development can result in maintaining petabytes of complex unstructured, semi-structured, and structured data without rigid schemas. When appropriately controlled, big data can enable cost-effective processing of massive volumes of data, support multiple concurrent queries and other analytics tasks, and rapidly generate results. However, if big data environments are not adequately controlled, costs may increase rapidly, and the examined entity may rely on inaccurate information for business decisions. For additional information on big data, refer to the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations" booklet.

The following subsections of this booklet provide more information on additional development types, including model development and database development.

V.D.1 Model Development

For purposes of this discussion, "model" refers to a quantitative method, system, or approach that applies statistical, economic, financial, or mathematical theories, techniques, and assumptions to process input data into quantitative estimates.[212] A model consists of three components: an information input component, which delivers assumptions and data to the model; a processing component, which transforms inputs into estimates; and a reporting component, which translates the estimates into useful business information. Models meeting this definition may be used in the following:

- Analyzing business strategies.
- Informing business decisions.
- Identifying and measuring risks.
- Valuing exposures, instruments, or positions.
- Stress testing.

[211] *Merriam-Webster* defines "backend" as "the part of a software system that is not usually visible or accessible to a user of that system."
[212] For purposes of this discussion, the term "model," is used as described in joint agency guidance, FRB SR Letter 11-7, "Guidance on Model Risk Management," and OCC Bulletin 2011-12, "Sound Practices for Model Risk Management: Supervisory Guidance on Model Risk Management."

- Assessing capital adequacy.
- Managing client assets.
- Measuring compliance.
- Maintaining the formal control apparatus of the bank.
- Meeting financial or regulatory reporting requirements.
- Issuing public disclosures.

V.D.2 Database Development

Databases are digital repositories of data designed to be accessed, managed, and updated. They can be internally developed or purchased from third parties. Commonly used database models include hierarchical, networked, relational, object-relational, and nonrelational. Developers should understand the different types, structures, and uses of databases to effectively mitigate database development risks. To do this, it is important to understand the benefits, limitations, and appropriate controls for the type of database used. Examples of risks developers of databases should consider include the following:

- Data errors and omissions.
- Inappropriate database access privileges.
- Misuse of legitimate privileges.
- Database injection attacks (e.g., SQL injection).
- Malware.
- Storage media exposure (e.g., backups).
- Exploitation of vulnerable databases (e.g., failure to patch and maintain the database).
- Unmanaged sensitive data (e.g., failure to maintain an accurate database inventory that includes each database’s content and the content’s sensitivity).
- Human error (e.g., failure to apply security controls, enforce policies, or implement appropriate incident response processes).

Potential mitigation strategies include the following:

- Maintaining a database inventory that includes the database location(s), contents, and sensitivity.
- Assessing databases for any vulnerabilities.
- Managing user access rights (e.g., removing user access and privileges when access is no longer required for business purposes).
- Logging and monitoring database access and usage to identify patterns and help detect data leakage, unauthorized or significant activity, and system and component attacks in real time.
- Blocking inappropriate web requests.
- Backing up the databases.
- Encrypting databases.
- Training database developers on their security-related development responsibilities, including identification of potential information security risks and appropriate mitigation practices.

Prudent practices include choosing databases that meet business and other stakeholder objectives and implementing controls, including access controls, to protect the data consistent with their criticality and sensitivity. Additional database design and selection considerations are flexibility, scalability, data integrity, and retrieval efficiency.

VI ACQUISITION

Action Summary

Management should establish acquisition processes that are commensurate with the entity's business and procurement needs. Management should also assess and mitigate procurement risks associated with an entity's overall strategic, regulatory, and operational risks.

Examiners should review the following:

- Project requests and plans for one or more recent acquisitions.
- Strategic and business plans, risk assessments, due diligence activities, and acquisition-related meetings with minutes.
- Contracts and licensing agreements.
- Acquisition policies, standards, and procedures, including evaluation criteria and documentation maintenance practices.
- Acquisition training and entity guidance for stakeholders.
- Acquisition and procurement-related audits.

NIST defines "acquisition" as "all stages of the process of acquiring a system, product, or service, beginning with the process for determining the need for that system, component, product, or service and ending with contract completion and closeout."[213] While acquisition and procurement are terms often used interchangeably, for purposes of this booklet acquisition refers to the overarching process and procurement refers to the specific steps to obtain the system, product, or service. To effectively manage the risks inherent in acquired systems, components, products, or services, entity personnel identify key roles and responsibilities. For example, once management completes contract negotiation and takes control of the system, component, product, or service, management determines the activities related to and responsibilities for oversight (e.g., entity's third-party or enterprise risk management office).

An entity may acquire technology or technology services from a third party, rather than developing it in-house. An entity may use or acquire systems, components, products, and services from various sources, including internal development teams, technology and software development firms, co-sourced arrangements, open-source repositories, and commercial sources. When an entity purchases systems, components, products, or services, effective management ascertains whether the product specifications are "fit for purpose" and meet the entity's requirements, whether purchasing directly from the OEM partners or a secondary market. This should be addressed in agreements with supply chain partners.

The acquisition of system, components, products, and services (versus in-house development) poses certain risks at each step of the acquisition process that should be assessed and mitigated. Risks associated with the third-party product or service may vary depending upon the specific

[213] Refer to NIST Glossary.

situation. These may include strategic (e.g., the system does not align with business needs), regulatory (e.g., the solution does not consider GLBA), and operational risks (e.g., the provider does not have an adequate testing environment or sufficient resources to maintain the solution). In addition, there may be third-party risk management concerns,[214] weaknesses in security controls, or other operational weaknesses at the third-party provider.

An entity may mitigate the risks associated with acquisition in several ways. The following are examples of effective risk mitigation:

- Develop policies, standards, and procedures to effectively carry out the entity's procurement processes aligned with the entity's acquisition activities.
- Perform potential supplier due diligence reviews.[215] Due diligence rigor should align with a risk assessment that identifies the criticality of the business function supported, the information sensitivity the system or component stores or processes, business function complexity, and external factors such as the supplier's location and ownership.
- Implement contract and licensing processes for all acquisitions, particularly for more complex relationships. An entity's processes should include contract and license review to help ensure that the rights, responsibilities, and accountability of each party are clear.

A structured procurement process (such as the procurement steps in figure 8) is likely to produce more consistent and sustainable results across an entity over time. The more critical the system, component, and service being procured the more important that each procurement step is successfully completed.

In the example in figure 8 below, the procurement process begins with the identification of an entity's business need for procuring a system, component, or service. Then management may engage in the following:

- **Planning.** Planning what information will be needed to procure the best system, component, or service, and requesting information from the project team and key stakeholders.
- **Defining requirements.** From planning and information provided, clearly defining what requirements are necessary.
- **Solicitation.** Requesting quotes or proposals.
- **Evaluation and selection.** Evaluating proposals and determining the best choice.
- **Contract negotiation and award.** Determining contract requirements and responsibilities, and which risks will not be covered in contract.

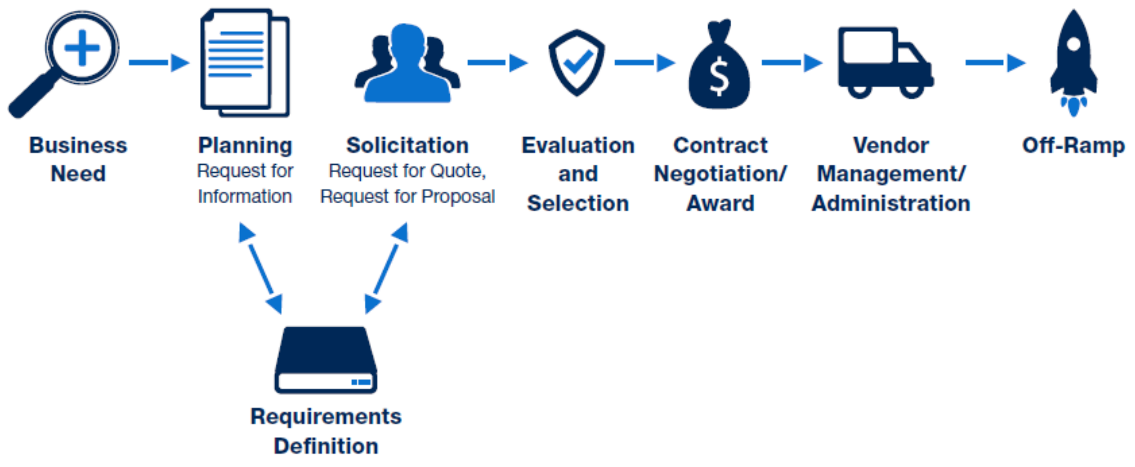
---

[214] Refer to the following guidance: *Interagency Guidance on Third-Party Relationships: Risk Management*, also noted in OCC Bulletin 2023-17, "Third-Party Relationships: Interagency Guidance on Risk Management"; FDIC FIL-29-2023, "Interagency Guidance on Third-Party Relationships: Risk Management"; and FRB SR Letter 23-4, "Interagency Guidance on Third-Party Relationships: Risk Management." Also see CFPB Compliance Bulletin and Policy Guidance 2016-02, "Service Providers."

[215] Refer to NIST SP 800-53, rev. 5, SR-3, *Supply Chain Controls and Processes*, notes that rigorous due diligence includes research on potential suppliers or products, as well as their upstream dependencies (e.g., fourth- and other parties beyond third-party suppliers), which can help enterprises avoid single points of failure in their supply chains. The results of this research can be helpful in shaping the sourcing approach and refining requirements.

- **Vendor management and ongoing administration.** Oversight and maintenance of the relationship, if ongoing.
- **Preparing for the off-ramp.** EOL or upgrade of system, component, or service.

Figure 8: Procurement Steps: Example



An effective procurement process helps management and personnel to address each step in the process, communicate and meet combined stakeholder requirements, select appropriate systems and components, and support the entity's overall business objectives. Additionally, an effective process helps minimize confusion (e.g., by identifying minimum requirements) and promotes coordination in the supply chain (e.g., by achieving efficiencies using one OEM for multiple products).

Specific procurement risks to address include the following:

- Actual costs being materially higher than budget.
- Procured system, component, or service does not meet all stakeholder's requirements.
- Due diligence does not reveal a material third-party deficiency.
- Procured system does not interoperate with existing systems.
- Procurement process informality leads to inconsistent system, component, and service selection over time.
- Contract or license does not specify characteristics critical to system, component, and service success.
- Lack of visibility into supply chain partners for identification of weaknesses (e.g., security, interoperability, or incident response).

VI.A Acquisition Policies, Standards, and Procedures

Acquisition policies, standards, and procedures that are enforced create acquisition consistency across the entity. Well-communicated and enforced policies, standards, and procedures are important for the acquisition process to help determine the appropriate technology for new IT procurement requests or requests for significant updates to current systems and components. Without adequate policies, standards, and procedures, management may expose the entity to the following:

- Procurement of a system or component that does not address the business need.
- Products or services that fail to protect the confidentiality, integrity, availability, and resilience of the entity's systems and components, which may adversely affect other systems in the entity's supply chain.
- Noncompliance with legal and regulatory requirements (e.g., Information Security Standards, Bank Service Company Act).
- Inadequate contracts and licensing agreements, due to failure to communicate and involve key stakeholders (e.g., IT, information security, privacy, legal, and compliance personnel).

An effective entity's acquisition policies, standards, and procedures address the following:

- Information requests, such as requests for information (RFI) and requests for proposals (RFP) to third parties. Responses to these requests provide the entity with vital information for identifying the best provider solution. Requests for quotes (RFQ) are also used before final selection to obtain the best pricing. For additional information on RFIs, RFPs, and RFQs, refer to the "Solicitation" section in this booklet.
- Third-party due diligence[216] includes the following:
  - Comprehensively assessing risks associated with third-party relationships.[217]
  - Reviewing system, component, and service operational and internal controls (e.g., security and resilience controls).
  - Evaluating and selecting (including criteria for selection) a third party.
  - Checking references (e.g., through external user groups or banking association members) to ascertain current and past customer views on product and service quality.
- Specialist stakeholders' (e.g., subject matter, legal, and compliance experts) review of proposal.
- Contract negotiation to ensure that the contract contains terms amenable to the entity.
- Transferability or portability of systems, components, products, and services if a third-party relationship is discontinued.
- Contract signature authority.
- System, component, and service validation to ensure that user requirements are met before acceptance.

In some cases, management may acquire systems, components, or services from foreign-based third parties. In addition to information security risks, there are several unique risk considerations in these relationships, including the following:

[216] Refer to the following guidance: *Interagency Guidance on Third-Party Relationships: Risk Management*.

[217] The risk assessment typically includes a third party's reputation, personnel background checks, customer service issues, financial condition, stability, and fourth-party products or services.

- **Legal.** Confusion over which country's laws will control the relationship. Additionally, it is important to consider that there may be specific U.S. laws with which the entity and its third-party service providers may need to comply (e.g., the restriction on the export of software applications employing encryption techniques).
- **Country.** The possibility that economic, social, or political conditions and events in a foreign country will adversely affect an entity's financial and business interests (e.g., financial defaults by obligors in a foreign country, nationalization of private assets, government repudiation of external indebtedness, exchange controls, or significant currency devaluations).
- **Currency (or exchange rate).** The risk that arises from the change in price of one currency in relation to another. Assets or business operations located across national borders are exposed to and affected by currency risk, creating unpredictable profits and losses, currency devaluations, and foreign exchange controls.
- **Geopolitical.** Risk associated with wars, terrorist acts, and tensions between states that affect the normal and peaceful course of international relations. For example, regime change in the country of operations may result in a loss of control over the entity's data stored in that country.
- **Resilience.** The risk that operations and assets may be undermined or destroyed by crises. This includes the capacities and resources of business systems affected by risks, stresses, and shocks. For example, if there is a disruption of utilities in a country, it may disrupt data center operations.

Management should be knowledgeable of these risks and their effects on the business before entering foreign-based third-party relationships. Therefore, effective management identifies, plans, and addresses the root causes of foreign-based risks or crises to minimize their effects on the entity's operations.

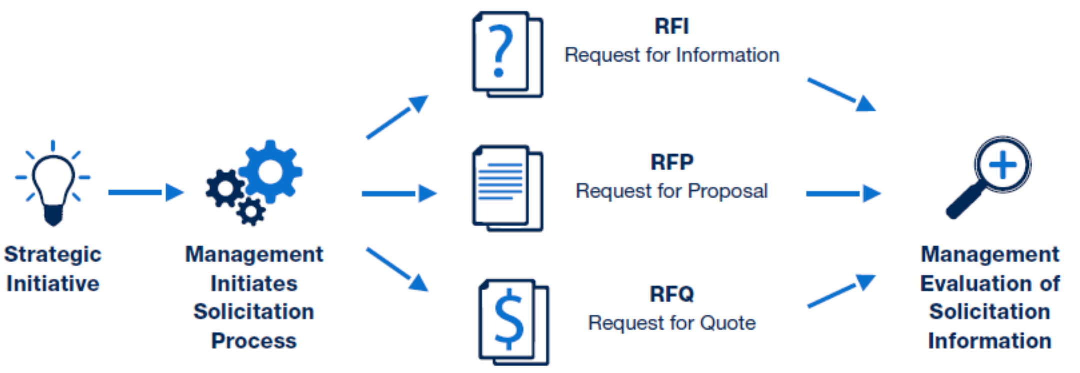
VI.B Acquisition Projects

If an entity has a PMO, acquisition projects may be managed according to established entity PMO processes. Smaller or less complex entities may not have the same formality or rigor in their acquisition project processes. However, it is still important to have project guidance and procedures for the management of acquisition projects. If a procurement project request is approved, the feasibility study should clearly define the IT, information security, functional, legal, and regulatory requirements in the RFIs and RFPs that management distributes to third parties in the solicitation process for bids, including RFQs. Acquisition projects should follow the same principles as noted above throughout section IV.N. "IT Project Management."

VI.C Solicitation

For procurement projects, management follows the entity's requirements definition with solicitation for information and evaluation. These solicitations do not represent binding agreements between the entity and a third party. An entity's solicitation process generally is initiated when it has a business need or problem to solve. Management may issue RFIs to determine potential third parties with solutions available to address the business need or problem and gather information. When the entity has gathered the appropriate information and narrowed down the number of potential third parties, management may issue an RFP to ask those third parties to propose specific solutions. This helps management evaluate and prioritize third parties and their solutions. When management has identified the third parties that can meet the business needs, it uses the RFQ to validate specific requirements and ask whether the third party is able to meet those requirements, and if so, to list all associated costs to help management determine its final selection. For smaller or less complex entities, these terms and steps may not be as clearly delineated or defined; however, management should still have a process that considers business needs, third-party solutions, specifications, and costs. A solicitation to a potential third party may involve an RFI, RFP, or RFQ, or a combination of the three if the entity chooses to move forward with a particular third party on a project, as shown in figure 9 and described in more detail below.

Figure 9: Third-Party Solicitation Types and Inputs


Strategic Initiative
Management Initiates Solicitation Process
RFI
Request for Information
RFP
Request for Proposal
RFQ
Request for Quote
Management Evaluation of Solicitation Information

- **RFI:** A request for information is a market research tool also referred to as a sources-sought notice that is used to obtain, for example, price, delivery, capabilities, and interest for planning purposes.[218] Management may use an RFI to identify third parties that should be targeted for more specific information or further requests. The RFI includes the entity's requirements and appropriate questions that will allow respondents to provide necessary information for stakeholders to make acquisition decisions. Third parties receive requests and provide information about their capabilities, including products or services they offer, and interest in establishing a relationship. There is no standard format for an RFI; however, a formal, well-crafted RFI may facilitate faster responses from potential third parties by providing an overview of the entity's requirements, including any specific questions for the third party. The RFI should clearly identify the entity's needs to avoid a third party's confusion with the request and omission of critical information needed for decision-making. When management receives information from the third parties, assumptions should not made based on the responses. Instead, if appropriate, management should consider asking follow-up questions to clarify concerns or solicit more information.
  Additional considerations when reviewing third parties' RFI responses include the following:
  - RFI format: Whether the provider followed the format specified in the RFI.

[218] For more information, refer to U.S. General Services Administration's (GSA), "RFP, RFI, and RFQ: Understanding the Difference."

- Third-party interest: Whether the third party is interested in forming a relationship.
- Business size: Whether the third party has the resources (e.g., personnel and infrastructure) to support or provide resilience for the suggested solution.
- Location: Proximity of the third party's location to entity locations and effect on resilience.
- Qualifications: Whether the third party's personnel have appropriate knowledge and experience, as well as whether the entity has the personnel and training to operate the solution.
- Subcontractor involvement: Third-party standards for subcontracted work, responsibilities and liabilities for subcontractors, third-party due diligence on subcontractors, and percentage of work to be performed by subcontractors.
- Conflicts of interest: Whether doing business with this provider causes conflicts with other entity third-party providers, internal conflicts of interest at the entity (e.g., a relative of entity personnel works at the third party), or anticipated conflicts of interest with the third parties.

- **RFP:** A request for proposal is a solicitation method that communicates the entity's requirements and requests proposals.[219] An RFP is a formal procurement activity to solicit bids. When drafting an RFP, it is critical to communicate an entity's key system and component requirements, especially those that may incur significant costs to implement. An RFP can include a solicitation for quotes. Bids in response to an RFP provide information needed to select a provider. Information such as the description of the bidder's approach, price, and personnel expertise are helpful in differentiating between bidders to make a final selection.

  Additional considerations when reviewing third parties' RFP responses include the following:
  - Terms and conditions that can help management identify limitations (e.g., length of agreement, resource ceilings, and potential interruptions in service) that the third party may have.
  - Whether a third party follows the RFP format and provides the required information (e.g., cost, experience of the third party, and resilience), which indicates the third party's attention to RFP requirements.
  - Whether the third party's responses are detailed, specific, and responsive (e.g., timely and with no exceptions to the solicitation).

- **RFQ:** A request for quote is a solicitation method used to obtain price, cost, delivery, and related information from suppliers.[220] An RFQ can provide more precise information for bidder comparison and budgeting. There is a potential risk in the RFQ process that bidders misunderstand when an entity is bound by an agreement and a contract is created. To

[219] For more information, refer to GSA, "RFP, RFI, and RFQ: Understanding the Difference."

[220] Ibid.

mitigate this risk, an entity should include legal counsel to help guide the engagement and help avoid potential contractual issues. For additional information on contracting,[221] refer to the "Contracts and Other Agreements" section of this booklet.

For more information on third-party risk management, refer to the *FFIEC IT Handbook's* "Outsourcing Technology Services" booklet.

VI.D Evaluation

A formal process for evaluating RFI, RFP, and RFQ responses produces higher quality selections of third parties across the entity and reduces legal risk. The process goal should be a fair and consistent approach to evaluating and eventually selecting a third-party provider. The process should include procedures to compare each proposal or quote to the entity's defined requirements for the systems, components, or services needed. Then, the entity can compare all the proposals or quotes that meet those requirements. When the process includes pre-determined evaluation criteria, the process is more equitable and legal risk is reduced. Careful consideration helps management determine appropriate evaluation criteria that assists in the review of the widely varying proposals (e.g., bidders proposing either SaaS, infrastructure-as-a-service [IaaS], or platform-as-a-service [PaaS])[222] to meet the basic requirements. Table 3 lists evaluation criteria examples.

Table 3: Evaluation Criteria Examples for Third-Party Solicitations

General criteria for all products and services:

• Ability to audit or receive independent audit reports. • Confidentiality and privacy standards. • Copyright standards. • Delivery dates. • Liability limitations. • Licensing restrictions. • Maintenance procedures. • Financial performance and condition. • Information security requirements set by the entity. • Backup and resilience options. • Costs, fees, and discounts.	• Operations and internal controls. • End-of-service or EOL. • Legal and regulatory requirements. • Subcontractor details. • Testing standards. • Third-party service provider customer reference. • Training provided. • Warranty specifications. • Event or incident response roles and responsibilities. • Insurance considerations.

Software products and services:

• Compatibility of Oss. • Escrow criteria.	• Next release date. • Programming language.

[221] Refer to the following guidance: *Interagency Guidance on Third-Party Relationships: Risk Management*, as noted in OCC Bulletin 2023-17, "Third-Party Relationships: Interagency Guide on Risk Management"; FDIC FIL-29-2023, "Interagency Guidance on Third-Party Relationships: Risk Management"; and FRB SR Letter 23-4, "Interagency Guidance on Third-Party Relationships: Risk Management." Also see CFPB Compliance Bulletin and Policy Guidance; 2016-02, "Service Providers."

[222] Refer to the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations" booklet for more information on SaaS, IaaS, and PaaS.

Hardware products and services:

- Maintenance requirements (for the entity and the third party).
- Capacity (e.g., memory, bandwidth, storage, power).
- Servicing options.
- Performance capabilities.

For more information on evaluation considerations, refer to the *FFIEC IT Handbook's* "Outsourcing Technology Services," "Architecture, Infrastructure, and Operations," "Information Security," and "Management" booklets.

VI.E Contracts and Other Agreements

Contracts and other types of agreements are used to document the terms and conditions under which systems, components, and services are provided. Acquisition policies, standards, and procedures specify how contracts and other agreements are documented, reviewed, and approved. Agreements[223] take many forms and may be included in a contract by reference. Agreement examples are statements of work (SOW), master services agreements (MSA), SLAs, and escrow agreements. To ensure that contracts and other agreements are legally enforceable, legal counsel should be involved in the acquisition process before finalization. Contracts are intended to be legally binding agreements between two or more parties. They generally identify the goods and services to be provided, the time frame for provision, and the associated costs and fees. There may be several documents involved in the procurement process; therefore, it is helpful to identify each of the agreements and contracts, which will assist management in completing the evaluation and selection steps. These documents may be useful to management in its effective oversight of the performance of the entity's supply chain partners. The following subsections describe several agreements and contracts commonly encountered during examinations.

VI.E.1 Statement of Work

An SOW is one supporting document for contracts, detailing the activities to be performed and the activity output in system and component development (e.g., performance level, operating reports, developer documentation, and user manuals). This documentation is critical to system and component maintenance because it details maintenance standards for operations. SOWs can also specify the security assurance requirements (e.g., code reviews, penetration tests, and SOC reports). The security assurance requirements should include the processes that the developer and other staff will follow. Additionally, the SOW may specify the verification process for delivery of products and services, including what task completion evidence is required for payment. SOWs often become part of a binding agreement, such as a contract.

While the contract defines the legal aspects of the relationship, the SOW generally contains elements, such as the following:

- Purpose of the relationship.

[223] For purposes of this booklet, *Black's Law Dictionary* defines "agreement" as "the consent of two or more persons concurring, the one in parting with, the other in receiving, some property, right, or benefit."

- Project objectives.
- Services and products to be provided (i.e., scope of work).
- Schedule, including deliverables and delivery deadlines.
- Requirements for providing the desired services and products, including potential penalties or payment retention for unsatisfactory work.
- Objective measure of satisfactory completion of work.
- Payment due date and method.
- Legal and regulatory requirements.
- Location and duration of the work.
- Acceptance criteria.

There is no standard SOW format across entities and industries. It is up to appropriate stakeholders at the examined entity to determine the format according to the entity's needs. Effective entities provide guidance to relevant personnel in support of a structured approach to SOW development to promote consistent and positive acquisition results, often achieved through SOW training. Similarly, involving all stakeholders in SOW development and establishing an appropriate management review and approval process are both effective practices in achieving positive acquisition results consistently across an entity.

SOW details are important because the type, duration, and complexity of work varies significantly. Clear and specific SOWs help increase the likelihood that bids are comparable, that selection decisions are made quickly, and that miscommunication and misunderstanding are avoided. For example, the SOW should identify the responsibilities of all parties involved. One risk is that service and outsourcing providers can become so focused on the customer entity's needs and requirements that they overlook their own. In order to meet requirements and provide for delivery of a quality product or service, it is important that the provider's needs and requirements (e.g., price, security, and interoperability) also be specified and met. Another example of effective SOW details includes the documentation required from supply chain partners.[224] When an SOW is specific, it can be used to measure whether requirements have been met to an acceptable level.

An SOW may include elements, such as the following:

- Statement of confidentiality.
- Description of work with assumptions and constraints.
- Services to be provided.
- Facility and security requirements.
- Marketing requirements.
- Transition requirements.
- Training and reporting requirements.
- Roles and responsibilities.
- Schedule.

[224] Refer to NIST SP 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*, and NIST's Response to Executive Order 14028, *Software Supply Chain Guidance: Software Security in Supply Chains*.

- Pricing.
- Approvals.
- Glossary.
- Appendix (e.g., containing an incident response plan with contractor responsibilities).

In addition to SOWs, there are alternative methods[225] to document details for proposals (e.g., RFI, RFP, and RFQ), which support the contract or other agreements. One example of these alternatives includes a statement of objectives (SOO), which identifies the broad business outcomes and objectives for performance.[226] An entity makes a request for detailed proposals (i.e., an SOO), which are provided in response. Another alternative is a performance work statement (PWS),[227] which emphasizes outcomes, desired results, and objectives at a more detailed and measurable level. An appropriate response is furnished by prospective providers. These alternative methods may be used in conjunction with the other documents noted above (e.g., RFI, RFP, and RFQ). Information from the SOO or PWS may then be included into the contract to produce a more effective engagement.

VI.E.2 Master Services Agreement

MSAs are "master" documents that govern multiple agreements or transactions between entities. The MSA often sets forth the principal legal terms that govern the parties' relationship and are usually drafted by the service provider. An MSA usually governs or incorporates additional contract documents, such as software license agreements, policies and procedures, terms and conditions, data protection agreements, end-user license agreements, and SOWs. These additional contract documents provide important information to operations personnel.

An MSA helps the parties change contracted services more quickly than without an MSA, because services do not have to be renegotiated each time there is a mutually agreed-upon change to underlying services. For example, a financial institution and a third-party core banking platform product or service provider may enter into an MSA that covers the core banking platform and the possible use of related software and services (e.g., payments software, consulting services, and cloud storage platform). Therefore, changes to the listed services noted in the MSA could occur without creating a new agreement.

MSAs typically include items such as the following:

- Service overview.
- Terms of agreement (e.g., payment, audit, subcontracting, or confidentiality).
- Proprietary rights.
- Intellectual property rights.
- Insurance and taxes.
- Representations and warranties (e.g., patents and trade secrets).

[225] Refer to GSA, "PWS, SOO, SOW - Finding the Best Fit," for more information.

[226] Refer to GSA, "Respond to a Solicitation," for more information.

[227] Refer to GSA, "Performance Work Statement," for more information.

- Limitation of liabilities (e.g., a force majeure clause).[228]
- Dispute resolution process.
- Agreement termination process.
- Legal jurisdiction (e.g., country, state, federal, or local).

VI.E.3 Service Level Agreement

An SLA is a formal agreement between two parties that records a common understanding about products or services to be delivered and how they will be delivered, including the following:

- Priorities, responsibilities, guarantees, and warranties between the parties.
- Nature, quality, security, availability, and scope.
- Timeliness of delivery and response of the parties.
- Point(s) of contact for end-user problems.
- Metrics to monitor the effectiveness of the process.
- Approvals.
- Other measurable objectives.
- Expected day-to-day situations and unexpected or adverse events.

An SLA represents a commitment between a third party and one or more customers and addresses specific aspects of the service, such as responsibilities, details on the type of service, expected performance level (e.g., reliability, acceptable quality, and response times), and requirements for reporting, resolution, and termination.[229]

SLAs are an effective practice to mitigate the risk of differing performance expectations. When procuring services related to a system or component, agreements should have clear and measurable expectations for the services provided, recourse when expectations are not met, and accountability for both parties. SLAs generally include how the third party expects to meet the SLA requirements and establishes incentives for the third party to meet, or penalties for failing to meet, those requirements. The SLA should be linked to clauses in the contract regarding incentives, penalties, and contract termination to protect the entity in the event of third-party performance failures. For more information, refer to the *FFIEC IT Handbook's* "Outsourcing Technology Services," "Management," and "Information Security" booklets.

VI.E.4 Contracts

Contracts between an entity and its third parties should clearly describe the rights and responsibilities of the respective parties. Contracts are formal binding documents, usually preferable to agreements, which can be informal and may not be enforceable in a court of law.

[228] A contract or agreement clause pertaining to "force majeure" (i.e., "act of God") risk, irresistible force, or uncontrollable event (e.g., war or civil unrest) that refers to an event or circumstances "when business is disrupted due to a factor beyond control." This type of clause allocates the risk of loss if performance is hindered, delayed, or prevented because of an event that the parties could not have anticipated or controlled. It helps provide a contractual defense, the scope and effect of which depends on the express terms of a particular contract.

[229] Refer to NIST SP 800-47, rev. 1, *Managing the Security of Information Exchanges*.

An example of important rights to specify are license rights[230] when the entity is using systems and components developed by third parties. Another example is rights and liabilities relating to leased hardware necessary to operate related software. The consequences for failure to meet contractual obligations should also be specified. Examples of software development and maintenance-related contract clauses may include the following:

- Software functional and performance specifications.
- Development process milestones and associated payments.
- Development roles and responsibilities for the third party and the entity.
- SLA information and pricing methods.
- Controls over foreign based third-party activities (i.e., activities performed by entities subject to international laws and regulations).
- Software modification restrictions.
- Objective pre-acceptance performance standards to measure the software's functionality. For example, the contract may identify particular tests that will be used to determine whether the software complies with the contract's performance and security standards.
- Training specifications.

Examples of hardware development or maintenance-related contract clauses may include the following:

- Hardware may not be altered from OEM status, as-delivered (i.e., alterations may void the contract or warranty).
- Proprietary hardware remains the property of the vendor or service provider, and access to it is limited.

A third-party's contributions to the mitigation of major risks are particularly important to specify in contract clauses. For example, security and privacy standards to be followed by third-party service providers or vendors that process, store, or transmit sensitive customer data should be specified in the contract. Table 4 has additional examples of contract clauses.

Table 4: Contract Clause Examples

Clause	Considerations
Legal and regulatory requirements	Outlines responsibility for maintaining software so functionality complies with applicable laws and regulations.
Representations, warranties, and indemnifications	Express representations, warranties, and indemnifications including provisions such as: • Licensed software does not infringe on the intellectual property rights of any third parties worldwide.[231]

[230] Typically, the developing entity provides licenses to the entity that is using the systems or components, granting it certain rights for their use.

[231] Under some state laws, noninfringement warranties are limited to the United States unless otherwise specifically stated.

Clause	Considerations
	• Warranties[232] that software performs according to specifications and how a third party responds in the event of problems. • Specifications for the length of the warranty and how the warranty relates to maintenance obligations and agreements. • Representations that software does not contain undisclosed restrictive code or automatic restraints not specifically authorized in the agreement.
Third-party liability limitations	Proposed limitations of liability in the contract should be reasonable based on the amount of loss or disruption the entity might experience. For mission-critical systems or components, broad exculpatory (i.e., cleared from fault) clauses limiting a third party's liability could adversely affect the safety and soundness of an entity (e.g., excessive force majeure clauses and nested third-party activity exclusions that exceed the entity's capital risk parameters).
Payments	Payment schedule and expense details (e.g., installation, conversion, maintenance, and travel). A payments clause may include provisions such as: • Partial payments at specified development milestones, with final payment due after successful system and component acceptance tests. Properly defined milestones can break development projects into segments with deliverables, allowing management to monitor developer progress and identify any potential problems. • Determination of compensation (e.g., based on the developer's time and materials versus a fixed-price agreement with specific payment milestones). Clearly defined compensation information allows for management control over the development process and total project costs. • Consequences if the third party fails to meet any of the requirements stipulated in the contract, such as failure to deliver system or component features and functions. The contract should allow for the right to reject the deliverable and withhold payment until the third party meets all contractual requirements.
Dispute resolution	Dispute resolution clauses specify the process to be used to resolve problems in a timely manner. Dispute resolution clauses may specify that system and component development will continue during a dispute resolution process.
Agreement modifications	These clauses mitigate the risk that the agreement is not modified without appropriate approval because agreement modification may affect pricing or result in necessary modification to systems and components.
Bankruptcy[233]	Clauses that protect the examined entity in the event of third-party bankruptcy. For example, a clause may specify a software code escrow process that would provide the entity access to the code (that is ordinarily accessed only by the third party) in the event of the third party's insolvency. A second example is a clause specifying that the entity will have direct access to cloud-based services and production data used by the third party for the entity's benefit in the case of the third party's bankruptcy.
Information security	Clauses that identify the information security requirements (e.g., compliance or performance standards) and align with the entity's information security program. Refer to the *FFIEC IT Handbook's* "Information Security" booklet. Examples are: • Clauses that specify the third party's ongoing responsibilities for maintaining the information security and confidentiality of an entity's resources and data. Provisions should address prohibitions on the third party and its contractors or agents using or disclosing an entity's information, except as necessary to provide contracted services. • Clauses guaranteeing that third party-developed systems and components do not and will not permit unauthorized or unknown access (e.g., "backdoors") to the system, component, or the entity's systems or data.

[232] Warranties generally address events and distinguish between mission-critical failures, which should receive an expedited response, and noncritical failures, which management can resolve in a routine manner. It is helpful to review warranties compared to the latest independent certification report demonstrating the software's capabilities.

[233] Refer to relevant provisions of the U.S. Bankruptcy Code.

Clause	Considerations
	• Clauses that specify the third party's actions in a breach, such as reporting requirements regarding the incident, its resolution, and communication timeliness. For more information on incident response, refer to the *FFIEC IT Handbook's* "Information Security," "Outsourcing Technology Services," and "Business Continuity Management" booklets. • Legal, regulatory, and entity standards. For example, applicable regulatory standards include GLBA[234] for financial institutions and the Federal Trade Commission's (FTC) *Standards for Safeguarding Customer Information* for financial institutions subject to the FTC's enforcement authority.[235]
Change in control	Clauses addressing contract revision after third party change of control (e.g., merger, acquisition, or change of ownership). This is important as service quality and support may not be maintained after a change in control.
Subcontracting	• Clauses specifying that the third party is responsible for the performance and security of the systems and components regardless of who designed or developed them. Some third parties may subcontract with other parties (referred to as subcontractors or fourth parties) to develop systems and components for their customers. • Notification and approval requirements regarding changes in subcontractors. Management should consider contract clauses that allow for a contract to be revisited if a key subcontractor changes, which may affect critical entity services. For more information, refer to the *FFIEC IT Handbook's* "Outsourcing Technology Services" booklet.
Audit	"Right to audit" clause allowing for periodic audits of critical third parties' operations. Periodic audits help validate that third-party service providers comply with the agreement's control terms such as the entity's information security requirements.
Continuing vendor support	Clause regarding system and component support including upgrading, patching, and bug fixing through EOL. The clause should include a minimum time frame for notification of the EOL date.

Appropriate entity personnel should be involved in contract drafting to help ensure that contracts cover all relevant aspects of the development, acquisition, and maintenance of the entity's systems, components, and services, including those that are customized for the examined entity. Contracts should contain such details as performance specifications, source code accessibility, software and data security, and hardware viability and replacement. Contracts should identify the recourse available to the entity if the vendor fails to meet defined requirements and user acceptance criteria. For more information, refer to the *FFIEC IT Handbook's* "Outsourcing Technology Services" booklet.

Before opening negotiations or issuing an RFP for customized systems and components, management should define objectives and understand the entity's present and planned IT

[234] Refer to GLBA further implemented by FFIEC members as follows: FDIC: 12 CFR 364, appendix B, "Interagency Guidelines Establishing Information Security Standards," and 12 CFR 364, supplement A to appendix B, "Interagency Guidance on Response Programs for Unauthorized Access to Customer Information and Customer Notice"; FRB: Regulation H, 12 CFR 208, appendix D-2, "Interagency Guidelines Establishing Information Security Standards," and Regulation Y, 12 CFR 225, appendix F, "Interagency Guidelines Establishing Information Security Standards"; NCUA: 12 CFR 748, appendix A, "Guidelines for Safeguarding Member Information"; OCC: 12 CFR 30, appendix B, "Interagency Guidelines Establishing Information Security Standards," (especially the requirement under Section III, D.2, for national banks and federal savings associations to "[r]equire its service providers by contract to implement appropriate measures designed to meet the objectives of these Guidelines").

[235] Refer to FTC 16 CFR 314, "Standards for Safeguarding Customer Information."

architecture, infrastructure, and operations. During contract negotiation, entity management may encounter situations in which third parties cannot or will not agree to the terms an entity sets out in the draft contract. Under such circumstances, management should determine whether it is willing to accept or can mitigate the risks related to systems or components without the requested terms. Any accepted risk or compensating control factors should be clearly documented with appropriate approval consistent with the entity’s policies and governance structure. If management cannot accept or mitigate the risk, it is important to consider alternative solutions. After contract negotiation or agreement execution, periodic audits help to verify that the third party is meeting its contractual responsibilities, such as agreed-on security controls and compliance with laws and regulations.

When negotiating contracts or agreements, management should anticipate that it may decide to convert to different products or providers in the future. Contracts or agreements should enable and facilitate conversions (e.g., minimize restrictions or provide appropriate assistance) to new systems or components. For example, when converting to a different product at the same provider, management and the provider should facilitate coordination between the old and new product teams and entity personnel should be aware of any differences among the products (e.g., hardware, interfacing software, and storage). Additionally, clauses should provide the ability to convert data to a new format as needed. Finally, if converting to a new provider, clauses that allow the old and new providers to coordinate in the conversion process may mitigate conversion related issues (e.g., incompatibility, integration problems, or lack of data availability).

VI.E.5 Escrowed Source Code Agreements and Documentation

Programs are written using proprietary or open-source code. Proprietary systems and components are normally copyrighted trade secrets of the company that wrote or owns the programs. Generally, to protect the integrity of code and software copyrights, vendors do not release proprietary code to the entities that buy or lease the products. Typically, an independent third-party escrow agent retains the source code and related documentation in escrow;[236] however, entity management is responsible for verifying annually that the third party maintains a current version of the source code in escrow. Some escrow agents provide services for reviewing and confirming source code version numbers and dates. Some agents perform automated code reviews to confirm integrity of the escrowed code.

The documentation held by the escrow agent should include system narratives, system flowcharts, program source listings, program narratives, file and record layouts, descriptions of individual fields in the records, and calculation routines. Portions of this documentation may be included with the user guides that are provided to the entity. These documents should cover transaction codes and descriptions of input forms and output reports. In addition, management should consider protecting the entity’s escrow rights by contractually requiring third parties to inform the entity if the software vendor pledges the software as loan collateral. It is important to consider the risks associated with a foreign-based escrow agreement.

[236] *Merriam-Webster* defines “escrow” as “a deed, a bond, money, or a piece of property held in trust by a third party to be turned over to the grantee only on fulfillment of a condition.” For purposes of this booklet, the code in escrow is held by a third party and provided to the entity only on the fulfillment of some condition.

Effective escrow agreement provisions may include the following:

- Requirements to include minimum code documentation with the code in escrow.
- System and component maintenance procedure documentation to be escrowed.
- Conditions that should be present before an entity can access the source code and related documentation.
- Assurances that the escrow agent will hold current versions of the source code and related documentation to validate that escrowed information is updated whenever significant program changes are made.
- Arrangements for auditing or testing escrowed code integrity.
- Source code descriptions and related documentation including the storage type and location (e.g., magnetic tape or cloud).
- Assurances that the source code storage type and location and related documentation is accessible, operable, and compatible with an entity's existing IT environment.
- Assurances that the source code can be compiled into executable code.
- If the escrow agent is based outside the United States, consideration of the practical and legal implications of establishing foreign-based escrow arrangements.

VI.E.6 Exit Strategy

An exit strategy is an entity's plan for transitioning to a replacement system, component, or service, or for terminating a system, component, or service. Entities design exit strategies to minimize disruption. Having a well-defined exit strategy is part of ongoing, effective strategic planning. During the procurement process, management should develop an exit strategy that at a minimum addresses the third party not meeting the contract terms.

To have the most leverage, entity personnel should negotiate default and termination language in the initial contract. The default and termination language may provide opportunities or methods to remedy contract defaults. Contract clauses should define termination rights; document time frames for transferring services to another third party or bringing the service in-house; and provide for how data, records, and IT assets are preserved, transferred, or destroyed.

Default and termination language may provide for the initial steps in executing an exit strategy, but it is important to consider identifying alternative third parties that provide similar products and services at the onset of the relationship. This could provide management with an initial plan to execute a transition between third parties. For more information on exit strategy planning, refer to the *FFIEC IT Handbook's* "Management," "Outsourcing Technology Services," and "Business Continuity Management" booklets.

VII MAINTENANCE

Action Summary

Management should establish policies, standards, and procedures for maintaining systems and components that include detailed roles and responsibilities. It is important that policies and procedures effectively ensure the availability and continued operability of systems and components.

Examiners should review the following:

- System and component inventory adequacy for key information (e.g., to facilitate the entity identifying and addressing vulnerabilities).
- Maintenance plans, including maintenance schedules and logs, vendor and developer recommendations, cost-benefit analyses, operational risk assessments, and qualified staffing adequacy.
- Change management processes, including change types, authorizations, and related risk assessments.
- Security monitoring reports (e.g., anomalous activity).
- Configuration management practices to validate appropriate implementation and enforcement of established change processes.
- Vulnerability and threat identification processes, including remediation plans.
- Any maintenance-related reports to the board of directors and senior management.
- IT asset inventory, including considerations for EOL, such as planned obsolescence and planning for upgrades to legacy systems and components.
- Termination, disposal, and off-boarding processes of systems, components, and change in third-party relationships.
- Maintenance documentation for any system, component, and configuration updates.

The maintenance[237] and operations[238] SDLC activities are necessary whether systems and components have been developed internally or by a third party. They include the routine servicing and periodic modification of everything needed for a system or component to run correctly including the network, hardware, system software, databases, and the related documentation. An entity's maintenance policies and procedures should include maintenance roles and responsibilities (which may include third-party roles and responsibilities), maintenance access policy (internal and remote), and maintenance monitoring and auditing mechanisms.

[237] As noted in the FFIEC Glossary, "maintenance" is "any act that either prevents the failure or malfunction of equipment or restores its operating capability."

[238] As noted in the FFIEC Glossary, "operations" are "the performance of activities comprising methods, principles, processes, procedures, and services that support business functions." IT operations include the tactical management of technology assets and daily delivery of services to capture, transmit, process, and store transactions and information that support the entity's overall business processes.

Well-managed modifications can improve a system's or component's functionality, confidentiality, integrity, availability, and resilience. Periodic modifications may address user functional requirements, correct security vulnerabilities, enhance performance, or increase storage.

An accurate and complete IT asset inventory, including system and component maintenance, is central to ITAM. For example, an inventory that contains system and component versions and patch levels allows an entity to quickly determine how a newly discovered vulnerability affects the organization. Another use of the IT asset inventory is to determine which systems are affected by components reaching their EOL. IT asset inventories should include IoT products.[239] For more information on ITAM, refer to the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations" booklet.

VII.A Preventive Maintenance

Preventive maintenance is the activity to proactively address situations that may cause issues or disruptions. For example, an entity may review a system's storage utilization periodically to determine whether storage is adequate to prevent related outages.

Management should establish a preventive maintenance plan, especially for mission-critical systems and components. When developing the preventive maintenance schedule, it is important to consider system or component criticality, developer maintenance recommendations, operational risks, and other relevant factors (e.g., emerging technologies and threats). These maintenance plans should be aligned with business plans. Appropriate entity personnel schedule maintenance activities during time frames that minimize the risk of interruption to business processes such as after business hours, during weekends, and during holidays.

System and component access for maintenance activities should be limited to only that part of the system needed to perform the maintenance and only for the time necessary to conduct the maintenance activity. Capturing and reviewing maintenance activity logs minimizes the risk that malicious activity occurs during a maintenance window.

For more information, refer to the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations" booklet.

VII.B Change Management

Change management encompasses the planning, governance (including approvals), testing, and implementation of any form of change. Examples of changes that are important to manage are configuration changes, software patches, and functional changes.

A key piece of change management is change control, further explained in the "Implementing Changes" section of this booklet. Changes may be necessary for a variety of reasons, including

[239] For more information, refer to NIST IR 8425, *Profile of the IoT Core Baseline for Consumer IoT Products*, and NIST IR 8259A, *IoT Device Cybersecurity Capability Core Baseline*.

updates or modifications to systems or components, requests from users, and results from vulnerability scans or penetration tests. The speed with which these changes should be implemented varies. For example, configuration changes or software patches that address a newly discovered vulnerability in a critical system should be completed quickly, while changes to improve user functionality in a low criticality system may have a slower implementation time frame.

The complexity and, in some cases, the system's or component's age may determine the type and frequency of changes needed to maintain the system or component. A system that is complex and contains multiple components may need to be changed more frequently than a simple system that is less reliant on multiple components. Similarly, a system or component that is new may experience more changes when it is initially placed into a production environment versus a system or component that has existed in a production environment for many years. Additionally, systems and components used by large groups may involve more frequent changes to meet the needs of a greater number of users.

Management should be aware of and manage the risks associated with unauthorized, untested, and unplanned changes. These risks include breach of confidentiality, breach of integrity (e.g., accidental configuration errors), lack of availability (e.g., system failure resulting from an inappropriate change or not implementing a change), or lack of resilience. Appropriate processes should be followed to manage changes (e.g., configuration management, vulnerability management, and patch management) regardless of whether the system or component was developed internally.

- **Configuration management:** Configuration management policies, standards, and procedures for developed and procured systems and components should be established, and validation performed to ensure that they are followed. The policies, standards, and procedures should include the identification and the use of system and component baseline configurations[240] (or original versions). The policies, standards, and procedures should specify that management evaluates, approves, documents, and disseminates changes to baseline configurations. Automated tools may simplify and promote consistency throughout the process. As previously stated, effective management helps ensure that changes to the configuration follow an established change management process. For more information, refer to the *FFIEC IT Handbook's* "Information Security" and "Architecture, Infrastructure, and Operations" booklets.
- **Vulnerability management:** Sources of vulnerability information (e.g., including threat intelligence) for the entity's systems and components should be identified. These sources may include the vendor, supplier, or developer, public-private information-sharing partnerships (e.g., the Financial Services Information Sharing and Analysis Center [FS-

[240] NIST defines "baseline configuration" as "the documented set of specifications for a system, or a configuration item within a system, that has been formally reviewed and agreed on at a given point in time, and which can be changed only through change control procedures."

ISAC][241] and NIST National Vulnerability Database[242]), and internal and external testing. Entity personnel should remediate relevant vulnerabilities and address threats continually and efficiently. The criticality of the identified vulnerability and the affected system or component can dictate the timing of the remediation. Appropriate senior leadership should review vulnerability management results, such as whether the most severe vulnerabilities are remediated within appropriate time frames, as specified in policy. Open or unresolved vulnerabilities in an entity's systems and components could result in loss of confidentiality, integrity, availability, and resilience.

- **Patch management:** Patch management processes are a critical part of an entity's effective change management practices. The entity's patch management processes should include procedures for identifying, evaluating, approving, testing, installing, and documenting patches or updates for systems and components. Regardless of whether the systems or components are developed internally or by a third party, it is critical to have a process for the timing, prioritization, testing, and deployment of patches to minimize the potential of vulnerability exploits that may result in system compromise or service disruption. Reasons for not applying patches should be documented and periodically reviewed by appropriate personnel, independent of the decision to not apply the patch. These processes help mitigate the risk that unaddressed vulnerabilities exist in systems and components and lead to system compromise. For more information, refer to the *FFIEC IT Handbook's* "Information Security," "Architecture, Infrastructure, and Operations," and "Audit" booklets.

VII.B.1 Implementing Changes

Action Summary

Management should have a process for the request, decision, approval or rejection, test, and implementation of changes. The goal of the change process should always be system or component confidentiality, integrity, availability, and resilience.

Examiners should review the following:

- Change management policies and procedures.
- Change logs.
- Relevant access controls.
- Processes for controlling system and software versions.
- Sample change documentation (e.g., change requests and approvals), including risk assessments.
- Change management program training.
- Conversion plans and coordination with conversion partners and stakeholders.

[241] FS-ISAC is an industry consortium dedicated to reducing cyber risk in the global financial system.

[242] NIST's National Vulnerability Database is the U.S. government's repository of standards-based vulnerability management data represented using the Security Content Automation Protocol. These data enable automation of vulnerability management, security measurement, and compliance. The database includes security checklist references, security-related software flaws, misconfigurations, product names, and impact metrics.

The process of implementing changes is sometimes referred to as change control. This process helps ensure that IT-related modifications are appropriately authorized, tested, documented, implemented, and monitored. Effective change control processes help management ensure that patches, updates, and configuration changes will not adversely affect a system, component, or application on the network. The characteristics and risks of a system, activity, or change should dictate the formality of the change control process. Management should maintain policies, standards, and procedures that specify the change control process, including defined roles and responsibilities. Examples of key roles with associated responsibilities are the project sponsor, change manager, quality manager, information security manager, auditor, network and system administrators, user support, and users.

A change manager is responsible for managing all changes affecting the entity to realize the intended improvements with minimal or no disruptions to operations. A group of stakeholders can aid the change manager in the assessment, prioritization, and scheduling of changes. Examples of this type of group include change advisory boards and change control review boards, which often consist of representatives from all areas in the entity's IT department, affected business lines, and third parties. This individual or group should develop communication protocols to ensure that all affected stakeholders are notified of changes.

Management should prioritize and categorize changes. Changes may be prioritized based on entity requirements (e.g., IT and information security); resources required; and legal, regulatory, and contractual reasons for the change. Change requests may be categorized as rejected, approved, or closed. Prioritizing focuses the organization on completion of the most important changes first. Categorizing changes helps the entity monitor change management performance. Therefore, appropriate change performance metrics and reports should be established.

Effective management has a method to track and report changes (e.g., standard change request forms, library and version controls, and spreadsheets or automated change logs). A change control tracking and reporting process, which may be manual or automated, gives management information about the number, priority, and status of change requests, allowing management to make informed decisions and adjustments as necessary. Maintaining and reviewing detailed change logs is critical for any change control processes. The absence of sound controls and adequate documentation of implemented changes can cause problems when designated personnel install subsequent system changes. Without adequate documentation and controls, personnel will not have the necessary information and may make a change that has an adverse effect on operations.

Effective management follows a defined change control process to make routine and planned changes to systems or components, adjust configuration settings, and make changes to remediate flaws. Management should also account for emergency and unscheduled changes in the change control process. When planning the change, effective management prepares for unexpected problems or failures with the change by creating a back-out plan and documenting the steps taken during the change in the order they occurred. This allows management to back out of a change partially if not all systems and components are affected. Generally, a change control process consists of defined steps, which may address the following:

- **Request:** The change request should include details of the change (e.g., system or component to be changed, description of the change, and justification); impact analysis to identify security needs, risks, and affected systems; change time frame; and back-out plan.
- **Review:** Stakeholders review change requests to determine their viability and business practicality.[243] If reviewers approve a change request, they also prioritize it based on the system's criticality, type of change, interdependencies and interconnections with the involved system, and duration of the change (i.e., time the involved system may be offline or degraded).
- **Approval:** An appropriate level of management, sometimes in collaboration with personnel or a committee, should determine approval of change decisions. Documentation of the decision and approval depends on the type of change initiated. Entity personnel should decide on changes in a timely manner (i.e., in advance of the change). Standard processes may not be followed in emergency situations. However, emergency changes should still be documented, even if after the fact. Emergency change processes should include a list of personnel approved to authorize emergency changes. All changes should be captured through the change control tracking and reporting process.
- **Design and build:** The developer (both internal and external) designs and builds the change to fulfill a request. When completed, the developer provides the requested change to the team or individual responsible for testing.
- **Test:** The team tests the proposed change independently for security, functionality, and any potential adverse effects of the change. Testing should verify that the change performs as intended, identifies any flaws (e.g., integrity issues), and that the change integrates with other systems as intended. Patches and updates that are deployed without testing may have unintended consequences (e.g., deployment delays and operational disruptions) and result in change rollback. Testing should be documented so that testing policy and procedure implementation may be audited.
- **Implementation:** Before deploying a change, an implementation plan (i.e., steps to deploy the change) should be created, full copy of the current production version backed up, and the back-out plan finalized. The team should deploy the approved change according to the implementation plan, preferably during off-peak hours or planned system downtime to minimize the system's or component's time offline.
- **Verify and close:** After implementation, management should perform a post-implementation review to verify that the change was deployed according to the implementation plan and functions appropriately. A comparison of modified programs with change authorization documents serves to validate that only approved changes were implemented. After verification, effective management follows processes to document completion of the change and close the change request. After closing, management should report on the status of the change and monitor the implemented change for unintended issues.

Security is critical when implementing changes for different systems or functions, such as network and client-server environments, mainframes, operating and application programs, and development and procurement projects. To help ensure that modifications do not adversely affect the security posture of varying systems, effective management integrates security into its change

[243] The review may be performed by IT management and stakeholders (e.g., system owner, technical staff, or financial personnel). The reviewers may form a formal committee (e.g., IT steering or operations) depending on the proposed change's scope, costs, risks, impact, and implementation time frame.

control processes. Failure to include information security considerations when implementing changes can result in operational disruptions, degradation in a system's performance, or inappropriate security controls in the system. A change impact analysis can help entity personnel determine the extent to which a change to a system or component may affect its operational, security, and compliance posture. Inaccurately assessing a change's effect could increase the likelihood of an operational disruption or a threat exploiting a vulnerability at the entity.

An impact analysis is completed before the entity makes a final decision regarding the change in the case of a planned change, or after implementation in the case of emergency changes. For a significant change, post-implementation reviews should identify any material impacts that were not anticipated, especially if there were any adverse effects on confidentiality, integrity, availability, resilience, or compliance.

VII.B.2 Additional Control Considerations in Change Management

VII.B.2(a) *Data Controls in the Testing Environment*

Use of production data in test environments should be accompanied by appropriate controls. Some entities have a testing environment that allows the testing of patches, updates, and fixes in a nonproduction environment that is similar or identical to the production environment, including the same configurations and data sets. Use of a test environment that matches the production environment raises the probability that testing will be effective in identifying flaws. If production data is copied into the test environment, it should be protected as if it were in the production environment. Another option is for an entity to generate synthetic data[244] for use in the test environment instead of using production data. Additionally, the testing environment should be isolated and otherwise protected to avoid it being a vulnerable exploit vector.

VII.B.2(b) *Library Controls*

Effective management strictly controls the movement of programs and files among development, quality management, and production libraries[245] for purposes of change control. Libraries allow programmers to access frequently used routines[246] and add them to programs without having to rewrite code. Libraries include executable code modules that run as part of larger applications. Inappropriate segregation of duties between development staff and non-development staff who manage the libraries can lead to production system defects, or the introduction of malicious programs, which in turn can negatively affect system confidentiality, integrity, availability, or resilience.

[244] According to NIST Glossary, "synthetic data generation" is "a process in which seed data are used to create artificial data that have some of the statistical characteristics of the seed data."

[245] In computing, the term "libraries" refers to collections of stored documentation, programs, and data typically segregated by the type of stored information, such as development, testing, and production-related programs, data, or documentation, and arranged for quick identification and reuse. Program libraries include reusable program routines or modules stored in source or object code formats.

[246] For purposes of this discussion, a "routine" is defined as "a sequence of computer instructions for performing a particular task."

Assigning librarian functions to independent personnel (e.g., quality management personnel in more complex entities or non-operations personnel in less complex entities) reduces the risk of errant or malicious code being moved to production environments. Additionally, independent program integrity verification, library activity logs, and documented approval processes also mitigate this risk. These controls should ensure that code in libraries is not altered or deleted. Additional library controls, to mitigate these risks include the following:

- **Object grouping and associated access controls:** Rather than establishing controls on individual objects in libraries, which can create security administration burden, entity personnel or automated programs group similar objects (e.g., executable and nonexecutable routines, test data, and production data) into libraries based on object type. Access can then be granted at library levels.
- **Role-based, automated library applications:** When feasible, the implementation of role-based, automated library applications can help management by restricting access at library or object levels. Additionally, such applications can produce reports that identify who accessed a library and what, if any, changes were made.

Effective management considers using these automated change controls commensurate with the complexity of the entity's IT environment and the number, types, and complexities of changes in that environment. Regardless of whether the entity has automated change control tools, management should strictly control access to production software libraries, particularly in distributed environments.

VII.B.2(c) *Code Repository Controls*

Code[247] can be stored in repositories.[248] They are used in the process of development and configuration management. Through version-control features, code repositories help management track and control different versions of developed source code and program documentation during development, as well as deployment scripts and configuration files. Code repositories may be hosted in-house, on third-party platforms, or on publicly available collaboration venues.

Management should establish code repository use policies. The policies should specify which code repositories the entity uses and where code repository data actually resides, especially for any cloud-based repositories (e.g., public, private, and community clouds). Effective policies and practices include the following actions:

- Prohibiting, when appropriate, the use of any public-based forums, especially if using a private account for entity business or data use.
- Establishing source code ownership.

[247] Code types include source code, executable code, and configuration-as-code.

[248] Repositories are locations where assets may be stored in no certain order. Repositories may contain assets such as libraries, data, and code.

- Requiring business account use to manage entity repository data and correspondingly prohibiting personal account use.
- Prohibiting entity credential use (e.g., email address, password) when creating personal repository accounts.
- Controlling and tracking entity accounts for personnel if cloud repositories are authorized.
- Prohibiting posting entity-owned code in open (i.e., public) forums.
- Evaluating security and other repository risks before using them.
    - Using available security measures, such as MFA.
    - Auditing to determine that all repository-suggested practices, including security measures, are appropriately adopted.
    - Ensuring sanitization of data, if applicable.
    - Restricting exfiltration of code to personal accounts.
- Implementing technical or logical controls (e.g., data loss prevention filters) to block sensitive data from being published to unapproved repositories.
- Searching open (i.e., public) code repositories for entity-owned source code for unauthorized posting of company-owned source code.

Code repositories can be used to manage software changes. For example, code repositories often have built-in version control functionality. Version control tools can be used to manage both software code and associated documentation. Lack of version control can result in updates to the wrong code version and inappropriate access to code. Additionally, inconsistency in code location and naming convention can result in development and maintenance inefficiency. Therefore, available version control features should be used to track changes made to the code and to increase individual accountability for changes.[249]

Another important security control for code repositories is access control. To prevent unauthorized access to or modification of code, access to code repositories should be role-based, follow the principle of least privilege, and limit access to authorized personnel, tools, and services. Management may consider authorization strategies (e.g., zero trust) to mitigate risk. The following are examples of effective access controls:

- Request and approval processes for access to code repositories (e.g., development, staging, or production).
- MFA requirement for access to all (internal and cloud-based) code repositories.
- Appropriate segregation of duties to prevent developer access to the staging and production code repositories, prevent quality management personnel from having access to production code repositories, and grant access to production code repositories only to release management personnel.
- Periodic review processes for access roles and repository logs.

[249] Refer to NIST SP 800-218, *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities*.

The entity's continuity planning should include code repository backup to facilitate recovery when the development, staging, or production environments are disrupted. Additional information related to the principles and concepts discussed in this section is available in the *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations," "Information Security," "Management," and "Audit" booklets.

VII.B.3 Change Types

There are several change or modification types that can be controlled through use of a defined change process. The change types are routine or planned modifications, major modifications, and emergency modifications. Effective management has procedures for changes that include change request, review, and approval that direct management to plan, test, and document changes before implementation. Regardless of the change type, management should ensure that changes to any IT system, component, or service are supported by an orderly, adaptable, documented, and measurable process to promote the consistent implementation of changes as well as provide an audit trail for changes. This gives management the necessary information for resilience purposes (e.g., specific rebuild point) or if there are problems when implementing the change (e.g., specific back-out procedures).

VII.B.3(a) *Routine Modifications*

Routine or planned modifications include changes to systems or components to improve performance, add functionality, enhance usability, address defects, improve security, or comply with a new law or regulation. Sometimes referred to as standard changes, routine modifications can be simple or complex but are not considered major and generally can be implemented during the normal course of business. Routine modifications follow repeatable procedures and are often implemented with automation. They include patches, version updates, and firewall changes. They also may include replacing or upgrading network hardware (e.g., printers, user computers, and networking devices). Defined implementation plans, which often include using automated deployment tools to deploy the modifications, are important especially when changes are implemented over numerous systems or components, or across widely dispersed networks.

Effective management plans and coordinates routine modifications because IT changes often affect multiple business lines. Change management discussions should involve sufficient representation from lines of business, IT, information security, quality management, and audit. It is important for involved personnel to receive training to ensure that changes support entity objectives and do not adversely affect confidentiality, integrity, availability, and resilience. Centralized oversight (e.g., through a committee) helps to manage the interdependencies of the entity's systems and operations. Committees providing oversight help clarify request requirements set by the entity and help ensure that all departments are aware of pending changes. Change management discussions to coordinate change activities may happen using specialized change control committees (e.g., CCB) or less formally through management discussions (e.g., during IT steering committee meetings).

After completing a routine modification, all systems and components should be backed up to an off-site location, secured, and maintained to ensure adequate confidentiality, integrity, availability, and resilience. Management should have an inventory of changes, including routine modifications, for back-out, resilience, and tracking purposes.

VII.B.3(b) *Major Modifications*

Major modifications include significant functional modifications to an existing system, conversion to a new system, or introduction of new systems, components, or data because of corporate mergers or acquisitions. In some cases, these may be referred to as “normal” changes as they are common but more complex than routine changes. These changes rely on extensive planning and additional management oversight as they often are longer-term; affect more of the entity’s systems, components, departments, and personnel; and incur higher cost. Major modifications should be implemented by adhering to a structured change management process, as noted in the “Implementing Changes” section of this booklet; however, some changes may need to go through the SDLC, particularly if there is any significant development activity. The goal of this process is to minimize risk, promote integrity and security, and maintain uninterrupted delivery of the service during and after a significant modification is implemented.

Entity personnel should assess risks related to major modification and all security-related changes. Key stakeholders should formally discuss these major modification requests and associated risks identified before approving the requests. Major modification policies and procedures should specify the decision criteria, accounting for requirements, feasibility, cost, project planning, software design, programming, testing, and implementation.

Entity personnel should determine the training required to make the conversion or major hardware and software upgrade successful. Training may start with ensuring that the major modification is adequately documented (e.g., new system information, user manuals, updates made to the system, and training information). Users affected by the changes should receive appropriate communication and training. Additionally, evaluation of the type, volume, and timing of training needs for each affected line of business and coordinate training programs with applicable third parties should be performed.

System conversions (and associated deconversions) are a major modification subset. A system conversion is a major change to an existing system and may involve migrating the system from one platform type to another (e.g., from on-premises to a cloud environment, or from one core provider to another). System conversions occur because of the introduction of new systems or corporate mergers and acquisitions. Changes to the foundation of the core platform may have additional effects throughout an entity’s operations, resulting in increased risk. Therefore, strong conversion controls (including documented conversion plans, adherence to entity change management processes, and comprehensive management oversight) are critical to prevent data corruption, performance degradation, and operational disruption. An improperly planned and executed conversion can disrupt the business and create inefficiencies, internal and external user dissatisfaction, and accounting problems, resulting in customer dissatisfaction, customer loss, and damaged entity, product, or service reputation.

Successful conversions rely on a comprehensive analysis of a conversion’s impact on existing operations (e.g., business processes and associated data processing, storage, and communication requirements), while accounting for interdependencies. Effective conversion policies, standards, and procedures should include provisions for internal and external communication channels, appropriate reporting, and lines of authority for timely decision-making and issue resolution. Elements such as the following should be considered when planning a conversion:

- **Conversion team:** The conversion team should include stakeholders with knowledge of the interdependencies of the system being converted and ability to articulate how the new system solves issues or improves outcomes.
- **Strategic fit:** The new system should meet as many of the entity's strategic and business objectives as possible (to limit the need for additional products or add-ons).
- **Integration:** The new system should integrate with existing systems and components. This often involves using internal APIs to securely connect to internal systems across lines of business and using external APIs to securely connect with the entity's third parties (e.g., cloud providers, fintechs,[250] third-party developers, partners, and customers).
- **Business intelligence:** The new system should enable comprehensive data analysis to support daily business activities and customer relationship management.
- **Ease of use:** The new system should be easily navigable and intuitive for entity personnel, which can improve workflow efficiency even if workflow efficiency was not an objective.
- **Evolution:** The new system should efficiently support evolving functional needs, changing technologies (e.g., evolution from on-premises to a cloud environment, artificial intelligence [AI], and quantum computing), and increasing capacity needs.
- **Compliance:** The new system should meet the entity's regulatory and internal policy requirements.
- **Security:** The new system should meet the entity's enterprise-wide information and cybersecurity requirements (e.g., security protocols, cyberattack monitoring, and resilience).
- **Training:** New system training should be available in a variety of formats (e.g., prerecorded, in-person, and virtual) and for a variety of users (e.g., end users and administrators).
- **Remote work flexibility:** An important variable to consider is whether the solution supports entity personnel remote work.
- **Conversion plans:** A key factor in conversion plans is ensuring the capability of personnel to map existing entity systems and components to the new system, to identify interdependencies, and to transfer data to the new systems. Additionally, an after-action review of the conversion can be useful for minimizing potential issues in future conversions.
- **Support:** Third parties should be able to support the conversion and provide ongoing support to the entity after the conversion.
- **Cloud:** Cloud-based solutions should meet applicable compliance requirements, include technology support, and integrate with the entity's existing systems and components.

Reference checking with a conversion partner's other clients can be helpful to learn the lessons from their conversions and improve the entity's conversion plan.

VII.B.3(c) *Emergency Modifications*

Emergency modifications correct severe and unexpected problems with systems and components

[250] As previously stated, for purposes of this booklet, the term "fintech" refers to using technology in novel ways to provide financial services.

and minimize operational disruptions after an incident. Examples of situations that may require emergency modifications include the following:[251]

- Natural disasters (e.g., floods, earthquakes, and storms) that disrupt critical facilities (e.g., data center outages and building evacuations for extended periods).
- Loss of a critical vendor or other supply chain component (e.g., cloud vendor, network vendor, telecom provider, electricity provider, and water system).
- IT incidents (e.g., denial of service attacks, malware, and network cable cuts).
- Emergency security patches released by vendors.
- Zero-day vulnerabilities that require timely configuration or other changes to mitigate risk until a patch can be developed, tested, and installed.

Entities that plan for emergency modifications are better able to limit negative effects on their business when situations like the above occur. To plan for emergency modifications, entities' policies, standards, and procedures should address expected emergency steps, such as the following:

- Specify who can declare an emergency.
- Designate a group (e.g., emergency change control review board) that can meet on extremely short notice to evaluate and approve proposed emergency modifications. The group could include senior leadership, critical line-of-business management, and key IT management.
- Create an emergency modification plan that includes a list of individuals in the emergency group (e.g., emergency change control review board) with contact information, and the procedures that are modified from standard change control and security processes to enable quick action.
- Plan for multiple methods of communication (e.g., in-person, phone, text, email, and collaboration site) in case the emergency occurs outside normal business hours, or a method is unavailable or insecure.
- Identify the responsible parties for enacting emergency changes and plan for their continuity.
- Compress certain phases of the implementation process to save time and still test, model, or simulate[252] before releasing the changes into the production environment. Any unwarranted delays in patching increases the risk that malicious actors or malware can exploit unpatched systems. For more information on exploits, refer to the *FFIEC IT Handbook's* "Information Security" booklet. Methods for reducing deployment time include the following:
    - Reducing the number of use cases addressed in testing and modeling, as well as planning to remediate issues with specific use cases.
    - Performing smaller scale testing in the modeling environment, such as deploying patches to 10 test machines instead of 50 or 500.

[251] Refer to CISA's *CRR Supplemental Resource Guide, Vol. 3: Configuration and Change Management, Version 1.1*, and *CRR Supplemental Resource Guide, Vol. 5: Incident Management, Version 1.1*.

[252] In computing, "modeling" or "simulation" is defined by *Merriam-Webster* as "the imitative representation of the functioning of one system or process by means of the functioning of another."

  - Maintaining a list of users who have the authority and access rights to easily acclimate to emergency changes and can communicate temporary remediation tactics to their peers.

- Establish emergency minimum recovery point objectives with stakeholders to reduce the time required to perform system backups.
- Establish emergency downtime procedures (e.g., failover to alternate systems) that allow systems to be taken completely offline in emergencies. This helps minimize the need for outages and reboots in times of heavy use and expedites change completion.
- Perform periodic emergency drills in the test and modeling environment to prepare IT staff to deal with the stress of emergency modifications. Review the results of these drills and adjust emergency modification plans accordingly.

As identified above, testing emergency modifications before deployment is preferable to minimize potential disruptions. Effective management weighs the potential damage from an untested emergency change against the known damage from the subject issue to determine the level of testing required. Because testing is likely to be shortened, it is even more critical in emergency situations to have backup files and programs available and to establish a back-out plan before emergency modifications are implemented. Appropriate backups, established back-out plans and procedures, and detailed documentation provide the ability to reverse a change if it disrupts the system.

The number of emergency modifications should be kept to a minimum, and management should validate every emergency modification after implementation to determine whether it was correctly classified. Control should be maintained over personnel attempting to circumvent routine modification or major modification control processes by incorrectly classifying changes as emergencies to avoid resource constraints (e.g., time, personnel, and cost).

Detailed documentation enhances management's ability to analyze the effect of the emergency modification during a post-implementation review and verify that the change was deployed appropriately. Evaluations and documentation reviews of emergency modifications should be completed as soon as possible after implementation. Findings from these evaluations and reviews are used to improve existing change control policies, standards, and procedures.

For more information, refer to the *FFIEC IT Handbook's* "Business Continuity Management," "Architecture, Infrastructure, and Operations," "Information Security," and "Management" booklets.

VII.B.4 Change Management Documentation

Documentation related to changes allows management to analyze changes over time, learn lessons from those changes, and the effect of those changes on the entity's systems and components. Documentation of all changes (e.g., configuration changes, patches applied, system conversions, and emergency modifications) and their impact on systems and components should be maintained. Some examples of change-related documentation are covered in the following subsections.

VII.B.4(a) *Change Request Form*

Entities use change request forms to communicate the specifics of the change to all stakeholders for analysis purposes after the change is completed. The forms should provide a clear description of the requested changes as well as sufficient information for affected parties to understand the effect of a change. Change requests typically include elements such as the following:[253]

- Change request title.
- Change control identification number.
- Requestor's name.
- Request date.
- Detailed description of the change, including system, component, or configuration item involved.
- Proposed date and time of the change.
- Stakeholder groups affected.
- Business justification for the change.
- Resources (e.g., personnel, expertise, cost, and time) needed to implement the change.
- Expected impact of the change.
- Expected impact on resilience requirements for the asset.
- Backup procedures.
- Rollback (or back-out) procedures.
- Change urgency (e.g., low, medium, high, or emergency; and scheduled, urgent, or unscheduled).
- If the change is an emergency, justification for emergency change.
- Change control review board input, such as the following:
    - Preliminary assessment of change.
    - Decision (e.g., approved, rejected, more information required, or deferred) and further action if not approved.
    - Comments.
    - Authorized approver's signature and date.

VII.B.4(b) *Impact Analysis*

Documented impact analyses assist in communicating the effect of a change on the entity's information security. The impact analysis is generally maintained with and may be presented

[253] Refer to CISA's *CRR Supplemental Resource Guide, Vol. 3: Configuration and Change Management, Version 1.1*, and *CRR Supplemental Resource Guide, Vol. 5: Incident Management, Version 1.1*. Also refer to NIST SP 800-128, *Guide for Security-Focused Configuration Management of Information Systems*.

with the change request to serve as supporting information for decision purposes. The impact analysis typically includes elements such as the following:[254]

- Change request title.
- Change control identification number.
- Requestor’s name.
- Request date.
- Analyst performing the impact analysis.
- Technical implications, such as other systems or components affected by the change or interface and integration issues (including with third parties).
- Functional impact, such as required changes to existing applications.
- Schedule impact.
- Financial impact, such as funding required to implement the change.
- People impact, such as end users, customers, expertise, and training.
- Security impact, including information security, cybersecurity, and physical security.
- Resilience impact.
- Entity risks from this change.
- Rollback (or back-out) implications.
- Alternatives to change and impact of not making the change.
- Authorized approval signature and date of approval.

VII.B.4(c) *Rollback or Back-Out Plan*

When implementing changes, there is always the potential that the deployment does not go as planned. When issues arise after a change, it may be necessary to rollback (or revert) to a prior version or level or back out a change and restore a system to its earlier state. Entity personnel should prepare for the possibility of a failed or incomplete deployment and have a plan with procedures to address the issues. Entity personnel should develop a rollback or back-out plan to reverse a failed deployment.

A rollback or back-out plan identifies the steps necessary for restoration to a previous secure and functional version of the baseline configuration.[255] This plan can help an entity recover from a system or component update that is not functioning correctly. The plan can explain what to do (e.g., shut down, failover, or replace) if a service or component fails and how to resume secure operations on a redundant system while still providing uninterrupted service to end users. Entity personnel should consider the time required to perform rollback procedures and when to trigger the rollback plan. Entity personnel should establish rollback plan timing that minimizes business disruption.

Change implementation failures or incomplete changes can cause expensive damage to systems and components. One effective plan step is to minimize damage by identifying a stable recovery

[254] Refer to CISA’s *CRR Supplemental Resource Guide, Vol. 3: Configuration and Change Management, Version 1.1*, and *CRR Supplemental Resource Guide, Vol. 5: Incident Management, Version 1.1.* Also refer to NIST SP 800-128, *Guide for Security-Focused Configuration Management of Information Systems*.
[255] Refer to NIST SP 800-128, *Guide for Security-Focused Configuration Management of Information Systems*.

point and archiving or imaging the system or component at that point. Entity personnel would develop and test related rollback procedures and identify spare systems and components (e.g., power supplies, system motherboards, hard drives, and communication switchboards) that may be needed, consistent with the criticality of the system or component. The entity documents the stable recovery point so multiple personnel can participate and view the same information during rollback (e.g., hardware, software, firmware, configuration files, and other configuration records).

It is important to control rollback tools because an unauthorized rollback could allow an attacker to restore a system or component to a prior vulnerable image, which in turn could allow the attacker to corrupt the system or component or inject malware. Effective management has appropriate security controls to prevent unauthorized rollback. The entity may have a special update process (e.g., physical presence of personnel) for rolling back the installed systems and components to an earlier version. This mechanism provides a dual control for management to guard against attackers using automation to install old systems and components with known vulnerabilities.

Effective management tracks and reports issues that lead to the use of a back-out plan and the effects of a failed or incomplete change deployment. Documentation of failed or incomplete deployments should include lessons learned for future change implementations.

VII.C End-of-Life

With respect to technology, EOL is a time frame usually defined by a technology vendor to describe when an asset has reached the end of its useful life cycle and when the vendor will no longer maintain and support the asset or continue to sell or license it. This is also applicable to a service provided in support of purchased IT assets. When the vendor no longer updates or plans to sunset services that support the IT asset or discontinues the sale of licensing of support services, this form of EOL is considered “end-of-service.”[256] If the IT asset was created in-house, EOL occurs when the entity has decided to no longer update and otherwise maintain it.

A vendor generally determines the useful life of the IT assets it sells. The vendor identifies when it will stop providing updates and communicates a timeline along which it will sunset the systems and components. Entity personnel who purchase IT assets from vendors should evaluate the expected life of the IT asset as part of their due diligence. EOL considerations should occur as part of the initial contract negotiation and at contract renewal and be specified in the resulting contract. For example, the contract should contain a clause specifying the amount of advanced notice the vendor will provide the entity before it discontinues updates and other support of the IT asset. This notification is critical to the entity in planning replacements (including migrations from the legacy system to the replacement), and in planning for the off-boarding of the IT asset. Therefore, management should have a process to determine EOL decisions (e.g., replace, upgrade, or migrate to a new system) before a system’s or component’s EOL.

[256] System, component, and service providers sometimes use the term “end-of-service” to indicate when maintenance services and updates will no longer be supported or provided.

If developed internally, the entity should plan for system or component obsolescence. Management, development staff, and users may have input into the system's or component's useful life timeline. If a system or component no longer meets users' needs, or maintenance costs are too high, management may conclude that the system or component has reached EOL and should be replaced.

The entity's IT asset inventory should have information about the useful life of IT assets in order for the entity to plan for orderly transitions from one system or component to another. An effective ITAM process includes monitoring each IT asset's depreciation and obsolescence schedule and planning well in advance for EOL. Items to consider in the process include the party responsible for development, version, patch levels (available and currently deployed), and the system's or component's expected life. An entity may leverage existing tools (e.g., depreciation schedule and IT asset inventory) to facilitate obsolescence planning and ensure that replacement resources have been provided for.

An IT asset that is obsolete and at EOL or end-of-service may be more vulnerable to exploitation; therefore, this risk should be managed to maintain confidentiality, integrity, availability, and resilience. With older and less resilient IT assets, an increased number of security vulnerabilities may result in greater risk of intrusion and exploitation (e.g., malware, ransomware, and data breaches). The potential impacts from these risks increases for interconnected systems and components because of the number and types of access points. These risks also exist for components provided by all parties in the supply chain. Continuously assessing the risks of using aging, outdated, and unsupported systems and components is the foundation for mitigating the risks. Effective risk mitigations include the following:

- Replacing systems and component in a timely manner.
- Establishing processes for vendors to notify the entity of component EOL status changes (e.g., end-of-production, end-of-service).
- Extending support from the vendor.[257]
- Adding or augmenting system or component controls:
    - Segregating the system or component from the network until it can be replaced or removed.
    - Locking down system or component devices and ports for access.
    - Increasing access controls and monitoring.
    - Minimizing connections to the system or component.

Management should consider EOL risks tied to maintaining legacy systems and components for extended periods of time. When continuing to use legacy systems, management should consider weaknesses related to the design, development, manufacturing, production, shipping and receiving, delivery, and operation of the system and component that can be exploited at EOL by

[257] Third-party providers often charge additional fees to continue providing updates after a system's or component's EOL. Extending support beyond EOL may result in risks, such as operating systems and components with compounded vulnerabilities and diminishing availability of expertise.

a threat source.[258] These legacy-related risks are two-fold. If management continues to use legacy systems or components, management puts its supply chain partners at risk; and if the entity's supply chain partners continue to operate legacy or unsupported systems or components, the entity may be at risk for EOL or end-of-support vulnerabilities. Effective management addresses EOL and the use of legacy systems and components and time frames for necessary replacement, upgrade, or migration in contracts between the entity and its supply chain partners.

For more information on ITAM and EOL, refer to *FFIEC IT Handbook's* "Architecture, Infrastructure, and Operations," "Information Security," "Outsourcing Technology Services," "Business Continuity Management," and "Management" booklets.

VII.D Termination and Disposal

When a system or component becomes obsolete, is no longer supported, no longer meets the users' or entity's needs, or becomes too costly to maintain, management may decide to terminate its use and dispose of the system or component. Effective management has appropriate termination and off-boarding procedures. The procedures should identify who has the responsibility to decide whether to terminate a vendor relationship, as well as an approval and acknowledgment process in which contracted parties agree to the relationship termination. Effective management negotiates contract clauses that specify termination criteria, including related to fourth or nth party relationships (e.g., affiliates, subcontractors) if necessary. It is important that the termination clauses address confidentiality, integrity, availability, and resilience of data and operations throughout the off-boarding process.

There are several important considerations when terminating a system, component, or service including disposing of decommissioned systems and components, and safely migrating and disposing of data and documentation. Data are particularly important when being transferred to a new replacement system to maintain confidentiality, integrity, availability, and resilience. Effective management has procedures for activities such as the following:

- Shutdown (i.e., sunset) of the systems and components.
- Identification of the tasks involved.
- Data identification and preservation (i.e., archival or transfer) or disposal.
- Secure disposal of unnecessary systems and components (e.g., hardware and software).

Effective management validates that appropriate personnel are aware of the procedures to coordinate the orderly disposition of the system, its components, and the data.[259] The entity's critical or sensitive data and documents should be removed (e.g., wiped, sanitized, or otherwise destroyed) logically and physically from the system and components before disposal. This extends to any data or documents maintained by the third party on behalf of the entity, and effective management addresses the entity's requirements in the contract termination and disposal clauses.

[258] Refer to "CISA Insights – Cyber: Remediate Vulnerabilities for Internet-Accessible Systems."

[259] Refer to U.S. Department of Justice's *Systems Development Life Cycle Guidance Document,* chapter 12, "Disposition Phase."

VII.E Maintenance Documentation

Maintenance documentation provides valuable descriptions for the upkeep of an entity’s systems and components and significantly enhances management’s ability to maintain, operate, and administer IT assets. The larger and more dispersed an entity, the greater role documentation plays in effective maintenance across the entity. Advantages of documentation for users include access to operation manuals, training materials, and online application help features. Documentation enhances IT personnel’s ability to maintain and update systems efficiently, track updates to the systems and components, and identify and correct programming defects.

Developing and maintaining current and accurate maintenance documentation can be complicated, time-consuming, and expensive. However, maintaining standardized documentation and indexing procedures are important to help ensure the uniform accessibility of existing maintenance documentation. Authorized personnel should update the documentation with system, component, and configuration updates according to the entity’s or third party’s prescribed standards, as applicable. Effective management works with third parties to help ensure that the entity receives current system, component, and user maintenance documentation.

APPENDIX A: EXAMINATION PROCEDURES

Examination Objectives

Examiners should use these procedures (also referred to as the work program) intended to help them determine the quality and effectiveness of the entity's management of IT, particularly regarding development, acquisition, and maintenance-related risks as outlined in this booklet. Examiners should use these procedures to measure the adequacy of the entity's ITRM process, including management awareness and participation, risk assessment, policies, standards, and procedures, reporting, ongoing monitoring, and follow-up.

Examiners may choose to use only particular examination procedure work steps based on the size, complexity, and nature of the entity's business.

Objective 1: Determine the appropriate scope and objectives for the examination.

1. Review past reports for outstanding issues or previous problems. Consider the following:

   a. Regulatory reports of examination.
   b. Internal and external audit reports (e.g., SSAE18).
   c. Internal or independent tests or reviews of controls (e.g., penetration tests, vulnerability assessments, SOC reports, business continuity reviews, and third-party management reviews).
   d. Regulatory and audit reports on service providers.

2. Review management's response to issues raised during, or since, the last examination. Consider the following:

   a. Adequacy and timing of corrective action.
   b. Resolution of root causes rather than just specific issues.
   c. Existence of any outstanding issues.
   d. Whether management has taken positive action toward correcting exceptions reported in audit and examination reports.
   e. Independent review of resolution and reporting of resolution to the audit committee.

3. Interview management and review responses to pre-examination information requests to identify changes to the technology infrastructure or new products and services that might increase the entity's risk. Consider the following:

   a. Any significant strategic or business line changes, including any third-party changes.
   b. Products or services delivered to either internal or external users.
   c. Current network diagrams and data flow diagrams, including changes to configuration or components.
   d. Hardware and software inventories.
   e. Loss or addition of key personnel.
   f. Inventories of third-party providers and software vendors, including services provided.

g. Organizational charts that include reporting relationships between business units and control functions (e.g., enterprise risk management, ITRM, and internal audit).
h. Credit or operating losses primarily attributable (or thought to be attributable) to IT (e.g., system problems, inadequate controls, improperly implemented changes to systems, and fraud resulting from cybersecurity attacks, such as account takeover).
i. Changes to internal business processes.
j. Internal reorganizations.

Objective 2: Management establishes, and the board of directors (board) oversees, an effective governance structure that includes development, acquisition, and maintenance activities. Additionally, the board oversees related IT project management processes used to manage projects related to those activities. (Section II, "Governance of Development, Acquisition, and Maintenance")

Review the following documents to understand the overall structure and hierarchy of management and oversight activities related to development, acquisition, and maintenance:

- Enterprise-wide IT policies, procedures, and standards describing the entity's requirements throughout the development, acquisition, and maintenance life cycle.
- Charters (e.g., board, management, or committee), organizational charts, relevant documentation, and practices to determine whether appropriate roles and responsibilities are identified and assigned with suitable decision-making authority.
- Project plans and meeting minutes of the board and committees to ensure that activities and projects align with the entity's strategic objectives and the board's risk appetite.
- Project audit reports to determine whether the audit function provides independent, objective assurance of the effectiveness of an entity's development, acquisition, and maintenance activities.
- Documentation of controls for personnel with access to program code to limit placement into the production environment.
- QA reports to evaluate processes for the detection of potential coding errors.

1. Determine whether the board and senior management provides for the following:

   a. An effective governance structure that allows for the effective oversight and management of the development, acquisition, and maintenance of the entity's systems and components.
   b. Oversight of development, acquisition, and maintenance IT project management processes and projects related to those activities.
   c. Oversight of significant projects to ensure that they align with an entity's strategic plans.
   d. Consideration of business strategies and objectives, resilience needs, information security requirements, legal and regulatory requirements, and allocation of resources (e.g., personnel, budget, and time) when evaluating IT projects and activities.
   e. Consideration of the needs of internal and external stakeholders when making decisions regarding IT development, acquisition, and maintenance activities.

f. Establishment of audit's role in reviewing development, acquisition, and maintenance activities, and ability to raise objections if they believe the control environment is inadequate.

2. Determine whether the board and senior management develops and implements comprehensive, entity-wide IT policies, procedures, and standards addressing entity requirements throughout the development, acquisition, and maintenance life cycle. Does the board, senior management, and designated committees of the board perform the following:

   a. Regularly review and approve IT policies and standards.
   b. Review and approve procedures to meet changing IT policies and standards.
   c. Provide for policies that clearly delineate development, acquisition, and maintenance responsibilities and provide for communication to all personnel, stakeholders, and appropriate third parties.
   d. Review and approve deviations from policies, standards, and procedures related to development, acquisition, and maintenance.
   e. Identify and assign appropriate roles and responsibilities for the entity's development, acquisition, and maintenance activities.
   f. Identify and address training needs to support development, acquisition, and maintenance responsibilities.
   g. Provide for an escalation process for development, acquisition, and maintenance issues.

3. Determine whether the board and senior management identified and provided for key management responsibilities related to development, acquisition, and maintenance activities. Examples of key roles which carry those responsibilities include the following:

   a. Board and senior management.
   b. CIO.
   c. CISO.
   d. IT steering committee.
   e. Business line management.

4. Determine whether management identified and provided for the responsibilities of IT project management related to development, acquisition, and maintenance activities. Examples of key project management roles that carry those responsibilities include the following:

   a. Sponsor.
   b. Project owners.
   c. Product owner.
   d. Project or program manager.
   e. Business change manager.
   f. Program management office or organization personnel.
   g. Stakeholders.

5. Determine whether management identified and provided for the responsibilities of key roles in development activities. Examples of key development roles that carry those responsibilities include the following:

   a. Network architects.
   b. Developers.
   c. Software engineers.
   d. Hardware engineers.
   e. Information security analysts.
   f. Systems analysts.

6. Determine whether management identified and provided for the responsibilities of key roles in acquisition activities, including examples such as procurement manager or third-party risk manager.

7. Determine whether maintenance personnel provide for appropriate maintenance throughout the life cycle of systems and components. Determine whether maintenance personnel

   a. Have knowledge and understanding of all relevant systems and components they are expected to operate and maintain.
   b. Analyze and understand the costs of maintenance (e.g., budget, time, or personnel) versus the costs of not performing maintenance (e.g., system failures, data breaches, and customer dissatisfaction).
   c. Track the analysis of costs for reporting to management.
   d. Maintain independence from development roles to prevent developers from accessing production environments.

8. Evaluate audit's role in reviewing development, acquisition, and maintenance activities. Consider the following audit activities:

   a. Validating that sufficient time is built into project schedules to define controls and verify that all appropriate stakeholders are involved.
   b. Determining the effectiveness of controls necessary in the entity's development, acquisition, and maintenance activities and recommending appropriate mitigation.
   c. Guiding developers in considering appropriate control standards and frameworks throughout IT projects.
   d. Reviewing the internal controls, testing, and audit trails included in systems and components during each SDLC phase.
   e. Performing post-implementation reviews shortly after implementation of new or revised systems or components.

Objective 3: Evaluate whether management implements continuous risk management processes within the entity's development, acquisition, and maintenance activities to identify reasonably foreseeable internal and external risks and threats, including those that could result in unauthorized disclosure, misuse, alteration, or destruction of customer information

or customer information systems. (Section III, "Risk Management of Development, Acquisition, and Maintenance")

1. Examiners should review management's continuous risk management processes for development, acquisition, and maintenance. Consider the following:

   a. Policies, standards, and procedures for identifying, measuring, mitigating, monitoring, and reporting risks related to development, acquisition, and maintenance activities.
   b. Documented processes and metrics used to measure the level of risk.
   c. Detailed documentation of processes used to review, accept, and document risks that management cannot mitigate or transfer.
   d. Documentation of risk assessment processes to identify key risks at the onset of IT projects and throughout their life cycles.
   e. Risk assessments that highlight internal and external risks related to development, acquisition, and maintenance activities.
   f. Documentation of ongoing processes and reports to monitor and communicate risk, including emerging risks related to development, acquisition, and maintenance activities.
   g. Reports to stakeholders that are timely, accurate, and include clear, relevant metrics.

2. Evaluate whether management has identified reasonably foreseeable internal and external risks and threats, including those that could result in unauthorized disclosure, misuse, alteration, or destruction of customer information or customer information systems. Consider management's risk management actions in addressing the following:

   a. Risks that could result in a severe disruption or material compromise to critical service delivery.
   b. Policies, standards, and procedures for identifying, measuring, mitigating, monitoring, and reporting risks related to development, acquisition, and maintenance activities.
   c. Repeatable process adoption, to ensure that risks are consistently addressed across the entity over time.
   d. Risk identification and assessments involving stakeholders with business process knowledge who may be affected by the development, acquisition, or maintenance activities or the related IT projects.
   e. Threat model usage commensurate with the overall threats to the entity, to assess appropriateness of security policies, standards, and procedures.
   f. Process to review, accept, and document risk when management cannot mitigate or transfer risk, and its acceptance by the board.
   g. Regular review and approval of risk acceptance decisions consistent with the entity's governance structure.
   h. Information security concerns.

3. Determine whether management has established effective risk measurement, monitoring, and reporting processes, including for emerging risks. Consider the following:

    a. Implementation of effective standards to measure risk in the entity's development, acquisition, and maintenance activities.
    b. Establishment of an ongoing risk measurement process, commensurate with the size and complexity of the entity's activities.
    c. Communication of technical aspects through reports to the board that are clear and understandable to board members (e.g., explaining acronyms or technical jargon) and that relate IT issues to business concerns.
    d. Receipt of reports on IT project risks, such as critical projects that may miss key milestone dates, identification of negative test results, or changes to business or technical requirements.

Objective 4: Evaluate whether management has implemented effective risk mitigation throughout development, acquisition, and maintenance activities regardless of the phase of the project in the life cycle and agnostic as to the type of technology. Specific risks and controls should be considered depending on management's chosen solution. (Section IV, "Common Development, Acquisition, and Maintenance Risk Topics")

For this objective, examiners should review and assess the following:

- Systems, components, and services, including related contracts and licenses, throughout the supply chain for appropriate risk identification and mitigation.
- Secure coding standards, configurations, and security controls used to harden each system or component, data, and activity logs for appropriate processes and implementation and maintenance of confidentiality, integrity, and availability.
- Inventory of all systems, components (e.g., open-source, proprietary, APIs, and container images and registries), including related licenses, and data as part of ITAM.
- Processes for access controls, authorization, and authentication for systems and components, data, and related documentation throughout the supply chain to ensure appropriate security.
- Roles and responsibilities to evaluate segregation of duties in and ongoing management commitment to development, acquisition, and maintenance activities.
- Process of selecting and implementing methodologies to enable effective management and control of development, acquisition, or maintenance projects and alignment with entity objectives.
- Operating parameters (e.g., timing, speed, throughput, and data validation) for systems and components to determine performance.
- Activity logs related to systems, components, and data to identify operating risks.
- Monitoring processes of development and maintenance activity for identification of anomalies and unauthorized access or modification to systems, components, and data.
- Reporting processes for decision-making and measuring the level of project success.
- Training on development, acquisition, and maintenance concepts (e.g., methodologies); effectiveness of training; and capability of personnel to implement concepts learned.
- Documentation of internally developed programs and externally procured products and services to effectively operate and maintain the systems and components.

- Evaluation process (e.g., post-implementation review, stakeholder interview, problem documentation and resolution, cost-benefit analysis, and reports to senior management) for development, acquisition, and maintenance projects.

1. Evaluate management's oversight of use of open-source and COTS components. Consider the following:

   a. Identification and mitigation of risks related to the use of open-source components.
   b. Evaluation of documentation providing support if there are open-source components supporting any line of business.
   c. Proactive maintenance of updated application and user documentation, especially when code changes are made.
   d. Consultation with third-party providers and vendors regarding recommended security controls to harden the COTS solution.
   e. Mitigation of risks related to integration (e.g., lack of interoperability or integration between systems and components).

2. Evaluate management's ability to negotiate, administer, and implement practices regarding licenses, agreements, and copyright protection. Consider whether management

   a. Accurately assesses current and future needs and ensures that licenses continue to meet the entity's needs.
   b. Reviews licenses, obtaining confirmation that they clearly state whether system or component usage is exclusive, the number of user licenses, and whether there are any time, place, manner, or other types of limitations with the system or component's use.
   c. Specifies in its agreements the appropriate time periods for all licenses and the minimum amount of notice required for license termination.
   d. Periodically reviews system and component licenses to compare all installed licensed systems and components with the respective license terms.
   e. Determines whether a third-party vendor provides access to source or object code ("code") when drafting agreements.
   f. Reviews licenses to determine whether they include retention and use of backup copies of any mission-critical system or component on which the entity may rely for disaster recovery or business continuity purposes at remote sites.
   g. Negotiates, when possible, with licensors to ensure the inclusion of retention and use of backup copies if they are omitted from the terms of a license.
   h. Understands any resulting limitations and consequent risks if involved in difficult license agreement negotiations, including when an entity has limited negotiating power.
   i. Periodically reviews system and component licenses to compare all installed licensed systems and components with the respective license terms.
   j. Considers what is stated in the licenses; costs of obtaining them; and risks, fines, and other compliance liabilities for violating their provisions or requirements.
   k. Includes related entities (e.g., subsidiaries or contractors) as users in their licenses when the entity plans to provide the systems or components to them.

l. Implements appropriate licenses for the systems or components developed when management is responsible for developing systems and components and providing them to other entities or its subsidiaries.
m. Maintains awareness of the liabilities that come with licensing activities, including considerations related to hardware maintenance, integration, compatibility, and fraud, when the entity builds hardware and licenses that hardware to other entities.
n. Specifies in its agreements the appropriate time periods for all licenses and the minimum amount of notice required for license termination.
o. Addresses inappropriate usage if there is a discrepancy between actual usage and the total number of installed licenses allowed by the agreement.
p. Addresses maintenance agreements, which outline available maintenance services (e.g., provision of new versions, releases, or updates).
q. Obtains appropriate vendor permission and participation for modifications to the code for systems and components.
r. Ensures that the agreement addresses and prompts updates to application and user documentation when any changes are made to procured systems or components.

3. Assess whether management's ITAM program includes oversight and management of the software and hardware licenses for components used by the entity. Consider management

   a. Awareness of the liabilities that come with licensing activities, including considerations related to software maintenance, integration, compatibility, and fraud if an entity develops software and licenses that software to others.
   b. Maintenance of an accurate inventory of FOSS as part of an effective ITAM process and understanding of and abiding by the requirements of FOSS licenses.
   c. Awareness of what information is shared when contributing in return to the FOSS development community.
   d. Maintenance of program provisions for identification and management of any hardware licenses.

4. Evaluate management's oversight of copyright protections. Consider management's actions

   a. When deploying procured systems or components for use in the entity's infrastructure, including physical and virtual networks.
   b. When reviewing and approving the procurement and use of systems or components in the entity's IT environment to ensure appropriate use.

5. Assess management's actions to promote secure development practices for products and services developed and used. Consider management's actions to

   a. Monitor third parties' activities to enforce conformance with the entity's contract requirements, Information Security Standards, and legal and regulatory requirements when management outsources development to third parties.
   b. Evaluate or have a method to evaluate the third party's secure coding standards (e.g., through independent certification or audit) when acquiring systems and components.

c. Determine that secure development standards apply to the development of any system or component changes (e.g., patches or updates) for the maintenance of systems and components.
d. Incorporate security in the code, and implement quality management, which is critical to the secure development process**.**
e. Determine that the embedded rules of the code review tools are appropriately configured and used if an automated code review process is implemented.
f. Maintain a secure operating environment throughout development, acquisition, and maintenance activities, promoting secure development practices whether operations are performed in-house by entity personnel or outsourced to a third party.

6. Assess management’s actions to securely manage data. Consider management’s actions to

   a. Identify the data, their characteristics (e.g., customer sensitive information or proprietary information), and sensitivity (e.g., confidential, internal use only, or public).
   b. Document the data in an inventory that includes locations and uses of data in development, acquisition, and maintenance activities and in IT projects.
   c. Include data security in development, acquisition, and maintenance activities.
   d. Provide appropriate data confidentiality, integrity, availability, and resilience procedures for data input and output, including the use of data in IT projects.

7. Assess management of microservices activities related to development, acquisition, and maintenance. Consider the following management actions:

   a. Setting parameters for the processes of registration to and deregistration from the service registry.
   b. Implementation options that meet the entity’s security requirements.
   c. Implementation and management of microservices-based architectures, including monitoring of the many services, which may be running on different servers or written in different languages.
   d. Consideration of the following monitoring controls:
      - Monitoring at the gateway and service level.
      - Implementing a centralized dashboard to display the status of multiple services and network segments.
      - Creating a baseline and implementing intrusion detection to provide alerts on deviations from the baseline.

8. Assess management’s consideration of risk in the use of containers, such as image risk, registry risk, orchestrator risk, container risk, and host OS risk, and effective countermeasures to risk. Consider management actions to address data isolation when setting up containers.

9. Evaluate management’s risk mitigation strategies for API activities. Consider the following:

   a. Configuration of the API with the necessary infrastructure services.
   b. Application of extra layers of security beyond those for standard end-point security, and implementation of appropriate authorization checks at the object level.
   c. Implementation of appropriate security, patches, updates for APIs, and disabling of unnecessary functions.
   d. Appropriate setting of all API processing parameters (e.g., execution timeouts, maximum allocatable memory, number of file descriptors, number of processes, request size, number of requests per client or resource, and number of records per page to return in a single request response).
   e. Verification that all access is denied by default and employ specific role-based permissions for users to gain access to functions.
   f. Denial of access to properties unnecessary for the user’s role, and access only allowed when necessary.
   g. Validation and filtering of all data coming to an API with appropriate parameters.
   h. Maintenance of logging of API activity (e.g., authentication, errors, redirects, rate limiting, and end points, including their parameters, requests, and responses).
   i. Implementation of continuous logging and monitoring of relevant API activity (e.g., failed authentication attempts, denied access, and input validation errors).
   j. Maintenance of secure logs and ensuring only appropriate access to them to maintain confidentiality and integrity of sensitive data in activity logs.

10. Assess management’s implementation and use of any methodologies for project management. Consider management actions

    a. Establishing appropriate methodologies to enable effective management and control of system and component development, acquisition, and maintenance activities.
    b. Aligning implementation of any methodology with the overall strategic and business objectives.
    c. Promoting effective planning through IT project management and support training for developers, quality management members, testers, and maintenance personnel.
    d. Considering the risk of not following established control procedures when using prototyping models.
    e. Considering security development process weaknesses when using agile methodologies and determining appropriate mitigation strategies.

11. Evaluate quality management actions. Consider

    a. Implementation of compensating controls when they cannot fully achieve segregation of duties between quality management and development roles.
    b. Provision of adequate support for projects throughout an entity to promote success of a project.

12. Assess documentation standards. Consider management's process for

    a. Maintenance of documentation for internally developed programs and externally procured products and services.
    b. Maintenance of necessary information and documentation to understand and convey how a system or component was developed, how it functions, and where appropriate security and resilience points are included.
    c. Maintenance of detailed documentation for each system and component in development and production.
    d. Implementation of established policies to prepare documentation before deployment and additionally when appropriate, such as when management makes changes.
    e. Obtaining documentation from the third party and incorporating it into the entity's own stored documentation for acquired systems and components.
    f. Validating before purchase (through an internal review or a third-party certification) that a procured system's documentation meets the entity's documentation needs and standards.

13. Assess management's post-implementation review processes. Consider effective evaluation of development, acquisition, and maintenance projects and management actions, such as

    a. Conducting post-implementation reviews at the end of a project to validate completion of project objectives and assess development activities.
    b. Interviewing key stakeholders actively involved in the operational use of a product to determine effectiveness of the completed project.
    c. Documenting and addressing any identified problems to improve future projects and remediate issues in current projects.
    d. Analyzing effectiveness of project management activities by comparing, among other things, planned and actual costs, benefits, risks, return on investment information, and development time frames.
    e. Documenting results and presenting them to senior management and determining who should be informed of any operational or project management deficiencies.

Objective 5: Evaluate whether management has developed and followed consistent processes to identify risks and oversee IT projects to address any risks identified. IT projects should be managed according to the size and complexity of the entity and its project characteristics and risks. (Section IV.N "IT Project Management")

For this objective, examiners should review and assess

- Project plans, proposals, and status reports to determine whether the IT project manager works with stakeholders (e.g., PMO) to oversee and carry out IT projects.
- Board and committee minutes to evaluate whether management reviews and prioritizes projects and whether it considers the project's effect on operations and the entity's needs.

- Project plans, project requests, and documentation standards to determine whether they are structured appropriately to clearly define the project purpose, entity's requirements, and include deliverables for each project phase to address business needs.
- Documentation of management approvals of any addition or modification of functional, security, or control features and demonstration that changes are aligned with the project charter and project plan.
- Testing and quality management plans used to validate that the project meets the entity's project goals and stakeholder requirements.
- Documentation for the closeout of the entity's IT projects.

1. Assess whether management develops and follows consistent processes to identify risks and oversee IT projects to address any risks identified. Consider whether management

    a. Establishes project management policies, standards, and procedures that apply enterprise-wide.
    b. Develops and follows consistent processes to identify risks and oversee IT projects to address any risks identified.
    c. Considers the project's effect on operations and the entity's needs (e.g., IT, information security, lines of business, customer needs, and regulatory and legal compliance) when reviewing and prioritizing projects.
    d. Performs analysis of system or component needs for a given project.
    e. Monitors changes for any addition or modification of functional, security, or control features to help ensure that these changes are approved and aligned with the project charter and project plan.
    f. Develops and executes testing plans, which validate whether the entity's project goals and stakeholder requirements are met.
    g. Maintains comprehensive and accurate documentation reflecting the testing methodology employed, actual tests performed, and test results throughout the testing process.
    h. Implements quality management (also may be referred to as QA and QC) processes to help ensure that project requirements are met and validated.
    i. Defines and maintains the required elements for documentation of the closeout of the entity's IT projects.
    j. Establishes standards and maintains appropriate documentation for IT projects to foster a consistent and accurate process across projects.
    k. Maintains processes for developing IT project plans that describe existing system benefits and weaknesses and explain project objectives.

Objective 6: Evaluate whether management has an SDLC to manage systems and system components throughout their life cycle; to achieve the objectives of confidentiality, integrity, availability, and resilience; and to meet the entity's business objectives. (Section IV.O "System Development Life Cycle")

For this objective, examiners should review and assess

- The documented management and control processes for development.

- Description and detail of entity SDLC-related controls, including for supply chain partners.
- Responsibility and accountability assignment.
- Key stakeholder involvement.
- Clear evidence of stakeholder communication and tracking of all SDLC phases and actions.

1. Assess management's oversight of the SDLC process. Consider management's implementation of a documented process, such as an SDLC, used to manage and control development activities, if the entity engages in internal development activities.

2. Evaluate management's maintenance of protections (e.g., information security and resilience) in the information systems, components, and networks for data in transit and at rest, including for the entity's and supply chain partners' information, through all phases of the SDLC.

Objective 7: Evaluate whether management implements effective processes to address the plans and actions needed in each phase of an IT project, as part of the SDLC. (Section IV.O.1 "SDLC Phases")

For this objective, examiners should review and assess the activities associated with the entity's SDLC phases, such as the following example phases:

Initiation Phase

- Description of the project's purpose, expected benefits, support for entity business objectives, and any legal and regulatory requirements.
- Plans for addressing additional costs, resource needs, and alternate solutions.
- Validation of initial impact analysis, including any baseline security concerns.
- Documentation of the project proposal for all stakeholders in the supply chain.

Development or Acquisition Phase

- Documentation of design specifications.
- Documentation of mitigation strategies for risks identified in the impact analysis.
- Risk assessments, baseline controls design, and effectiveness of security controls.
- Documentation of initial development testing, conversion, implementation, and training plans, including confirmation of the functionality and controls.
- Documentation of additional design requirements and development changes.
- Documentation of draft user, operator, and maintenance manuals.

Implementation and Assessment Phase

- Documentation of design reviews and system tests, including any new specifications.
- Documentation of deployment approach selected.
- Training provided, including user and system support documentation.
- Post-implementation review, including any configuration changes.

Operations and Maintenance Phase

- Documentation of performance and controls monitoring.
- Documentation of configuration and change management controls and proposed changes.
- ITAM inventory and associated risk assessments.

Sunset and Disposal Phase

- Plans and validation measures for accessible data retrieval from archives.
- Documentation of off-boarding procedures, including for third-party providers.
- Documentation of a post-disposal review, including processes and lessons learned.

1. Assess management's oversight of SDLC phases. Consider whether management

    a. Understands the SDLC phases and the actions in each phase.
    b. Identifies the actions and assign responsibility and accountability for completing those actions.
    c. Defines the phases that will be included in the entity's SDLC and helps ensure the completion of each phase to promote transparency and accountability for, and agreement and acceptance by, the stakeholders.

2. Assess management's actions during the initiation phase of the SDLC. Consider management's actions to

    a. Describe the development project's purpose, identify expected benefits, and explain how the proposed system or component supports the entity's objectives.
    b. Address legal and regulatory requirements.
    c. Identify alternative solutions and explain the entity's confidentiality, integrity, availability, and resilience requirements.
    d. Consider and address any potential baseline security concerns during this phase by performing an initial impact analysis.
    e. Address planning items, including the following:
        - Responsibilities of third-party service providers, internal audit, information security, and IT staff.
        - Established acceptance criteria for each SDLC phase.
        - Established review and approval procedures to help ensure that development teams complete all SDLC phase or independent sprint requirements before moving into subsequent phases.
        - Identify control and security features to be designed, built or acquired, and implemented.
        - Create processes for development and change management to minimize disruptions to the development process.
        - Create processes that address project methodology selection, approval authority, and risk management procedures.
        - Identify costs associated with project overhead (e.g., office space, hardware, and software used during the project) as well as soft costs in budgeting personnel expenses and outsourced activities.

f. Consider input from all stakeholders.
g. Evaluate the appropriateness of each requested functional requirement.
h. Consider all proposals and analyze them to determine whether to develop or acquire the system or component.

3. Assess management's actions during the development or acquisition phase of the SDLC. Consider management's actions to

a. Determine other functionality, such as whether the configuration of the system or component (e.g., an IoT product) is possible, whether the system or component can be restored to a secure default setting, and whether there are restrictions on users or services that can make changes.
b. Consider the results of the impact analysis performed in the SDLC initiation phase and review those results throughout the SDLC.
c. Address or mitigate concerns from the impact analysis or new concerns that present risk beyond the board's risk appetite.
d. Conduct a risk assessment and use the results to design baseline controls to meet the entity's requirements.
e. Build in appropriate controls, including for network interfaces, to any products the entity develops or acquires to maintain confidentiality, integrity, availability, and resilience throughout the supply chain.
f. Review the appropriateness of new or modified design requirements or development changes and monitor for and minimize scope creep.

4. Assess management's actions during the implementation and assessment phase of the SDLC. Consider management's actions to

a. Perform design reviews and system tests before implementation of the system or component to ensure that all entity specifications are met.
b. Perform additional acceptance tests if new features or controls are added to the system or component.
c. Use and document the results of the design reviews and system tests, update documentation after performing new reviews or tests, and maintain test results for future review and use.
d. Determine the appropriate implementation strategy for the entity and system or component being deployed.
e. Coordinate training logistics, including who should be trained, what the training involves, and when training should be conducted.
f. Organize a training and awareness campaign, and notify users of any implementation and training responsibilities.
g. Perform a post-implementation review once the product is implemented.
h. Validate the initial impact analysis during the post-implementation review and report on any changes to the results.
i. Confirm that the implementation occurred as planned, and determine whether there were any unanticipated effects of the change on existing controls.

j. Validate whether configurations (e.g., security, functionality, or performance) on the new system are appropriate.

5. Assess management’s actions during the operations and maintenance phase of the SDLC. Consider management’s actions to

   a. Monitor performance of a system or component to help ensure that it is consistent with pre-established user, security, and other entity requirements, and making necessary modifications.
   b. Conduct configuration management and control activities and document any proposed or actual changes in the entity’s security or operational plan of the system or component.
   c. Assess and document changes in the entity’s ITAM inventory and related risk assessments.
   d. Follow established change management policies, standards, and procedures to minimize the potential of a modification disrupting or degrading operations.
   e. Perform operations and maintenance phase tasks regardless of whether entity personnel or a third party performs them.

6. Assess management’s actions during the sunset and disposal phase of the SDLC. Consider management’s actions to

   a. Consider the need and plan for methods for future data retrieval before archiving information and if necessary.
   b. Maintain archived data in an accessible and readable format, adhering to data retention guidelines.
   c. Periodically validate the accessibility of the archived data or develop a plan to migrate potentially inaccessible data to an accessible format.
   d. Maintain confidentiality, integrity, availability, and resilience throughout this phase.
   e. Develop comprehensive off-boarding procedures if the system or component requiring disposal is managed by, or the related data are stored by, a third party.
   f. Perform a post-disposal review at the end of the sunset and disposal phase that details the processes and lessons learned from shutting down and archiving the terminated system or component.

Objective 8: Evaluate management’s third-party relationship risk management processes related to development, acquisition, and maintenance. *(Section IV.P “Third-Party Relationship Risk Management”)*

1. Assess management’s due diligence practices as part of its third-party selection and relationship risk management processes. Consider whether management

   a. Identifies and documents any limitations of its due diligence, understands the risks from such limitations, and considers alternatives regarding how to mitigate development, acquisition, and maintenance risks.
   b. Evaluates the conclusions from supplemental due diligence efforts, such as information derived from third parties (e.g., industry utilities or consortiums,

consultations with other organizations, or engaging in joint efforts), based on the entity's own specific circumstances and performance criteria for the activity.

c. Involves the board, through its oversight responsibilities, to ensure awareness of—and, as appropriate, request approval or delegated approval of—contracts involving higher-risk development, acquisition, and maintenance activities.

Objective 9: Evaluate whether management implements policies to measure, monitor, and track changes and use all related information obtained to inform SCRM activities. (Section IV.Q "Supply Chain Considerations")

Examiners should review

- Policies, procedures, project plans, and SDLC procedures to determine whether SCRM activities are integrated into the SDLC for the entity and applicable third parties.
- Control configurations and reports used by management to monitor, control, and protect communications at the key access points of supply chain information systems.
- Architectural designs, software development techniques, and systems engineering principles that promote effective information security throughout the supply chain.
- Topologies and process flow diagrams that identify interconnectivities and potential single points of failure, while also considering continuity and resilience.
- Inventory and validation of documentation (e.g., due diligence, including contracts) for supply chain partners, as well as documentation for provenance of systems, components, and data.
- Communication processes and documented communication of threat intelligence and vulnerability identification with supply chain partners.
- Management's assessment of inauthentic or unapproved systems or components (e.g., counterfeit or shadow IT).
- SCRM controls in maintenance-related situations, including monitoring for unauthorized modifications, communicating changes, and monitoring for EOL.

1. Assess implementation of policies for tracking changes and use of the obtained information to inform the entity's SCRM activities. Consider management's actions to

   a. Ensure that SCRM activities are integrated into the SDLC for the entity and applicable third parties.
   b. Monitor, control, and protect communications (i.e., information transmitted or received) at key access points (e.g., external boundaries and key internal boundaries) of supply chain information systems.
   c. Use architectural designs, software development techniques, and systems engineering principles that promote effective information security in the supply chain.
   d. Identify potential single points of failure among all entities in the entity's supply chain numerous interconnectivities and risks from all partners.
   e. Identify and consider any nested third-party relationships through common ownership (e.g., affiliates, subsidiaries).

f. Identify and consider risks related to supply chains by tracking interdependencies between all parties involved (e.g., owners and third-party service providers, including vendors) in the supply chain.
g. Consider any information pertinent to the security, integrity, resilience, quality, trustworthiness (e.g., not on the OFAC list, other concerns), or authenticity of their supply chain partners or products.
h. Evaluate supply chain partners consistently, based on available information, and depending on the specific context and purpose for which the assessment is being conducted, select additional factors for consideration.
i. Document the reference sources for assessment information to help establish the quality of information (e.g., its relevance, completeness, and accuracy) relied on for supply chain assessments.

2. Assess supply chain risk management practices. Consider management's processes and plans to analyze policies, standards, and procedures, which outline and describe an entity and third parties' processes for SCRM (including for subcontractors). Consider activities such as the following:

a. Maintaining a current and accurate inventory of all applicable suppliers and identify their criticality to the business.
b. Considering established national and international standards, as applicable, as a baseline for security requirements for the entity's supply chain.
c. Assessing and addressing supply chain risks associated with a given geographic location and apply appropriate risk responses (e.g., defining acceptable locations).
d. Defining roles and responsibilities for personnel (e.g., development, acquisition, and maintenance) to address various supply chain activities.
e. Providing general controls and processes.
f. Developing, documenting, and maintaining an accurate inventory of third parties that reflects the entity's key supply chain partners.
g. Considering configuration management minimum security requirements for the supply chain.
h. Applying appropriate configuration management controls to its own systems and encouraging or requiring the use of comparable controls by all parties in the entity's supply chain through contracts.
i. Understanding and mitigating all relevant risks associated with interdependencies throughout the various supply chains potentially affecting the entity.
j. Planning for resilience, including scenarios related to supply chain issues.
k. Documenting provenance for systems, components, and data, and monitoring for changes in the chain of custody that may increase risk to the entity throughout the SDLC.
l. Considering provisions for excess capacity, bandwidth, and redundancy in agreements with supply chain partners, and take appropriate mitigation action, as necessary.
m. Considering creation of SBOM for applicable and appropriate classes of software (e.g., purchased, open-source, and in-house developed software).

n. Validating that existing SCRM capabilities (e.g., vulnerability management practices and vendor risk assessments) are not deprioritized under the mistaken assumption that SBOM replaces these activities.
o. Ensuring the quality of information (e.g., its relevance, completeness, and accuracy) relied on for an assessment and documenting the reference sources for assessment information.
p. Establishing effective project management processes that can help identify critical components, especially those that are used by multiple business lines, functions, systems, and components, such as the following:
   - Determine whether there is potential foreign ownership or influence, and whether the supply chain partner may have relationships with OFAC-sanctioned individuals, organizations, or countries.
   - Evaluate supply chain partner oversight of its subcontractors (i.e., fourth parties) or developers.
   - Identify the level of open-source systems and components used by the entity, and determine how the supply chain partner demonstrates its compliance with applicable open-source licensing agreements.
   - Conduct research on COTS systems and components (e.g., via publicly available resources) or request proof to determine whether the supply chain partner (e.g., OEM) has performed testing as part of their quality or security processes.
   - Use authorized resellers or distributors with an ongoing relationship with the supply chain partner for systems and components not directly acquired from an OEM entity.
   - Acquire directly from vetted OEMs or their authorized distributors and resellers when obtaining alternative sources for continued support.
   - Track chain of custody of systems, components, and underlying code as they move throughout the supply chain to minimize the potential for counterfeit or altered products.
q. Communicating with its supply chain partners to promote awareness of threat intelligence and relevant vulnerabilities in the entity’s supply chain to inform operational security processes.
r. Monitoring for supply chain system and component security and threat intelligence alerts and advisories from supply chain partners and taking appropriate actions in response.
s. Considering employment of techniques to introduce randomness into entity operations and assets in the entity’s systems or networks.
t. Considering concealment techniques, such as masking metadata that may be accessible in downloads or deliveries of systems and components, whether developed or acquired.
u. Consulting with the entity’s legal counsel and board, as appropriate, regarding advanced security protection techniques (e.g., misdirection, honeypots) beyond standard industry techniques (e.g., basic randomness and concealment) techniques before implementation, as they may present liability and additional risk to the entity.
v. Considering incident response-related information (e.g., definition of an incident, roles, and responsibilities) in its agreements with its supply chain partners.

w. Considering correlation of available threat information with potential threats and vulnerabilities.
x. Considering implementation of advanced monitoring over activities performed by higher-risk personnel (e.g., users with elevated authority or privileges).
y. Considering information-sharing clauses in agreements and contracts.
z. Performing the following when implementing maintenance processes throughout the supply chain:
    - Defining agreements that identify roles, responsibilities, and practices that may be used for maintenance activities, especially when third parties are responsible for maintaining the entity's systems or components.
    - Monitoring for unauthorized modification or removal of the entity's systems or components (e.g., use of counterfeit systems and components, malware) in the supply chain.
    - Monitoring systems and components for planning for EOL to prepare for replacement or upgrade to the systems and components.
    - Considering any potential availability issues in the supply chain for systems and components, or providing agreements to continue support for legacy systems or components until a change can be made without significant disruption to operations.

aa. Understanding and considering interconnectivities throughout the supply chain to effectively manage supply chain risks.
bb. Incorporating third parties into business continuity and resilience activities throughout the supply chain, when applicable.
cc. Considering use of available information, such as that provided by third-party user groups and associations, to augment ongoing monitoring and due diligence, threat intelligence, and security throughout the supply chain.
dd. Implementing a process to validate the effectiveness of the security of systems and components throughout the supply chain when performing development, acquisition, and maintenance activities, and make appropriate adjustments to risk assessments to account for interconnectivity risk.
ee. Considering risk management factors in the entire life cycle of a third-party relationship, including planning, due diligence, contract negotiation, ongoing monitoring, and termination.

Objective 10: Evaluate whether management implements sound processes for the development of systems and components supporting the entity's business needs and operations. (Section V "Development")

Examiners should review the following:

- Documentation of training in secure design and coding techniques for those responsible for development activities.
- Development standards, including procedures for managing changes and a back-out plan.
- Evidence of communication throughout the supply chain regarding identification of critical systems and components, potential threats and vulnerabilities, configurations, and responsibilities for selection and maintenance of system and network components.

- Entity coding standards and list of entity-approved software development languages.
- Documentation of security certification for completed systems and component-related code.
- Development designs used to validate security and functionality throughout the supply chain.
- Documentation of risk assessment related to use of or switch to a DevOps approach and effectiveness of controls.

1. Assess management’s ability to provide policies, procedures, and practices to oversee and manage development activities. Consider management’s actions to

   a. Provide access to training in secure design and coding techniques to those responsible for development activities to securely design and address key issues early in the development process for entities that develop or modify their own systems and software.
   b. Maintain and follow policies, standards, and procedures documented for developing systems and components incorporating the entity’s requirements for confidentiality, integrity, availability, and resilience.
   c. Appropriately manage risks (e.g., eliminating inaccurate techniques or outdated tools and prototypes) with use of any additional design and coding techniques beyond standard coding techniques (e.g., computer-aided design) and prototypes.
   d. Develop, as part of the SDLC, a trustworthy system that meets specific security and other critical requirements defined and set by entity management.
   e. Employ secure program coding practices to help develop a trustworthy system.
   f. Incorporate supply chain processes when choosing system and network components.
   g. Plan for maintenance activities from the outset of the development process to address secure continuity of operations.

2. Assess management’s development of standards and controls. Consider the following:

   a. Procedures for managing changes during the development process, that address:
      - System controls.
      - Quality management.
      - Release management.
      - Documentation.
      - Reporting.
   b. Documentation of reasons for using specific programming styles and languages on a project or service in project documentation.
   c. Documentation of completed systems and component-related code that have passed security certification in program libraries as discussed in the “Additional Control Considerations in Change Management” section of this booklet.
   d. Design of information systems, components, and elements to be difficult to disable (e.g., tamper-proofing techniques), and, if they are disabled, trigger notification methods such as audit trails, tamper evidence, or alarms.
   e. Design of delivery mechanisms (e.g., downloads for software) to avoid unnecessary exposure or access to the supply chain and systems or components traversing the supply chain.

f. Design of relevant validation mechanisms to be used during implementation and operation.
g. Agreement with partners on standards and controls used in customized system and component development throughout the supply chain.
h. Work with suppliers and partners to ensure that critical systems and components are identified.
i. Validate that suppliers or the entity itself has a continued ability to maintain customized systems and components that are critical to the entity's operations.

Objective 11: Evaluate whether management maintains formal testing processes and standards to govern the effectiveness of testing internally and externally developed systems and components. (Section V.B "Testing")

Examiners should review the following:

- Testing policies and procedures to confirm that systems and components meet the entity's requirements.
- Testing scope documentation, including for application interoperability and discovery of vulnerabilities.
- Documentation regarding appropriate controls and approvals over the use of production data in testing.
- Documentation of the type of testing, testing results, corrective actions, and testing completion.
- Documentation of testing for any new controls added to systems and components to avoid conflicts.
- Updates to manuals and training plans after testing.

1. Assess whether management appropriately tests systems and components to identify and mitigate risks or vulnerabilities before deployment. Consider the following management actions:

    a. Determining the need for use of production data in testing and employing appropriate controls (e.g., masking) if its use is deemed necessary.
    b. Documenting approval of the use of nonsanitized data by the board when sanitizing data is not feasible and implementing and maintaining controls similar to those used in the production environment to appropriately protect the data for compliance with legal and regulatory requirements.
    c. Performing additional acceptance tests of new controls if they are added to the system to help ensure that new controls meet security specifications and do not conflict with or invalidate existing controls or functionality.
    d. Establishing a methodical process to define and conduct testing necessary to demonstrate the effectiveness of a developed system or component.
    e. Implementing practices to document, report, and address identified issues, including security-related issues, in a timely manner, regardless of the testing methods used.
    f. Reviewing and finalizing all supporting documentation, such as user, operator, and maintenance manuals as well as any conversion, implementation, and training plans

associated with a new release or significant update during the implementation and assessment phase.

2. Assess management's oversight and control of DevOps and DevSecOps activities. Consider management's activities to

   a. Assess risks involved in using or switching to a DevOps approach and implementing appropriate controls.
   b. Consider controls to secure the CI/CD pipeline process, such as the following:
      - Hardening servers hosting code and artifact repositories.
      - Securing credentials (e.g., authorization tokens) used for accessing repositories.
      - Implementing controls on who can check in and check out artifacts in container image registries.
      - Logging all code and build update activities.
      - Sending build reports to developers and stopping further pipeline tasks when a build or test fails in the CI pipeline and configuring code repositories to automatically block all subsequent pull requests from CD or continuous deployment pipeline until issues are resolved.
      - Sending build reports to the security team and stop further pipeline tasks when a code or build audit fails.
      - Ensuring that developers can only access the application code.
      - Digitally signing (preferably multiparty digital signing) the release artifact during each required CI/CD stage during the build and release process.
      - Verifying that all required digital signatures are present during production release to ensure that no one bypasses the pipeline.

3. Assess management's oversight of database development. Consider management's actions to

   a. Understand the different types, structures, and uses of databases to effectively mitigate the risks of database development.
   b. Understand the benefits, limitations, and appropriate controls for the type of database used.
   c. Choose the appropriate database to meet business and stakeholder objectives.
   d. Consider access needed and services (e.g., management, analysis, and data services) that the databases will provide for internal and external customers.

Objective 12: Evaluate whether management establishes acquisition processes that are commensurate with the entity's business and procurement needs and assesses and mitigates procurement risks associated with overall entity strategic, regulatory, and operational risks. (Section VI "Acquisition")

Examiners should review the following:

- Project requests and plans related to acquisition.

- Risk assessments, strategic and business plans, and due diligence activities, including meeting minutes related to the acquisition process.
- Contracts and licensing agreements for appropriate provisions, responsibilities and accountability, and mitigation strategies.
- Acquisition-related requests containing documented evaluation criteria and sufficient information for decision-making.
- Documentation standards and procedures, including for updating documentation.
- Stakeholder training and guidance in the acquisition process.
- Audits related to the acquisition process and procurement.

1. Assess management's mitigation of the risks associated with acquisition of systems or components. Consider its actions to coordinate implementation of the following:

   a. Ascertaining whether the product specifications are "fit for purpose" and meet the entity's requirements, (regardless of whether the entity purchases directly from OEM partners or a secondary market) and are addressed in agreements with supply chain partners.
   b. Developing policies, standards, and procedures to effectively carry out the entity's procurement processes aligned with the entity's acquisition activities.
   c. Performing rigorous due diligence reviews of potential suppliers.
   d. Aligning the depth of the due diligence evaluation with the complexity, scope, and risk assigned by the risk assessment for the system or component to be procured.
   e. Implementing contract and licensing processes.
   f. Reviewing contracts and licensing agreements to help ensure that the rights and responsibilities of each party are clear and identify accountability.

2. Assess management's procurement process. Consider management actions to

   a. Develop and manage an effective IT procurement process to meet the entity's acquisition and contract fulfillment needs.
   b. Follow a structured procurement process for any IT procurement, but particularly when procuring significant systems, components, products, and services to help ensure that each step in the process is addressed.
   c. Ascertain that, at a minimum, procured hardware, software, and services adhere to entity standards.

3. Assess whether management has established policies, standards, and procedures to better assist in the effective and consistent management of the entity's acquisition process. Consider management's actions to

   a. Provide for acquisition policies, standards, and procedures that promote the
      - Definition of a process for issuing information requests (e.g., RFIs and RFPs) to third parties.
      - Performance of appropriate third-party due diligence processes.

    - Negotiation of the contract to ensure that it contains terms amenable to entity management and types of IT-related provisions to consider before contract execution.
    - Establishment of processes for the entity's contract signature authority and process for specialist review (e.g., subject matter experts, key stakeholder, legal, and compliance) of IT-related contracts before approval and signature.
    - Validation of the systems, components, products, and services to ensure that they meet user requirements before acceptance.
    - Transferability or portability of systems, components, products, and services if a third-party relationship is discontinued.

b. Understand the following risks and their effects on the business before entering foreign-based third-party relationships:
    - Legal.
    - Country.
    - Currency (or exchange rate).
    - Geopolitical.
    - Resilience.

c. Identify, plan, and address the root causes of foreign-based risks or crises to minimize their effects on the entity's operations.
d. Plan for maintenance activities from the outset of the acquisition process to address secure continuity of operations.

4. Assess management's project selection process. Consider management's actions to

    a. Consider business needs, third-party solutions, specifications, and costs.
    b. Involve appropriate stakeholders in defining the entity's IT, information security, and functional requirements, and that the project meets applicable legal and regulatory expectations.
    c. Identify the entity's needs to avoid a third party's confusion with a request for information and omission of critical information needed for decision-making.
    d. Refrain from making assumptions based on third-party responses when management receives information from them, and ask follow-up questions to clarify concerns or solicit more information.
    e. Determine when a relationship is established and a contract is created when using an RFQ.
    f. Consult with the entity's legal counsel to identify and resolve contractual issues.
    g. Implement a formal process for evaluating information, proposals, and quotes received from the RFIs, RFPs, and RFQs that includes key information to help management select appropriate providers.

5. Assess management's oversight of contracts and agreements. Consider management's actions to

    a. Implement standards and procedures for documentation, review, and approval.
    b. Consult with legal representation to determine the entity's rights and enforceability of terms, because agreements may not always be enforceable in a court of law.

c. Identify each of the agreements and contracts that help management complete the evaluation and selection steps.
d. Validate that contracts have clauses that address relevant security and privacy standards with third-party service providers or vendors that process, store, or transmit sensitive customer data or provide critical services to meet applicable legal and regulatory requirements.
e. Consider providing training and guidance to support a structured approach to SOW development.
f. Involve appropriate personnel (e.g., relevant stakeholders) in developing an SOW and establish an appropriate senior management review and approval process.
g. Validate that the provider's needs and requirements (e.g., price, security, and interoperability) are met in order to provide for delivery of a quality product or service.
h. Establish clear and measurable expectations for the services provided, recourse when expectations are not met, and accountability for both parties if management is procuring services related to a system or component.
i. Link SLAs to clauses in the contract regarding incentives, penalties, and contract termination to protect the entity in the event of third-party performance failures.
j. Outline rights in contracts, including licenses, if applicable, when using systems and components developed by third parties.
k. Consider risks and address them through contract clauses (e.g., representations, warranties, and indemnifications to vendor liability limitations, information security, and agreement modifications).
l. Ensure that contracts have clauses that address relevant security and privacy standards with third-party service providers or vendors that process, store, or transmit sensitive customer data or provide critical services to meet applicable legal and regulatory requirements.
m. Consider contract clauses that allow for a contract to be revisited if a key subcontractor changes, which may affect critical services in the entity.
n. Determine during contract negotiation whether management is willing to accept or can mitigate the risks related to systems or components without requested terms in situations when third parties cannot or will not agree to the terms an entity sets out in the draft contract.
o. Document, with appropriate approvals consistent with the entity's policies and governance structure, any accepted risk or compensating control factors if third parties cannot or will not agree to the terms an entity sets out in the draft contract.
p. Validate that the third-party software developer's development policies and standards meet or exceed those of the entity.
q. Validate that third-party developers are following industry standards, at a minimum.
r. Negotiate terms, if possible, that would enable another third-party service provider to access an affected system or component and help management in the conversion without violating agreement restrictions.
s. Consider alternative solutions if management cannot accept or mitigate the risk.

6. Assess whether management has provided appropriate oversight of escrow arrangements with third parties. Consider management's actions to

a. Validate, at least annually, that the third party maintains a current version of the source code for software in escrow.
b. Consider incorporating provisions into escrow agreements such as
    - Definitions of minimum programming and system and component documentation.
    - Definitions of system and component maintenance procedures.
    - Conditions that should be present before an entity can access the source code and related documentation.
    - Assurances that the escrow agent will hold current versions of the source code and related documentation to validate that escrowed information is updated whenever significant program changes are made.
    - Arrangements for auditing or testing the integrity of the escrowed code.
    - Descriptions of the source code and related documentation and the storage type or location (e.g., magnetic tape or cloud) containing it.
    - Assurances that the storage type or location containing the source code and related documentation is accessible, operable, and compatible with an entity's existing IT environment.
    - Assurances that the source code can be compiled into executable code.
    - If the escrow agent is based outside the United States, consideration of the practical and legal implications of establishing foreign-based escrow arrangements.

7. Assess management's exit strategy plan for transitioning or terminating a product, service, or third-party relationship. Consider management's actions to

a. Help minimize disruption to an entity's operations.
b. Develop an initial exit strategy during the procurement process to address a situation when a third party cannot or does not meet the contract terms.
c. Consider identifying alternative third parties that provide similar products and services at the onset of the relationship.

Objective 13: Evaluate whether management establishes formal policies, procedures, and responsibilities for managing systems and component maintenance, and processes that ensure the availability and continued operability of systems and components. (Section VII, "Maintenance")

Examiners should review the following:

- System and component inventory for key information (e.g., version, update, and patch level) to identify and address vulnerabilities.
- Maintenance plans, including maintenance schedule and logs, vendor and developer recommendations, cost-benefit analyses, operational risks, availability of qualified personnel, and other relevant industry factors.
- Change authorizations and review any reports for identification of anomalous activity.
- Change types and risk assessment processes for those changes.

- Configuration management practices to validate appropriate implementation and enforcement of established change processes.
- Vulnerability and threat identification processes and remediation plans.
- Reports to board and senior management regarding maintenance issues.
- IT asset inventory including considerations for EOL, vulnerability management, planned obsolescence, and legacy systems and components.
- Termination, disposal, and off-boarding processes of systems, components, and third-party relationships.
- Maintenance documentation for any system, component, and configuration updates.

1. Assess whether management provides for development of policies for the maintenance of systems and components, including remote access for performing maintenance, roles and responsibilities of personnel with access for maintenance activities, and monitoring and audit mechanisms of maintenance activities. Consider management's actions to

    a. Analyze and understand the costs of maintenance (e.g., budget, time, or personnel) versus the costs of not performing maintenance (e.g., system failures, data breaches, and customer dissatisfaction), and track costs for reporting purposes.
    b. Plan for maintenance activities from the outset of the acquisition and development processes to address secure continuity of operations.
    c. Perform preventive maintenance throughout an IT asset's useful life to prevent or minimize catastrophic failure and promote confidentiality, integrity, availability, and resilience.
    d. Consider vendor or developer recommendations, cost-benefit analyses, operational risks, availability of qualified personnel, and other relevant industry factors (e.g., emerging technologies and threats) when developing the schedule for the maintenance plan.
    e. Track system and component information, such as version, update, and patch level, as tracking and using this information helps personnel manage (e.g., through regular updates, data protection, and digital forensics) technology vulnerabilities.
    f. Develop a process for identifying maintenance and vulnerability information and mechanisms for responding to user questions about developed or acquired IoT systems and components.
    g. Validate that while a system or component is in use, it is appropriately maintained and updated through preventative maintenance and change management.
    h. Establish a preventive maintenance plan, especially for mission-critical systems and components.
    i. Maintain logs of maintenance activities and perform reviews of those logs as needed for authorization of changes, as well as anomalous or malicious activity.
    j. Maintain awareness of the risks associated with unauthorized, untested, or unplanned changes.
    k. Consider the capability of the product to receive, verify, and apply verified updates for configuration, patch, and vulnerability management for developed or acquired products.
    l. Implement processes to manage changes regardless of whether the system or component was developed internally.

m. Establish configuration management processes for developed and procured systems and components and help ensure that those processes are followed and enforced.
n. Validate that only authorized personnel implement configuration changes and make those changes only to designated systems.
o. Identify the sources of vulnerability information and threat intelligence for the entity's systems and components.
p. Receive the results of internal or external vulnerability scanning and penetration testing and identify remediation plans to resolve any issues detected through these tests.
q. Evaluate and mitigate the risk associated with the vulnerability to the entity's affected systems and components when obtaining vulnerability and threat intelligence from vendors, developers, third parties, and other sources.
r. Report the results of the testing and the remediation plans to the board.
s. Validate that the entity's patch management processes include procedures for identifying, evaluating, approving, testing, installing, and documenting patches or updates for systems and components.

2. Assess management's oversight of end-of-life activities. Consider management's actions to

a. Maintain an accurate inventory of the entity's IT assets, including systems and components, regardless of whether they are developed internally or acquired from a third party.
b. Account for planned obsolescence if systems or components are developed internally.
c. Include a contract clause regarding advanced notification before the vendor's sunset timeline of an IT asset if a vendor decides not to replace, upgrade, or continue to support the asset.
d. Implement a process to determine EOL decisions (e.g., replace, upgrade, or migrate to a new system) before a system or component's EOL.
e. Document and monitor depreciation and obsolescence schedules of the entity's systems and components in its ITAM inventory to plan for EOL, as a part of its ITAM process.
f. Consider EOL risks tied to interconnections in the supply chain derived from maintaining legacy systems and components for an extended period of time.
g. Address EOL and the use of legacy systems and components, including time frames for necessary replacement or upgrade in contracts between the entity and its supply chain partners.
h. Address EOL risks related to the use of legacy systems and components by mitigation strategies such as the following:
    - Assess the risks of continued use of aging, outdated, and unsupported systems and components.
    - Consider transitioning to newer, supported systems and components.
    - Employ compensating controls.

3. Assess management's practices for termination and disposal of systems and components. Consider management's actions to

   a. Implement appropriate termination and off-boarding procedures.
   b. Include appropriate clauses in contracts with third parties to manage the termination of services, including related relationships (e.g., affiliates, subcontractors), if necessary. Clauses and contracts should address
      - Maintaining confidentiality, integrity, availability, and resilience of data and operations throughout the off-boarding process.
      - Specifying criteria defining termination.
   c. Develop and implement procedures for the following:
      - Shutdown (i.e., sunset) of the systems and components.
      - Identification of the tasks involved.
      - Preservation and identification of the data and its disposition (i.e., archival or transfer).
      - Secure disposal of unnecessary systems and components (e.g., hardware and software).
   d. Validate that appropriate personnel are aware of the procedures to coordinate the orderly disposition of the system, its components, and the data.
   e. Develop and implement procedures to help ensure that the entity's critical or sensitive data and documents are removed (e.g., wiped, sanitized, or otherwise destroyed) logically and physically from the system and components before disposal.
   f. Validate that data or documents maintained by the third party on behalf of the entity are addressed by the entity's requirements in the termination and disposal clauses of the contract.

4. Assess management's oversight of maintenance documentation, including working with third parties to help ensure that the entity receives current system, component, and user maintenance documentation.

Objective 14: Evaluate whether management establishes a formal process for the review, justification, approval, implementation, testing, and disposition of changes, and maintains confidentiality, integrity, availability, and resilience in the change management process. (Section VII.B, "Change Management")

Examiners should review the following:

- Change requests demonstrating prioritization, categorization, tracking, and reporting.
- Change control process, including for back-out plans and inventory, for all change types.
- Change management controls, including change logs, access control, and version control features.
- Risk assessment and change documentation, including approvals.
- Training for change management.
- Conversion plans and coordination with conversion partners and stakeholders.
- Reviews of change control processes for comprehensiveness and ongoing effectiveness, including review of all significant change reports and related documentation.

1. Assess whether management maintains policies, standards, and procedures that guide the change control process, including defined roles and responsibilities of key personnel in the change control process. Consider management's actions to

   a. Designate personnel with the ability to create or initiate a request for a change and provide the processes they should follow.
   b. Designate an individual (e.g., change manager) to facilitate a change who is responsible for managing all changes affecting the entity with minimal or no disruptions to operations.
   c. Consider using a group of stakeholders that can aid the change manager in the assessment, prioritization, and scheduling of changes.
   d. Prioritize and categorize changes.
   e. Implement a method to track and report changes (e.g., standard change request forms, library and version controls, and spreadsheets or automated change logs).

2. Assess management's defined change control process to make routine and planned changes to systems or components, adjust configuration settings, and make changes to remediate flaws. Consider management's actions to

   a. Follow a defined change control process to make routine and planned changes to systems or components, adjust configuration settings, and make changes to remediate flaws.
   b. Account for
      - Emergency and unscheduled changes in the change control process.
      - Preparations for unexpected problems or failures with the change by defining a back-out plan and documenting the steps taken during the change in the order they occurred.
      - Requests for change.
      - Review of changes.
      - Approval of changes.
      - Design and build of changes.
      - Testing of changes.
      - Implementation of changes.
      - Verification of changes and close of the change process.
   c. Follow processes, after verification of changes, to document completion of the change and close the change request.
   d. Report on the status of the change after closing and monitor the implemented change for unintended issues.
   e. Integrate security into its change control processes to help ensure that modifications do not adversely affect the security posture of varying systems.
   f. Define the implementation plan (i.e., steps to deploy the change), create a full copy of the current version in production, and finalize the back-out plan before deploying the change.
   g. Perform a post-implementation review to verify that the change was deployed according to the implementation plan and functions appropriately.

h. Follow processes after verification to document completion of the change and close the change request.
i. Report on the status of the change after closing and monitor the implemented change for unintended issues.

3. Assess additional controls in change management. Consider management's actions to

a. Isolate and protect the testing environment to avoid it being a vulnerable vector for exploit in the entity.
b. Strictly control the movement of programs and files among development, quality management, and production libraries for purposes of change control.
c. Assign librarian functions to independent personnel, such as quality management personnel in larger, or more complex entities, or to nonoperations personnel in smaller or less complex entities.
d. Validate that library attributes include an auditable activity log with appropriate controls to ensure that information has not been altered or deleted.
e. Implement library controls, such as
   - Automated access controls.
   - Automated library applications.
f. Consider using these automated change controls commensurate with the complexity of the entity's IT environment and the number, types, and complexities of changes in that environment.
g. Strictly control access to production software libraries, particularly in distributed environments, regardless of whether the entity has automated change control tools.
h. Establish appropriate policies for the use of code repositories, such as determining what code repositories they use and where code repository data actually resides, especially for any cloud-based repositories (e.g., public, private, and community clouds).
i. Conduct periodic reconnaissance of open (i.e., public) code repositories for personal employee accounts and unauthorized posting of company-owned source code.
j. Use available version control features to track changes made to the code in the repository, providing accountability to the individual account that made the changes.
k. Implement access control processes, such as the following:
   - Request and approval processes for access to code repositories (e.g., development, staging, or production).
   - Appropriate access controls, such as use of MFA for employee access to all (internal and cloud-based) code repositories.
   - Appropriate segregation of duties to prevent developer access to the staging and production code repositories, prevent quality management personnel from having access to production code repositories, and grant access to production code repositories only to release management personnel.
   - Periodic review processes for access roles and repository logs.

4. Assess management's oversight of change or modification types. Consider management's actions to

a. Develop procedures for changes that include change request, review, and approval that direct management to plan, test, and document changes before implementation.
b. Validate that changes to any IT system, component, or service are supported by an orderly, adaptable, documented, and measurable process to promote the consistent implementation of changes and provide an audit trail for changes, regardless of the change type.
c. Train personnel involved in changes to ensure that those changes support entity objectives and do not adversely affect confidentiality, integrity, availability, and resilience.
d. Coordinate change management for routine modifications, as IT changes often affect multiple business lines.
e. Develop an inventory of changes, including routine modifications, for back-out, resilience, and tracking purposes.

5. Assess management's oversight of planned changes. Consider management's actions to

a. Perform a risk assessment for all major modifications and significant security-related changes.
b. Discuss requests for major modifications formally and assign appropriate key stakeholders or committees to have responsibility for approving the requests, based on predefined criteria.
c. Define the entity's training requirements associated with conversions or major hardware and software upgrades.
d. Evaluate the type, volume, and timing of training needs for each affected line of business and coordinate training programs with applicable third parties.
e. Plan when a conversion affects a core platform supporting business operations whether managed on-premises or at a third party to minimize conversion issues and costs.
f. Consider costs related to deconversions, system and component upgrades, and training for changes with the new system.
g. Effectively manage conversions beginning with due diligence, including a comprehensive analysis of a conversion's impact on existing operations (e.g., processing, storage, and communication requirements), while accounting for interdependencies.
h. Establish communication procedures, reporting needs, and lines of authority for timely decision-making and issue resolution, when working with a third-party partner.
i. Coordinate with conversion partners to consider the entity's conversion needs.

6. Assess management oversight of emergency modifications. Consider management's actions to

a. Maintain appropriate implementation control standards, although an emergency modification should be completed quickly.
b. Periodically review the change control processes to optimize efficiency and determine whether they need revision.

c. Perform testing before deployment in the case of an emergency change, if possible, as untested changes may cause even more damage than the initial risk being mitigated.
d. Evaluate whether the potential damage resulting from an untested emergency change outweighs the immediate damage from the identified risk to determine the appropriateness of the decision to implement the emergency change.
e. Develop a process for system and component owners to identify all sources of change to help ensure that emergency modifications go through the change control process, even if it is after the fact.
f. Accelerate change management processes to implement changes rapidly to effectively mitigate the impact of emergency situations.
g. Develop effective processes to address emergency situations.
h. Strategically plan for emergency modifications, including budgeting contingency funds for execution and management of emergency situations.
i. Consider appropriate measures to help effectively implement changes during emergencies. Examples include the following:
    - Organize a group (e.g., emergency change control review board) that can meet on extremely short notice to approve changes that need immediate implementation. The group could include senior leadership, management of critical lines of business, and management of key IT areas.
    - Plan for multiple methods of meeting (e.g., in-person, phone, and virtual) in case the emergency occurs outside normal business hours.
    - Create an emergency modification implementation plan that includes a list of individuals in the emergency group (e.g., emergency change control review board) and a list of procedures that are modified from standard change control and security processes.
    - Compress certain phases of the implementation process to save time but still perform testing and modeling or simulation before releasing the changes into the production environment.
    - Discuss and implement minimum recovery point objectives with stakeholders for emergencies to reduce the time required to perform backups of systems to adequately implement an emergency modification.
    - Define and use emergency downtime procedures (e.g., failover to alternate systems) that allow systems to be taken completely offline in emergencies.
    - Follow emergency procedures, such as use of "out-of-band" communication provisions, if normal channels for secure transmission of information becomes unavailable.
    - Leverage senior management to communicate the urgency of emergency modifications and help achieve rapid consensus for decisions affecting the change implementation timeline.
    - Perform periodic emergency drills in the test and modeling environment to prepare IT staff to deal with the stress of emergency modifications. Review the results of these drills and document lessons learned.
j. Plan for emergency modifications, such as doing the following:
    - Establish who can declare an emergency and facilitate decision-making during emergencies.

- Designate individuals who can approve emergency changes and maintain contact information.
- Develop a plan that outlines the responsible parties and procedures for enacting emergency changes.
- Identify personnel responsible for change testing and implementation, as well as for initiating a back-out plan if implementation fails.

k. Validate every emergency modification after implementation to determine whether it was correctly classified, in order to keep the number of emergency modifications to a minimum.
l. Review reports of emergency modifications periodically for appropriateness of classification.
m. Control attempts by personnel to circumvent routine modification or planned change control processes by classifying changes as emergencies to avoid resource constraints (e.g., time, personnel, and cost).
n. Complete evaluations and documentation reviews of emergency modifications as soon as possible after implementation.

7. Assess management's change management documentation practices. Consider management's actions to

a. Maintain documentation to track all changes (e.g., configuration settings, patches applied, system conversions, and emergency modifications) to systems and components.
b. Maintain appropriate documentation to support the impact analysis.
c. Prepare for the possibility of a failed or incomplete deployment and have procedures to address the issues and develop rollback or back-out procedures to reverse a failed deployment.
d. Consider the time required to perform rollback procedures, when to trigger the rollback plan, and how long it may take to roll back.
e. Determine whether the rollback plan can be accomplished in the defined timeline for the maintenance action.
f. Implement appropriate security controls to prevent unauthorized rollback.
g. Track and report issues that lead to the use of a back-out plan and the effects of a failed or incomplete change deployment.

Objective 15: Communicate and discuss findings, conclusions, and corrective actions.

1. Review preliminary conclusions with the examiner-in-charge regarding

a. Violations of law and regulation.
b. Significant issues warranting inclusion as matters requiring attention or recommendations in the report of examination.
c. Proposed Uniform Rating System for Information Technology management component rating and the potential impact of the examiner's conclusions on composite or other component IT ratings.
d. Potential impact of the examiner's conclusions on the entity's risk assessment.

2. Discuss findings with management and obtain proposed corrective action for significant deficiencies.
3. Document conclusions in a memorandum to the examiner-in-charge that provides report-ready comments for all relevant sections of the report of examination and guidance to future examiners.

4. Organize work papers to ensure clear support for significant findings by examination objective.

APPENDIX B: GLOSSARY

The purpose of the glossary is to define technical terms used in the *FFIEC IT Examination Handbook* booklets in the context of supervisory activities for the entities over which FFIEC members may have supervisory authority. The FFIEC members strive to align terminology in the glossary with appropriate authoritative standards, including the *NIST Computer Security Resource Center Glossary* (NIST Glossary) as the primary source for cyber-related definitions, as appropriate. FFIEC members employed the following process to select, modify, or develop definitions.

When a NIST definition existed:

- If NIST had a defined term and modifications to the definition were unnecessary, the FFIEC members included the NIST definition in this glossary. When multiple NIST definitions were available for the same term, the FFIEC members selected a definition for supervisory purposes.
- If NIST had a defined term, but the definition needed additional clarity for supervisory purposes to assist with the identification of safety and soundness and enterprise risks related to IT, the FFIEC members included both the NIST definition and the FFIEC-adapted definition. These definitions are labeled “FFIEC Adapted for Supervisory Purposes” in this glossary’s source column.

When a NIST definition did not exist, or the definition was not appropriate for supervisory purposes:

- If NIST did not have a defined term, but there was an appropriate authoritative third-party source (e.g., the ISO Glossary), the FFIEC members included that authoritative definition.
- If NIST did not have a defined term and there was not an appropriate authoritative third-party source, the FFIEC members developed a definition for supervisory purposes. These definitions are labeled “FFIEC Developed for Supervisory Purposes” in this glossary’s source column.

Due to the constantly evolving nature of IT and its associated risks, the FFIEC members may update definitions to maintain alignment with other government agencies and the financial services industry.

Table 5: Glossary of Terms for Development, Acquisition, and Maintenance

Term	Definition	Source
	A	
Access control	Procedures and controls that limit or detect access to critical information resources. This can be accomplished through software, biometrics devices, or physical access to a controlled space.	NIST Glossary
Acquisition	All stages of the process of acquiring a product or services, beginning with the process for determining the need for the product or services and ending with contract completion and closeout.	NIST Glossary
Application	A system for collecting, saving, processing, and presenting data by means of a computer. The term “application” is generally used when referring to a component of software that can be executed. The	NIST Glossary

Term	Definition	Source
	terms "application" and "software application" are often used synonymously.	
Application programming interface (API)	A system access point or library function that has a well-defined syntax and is accessible from application programs or user code to provide well-defined functionality.	NIST Glossary
	Software code that allows two or more different programs to communicate with each other.	FFIEC Adapted for Supervisory Purposes
Architecture	Refers to the manner in which the strategic design of the hardware and software infrastructure components (e.g., devices, systems, and networks) are organized and integrated to achieve and support the entity's business objectives.	FFIEC Developed for Supervisory Purposes
Artificial intelligence (AI)	Refers to the ability of machines to perform tasks that normally require human intelligence—for example, recognizing patterns, learning from experience, drawing conclusions, making predictions, or taking action—whether digitally or as the smart software behind autonomous physical systems.	U.S. Department of Defense, *Summary of the 2018 Department Of Defense Artificial Intelligence Strategy*
Assembler	A computer program that automatically converts instructions written in assembly language into machine language.	*Merriam-Webster*
Attack surface	The set of points on the boundary of a system, a system component, or an environment where an attacker can try to enter, cause an effect on, or extract data from, that system, component, or environment.	NIST Glossary
Authentication	A process that establishes the source of information, provides assurance of an entity's identity or provides assurance of the integrity of communications sessions, messages, documents or stored data.	NIST Glossary
	A process designed to establish the source of the information, validity of a transmission, message, or originator, or a means of verifying an individual's authorization to receive specific categories of information.	FFIEC Adapted for Supervisory Purposes
Authorization	The granting or denying of access rights to a user, program, or process.	NIST Glossary
Availability	Ensuring timely and reliable access to and use of information.	NIST Glossary
	B	
Backdoor	An undocumented way of gaining access to a computer system. A backdoor is a potential security risk.	NIST Glossary
Backup	A copy of files and programs made to facilitate recovery, if necessary.	NIST Glossary
Barcode	An optical machine-readable representation of data about an object.	NIST, "Automated Identification Technologies for Forensic Science"
Baseline configuration	A set of specifications for a system, or configuration item within a system, that has been formally reviewed and agreed on at a given point in time, and which can be changed only through change control procedures. The baseline configuration is used as a basis for future builds, releases, and/or changes.	NIST Glossary
Big data	Extensive datasets—primarily in the characteristics of volume, variety, velocity, and/or variability—that require a scalable architecture for efficient storage, manipulation, and analysis.	NIST SP 1500-1r2, *NIST Big Data Interoperability Framework: Volume1, Definitions*
Blockchain	A distributed digital ledger of cryptographically signed transactions that are grouped into blocks. Each block is cryptographically linked to the previous one (making it tamper evident) after validation and undergoing a consensus decision. As new blocks are added, older	NIST Glossary

Term	Definition	Source
	blocks become more difficult to modify (creating tamper resistance). New blocks are replicated across copies of the ledger in the network, and any conflicts are resolved automatically using established rules.	
Bounded context	In relation to microservices, it refers to having limited responsibility and dependence on other services.	NIST SP 800-204, *Security Strategies for Microservices-Based Application Systems*
Business case	A business case defines the value a project will deliver.	Project Management Institute (PMI), "Is This Really Worth the Effort? The Need for a Business Case"
	An evaluation of the benefit, cost, feasibility, and risk of alternative options to justify undertaking a project, program, or portfolio and define the value it will deliver.	FFIEC Adapted for Supervisory Purposes
	C	
Capacity management	The process of planning and monitoring an entity's technology resources to support current and future strategic objectives.	FFIEC Developed for Supervisory Purposes
Central processing unit (CPU)	Computer hardware that houses the electronic circuits that control/direct all operations of the computer system.	Information Systems Audit and Control Association (ISACA) Glossary
Change control	Change control is the process through which all requests to change the approved baseline of a project, program, or portfolio are captured, evaluated and then approved, rejected, or deferred.	Association for Project Management
	Change control is the process through which all requests to change the approved baseline of a project, program, or portfolio are documented, evaluated and then approved, rejected, or deferred. Change control is an element in an overall change management process.	FFIEC Adapted for Supervisory Purposes
Change control board (CCB)	A group of qualified people with responsibility for the process of regulating and approving changes to hardware, firmware, software, and documentation throughout the development and operational life cycle of an information system. Also referred to as a configuration control board.	NIST Glossary
Change management	The continuous process of maintaining the integrity of hardware, software, firmware, and documentation and controlling and approving changes (e.g., addition, modification, or elimination) to information or technology assets or related infrastructure (aka configuration and change management).	CISA, *CRR Supplemental Resource Guide, Volume 3: Configuration and Change Management, Version 1.1*
Checksum	A value computed on data to detect error or manipulation.	NIST Glossary
Circuit	A dedicated single connection between two end points on a network.	NIST Glossary
Cloud access security broker	A software tool or service that sits between an entity's on-premises infrastructure and a cloud service provider's infrastructure as a "gatekeeper" to monitor activity and enforce the entity's security policies (e.g., authentication, single sign-on, authorization, credential mapping, and encryption) as the cloud-based resources are accessed.	FFIEC Developed for Supervisory Purposes
Cloud computing	A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g.,	NIST Glossary

Term	Definition	Source
	networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.	
Cloud service provider	A cloud service provider, or CSP, is a company that offers some component of cloud computing; typically, when you search the internet a cloud service is defined as infrastructure as a service (IaaS), software as a service (SaaS), or platform as a service (PaaS) to other businesses or individuals.	Cloud Security Alliance
	A third-party service provider who offers clients services over the public internet. Examples of services could be applications (SaaS), operating systems (PaaS), or infrastructure (IaaS).	FFIEC Adapted for Supervisory Purposes
Code	See machine code.	
Commercial off-the-shelf (COTS) software	A software and/or hardware product that is commercially ready-made and available for sale, lease, or license to the general public. Also referred to as off-the-shelf.	NIST Glossary
Compiled	Processed through software that translates a complete set of high-level computer instructions into machine language before executing any of them.	*Merriam-Webster*
Component	A unique part of a system that is needed to do something, perform a function, or finish a product.	*Black's Law Dictionary*
	A unique part of a larger system, such as hardware, software, and firmware, that is a building block to do something, perform a function, or finish a product.	FFIEC Adapted for Supervisory Purposes
Compromise	The unauthorized disclosure, modification, substitution, or use of sensitive data (e.g., keying material and other security-related information).	NIST Glossary
Confidentiality	The property that sensitive information is not disclosed to unauthorized entities.	NIST Glossary
Configuration	The selection of one of the sets of possible combinations of features of a system.	NIST Glossary
	The selection of combinations of conditions, parameters, features, and specifications of a system.	FFIEC Adapted for Supervisory Purposes
Configuration control board (CCB)	A group of qualified people with responsibility for the process of regulating and approving changes to hardware, firmware, software, and documentation throughout the development and operational life cycle of an information system. Also referred to as a change control board.	NIST Glossary
	A collection of activities focused on establishing and maintaining the integrity of information technology products and systems, through control of processes for initializing, changing, and monitoring the configurations of those products and systems throughout the system development life cycle.	FFIEC Adapted for Supervisory Purposes
Configuration settings	The set of parameters that can be changed in hardware, software, or firmware that affect the security posture and/or functionality of the information system.	NIST Glossary
Container	A container is a standard unit of software that packages up code and all its dependencies, down to, but not including the operating system (OS). It is a lightweight, standalone, executable package of software that includes everything needed to run an application except the OS: code, runtime, system tools, system libraries, and settings.	*DOD Enterprise DevSecOps Reference Design: Version 1.0*
Container image	A package that contains all the files required to run a container.	NIST SP 800-190, *Application Container Security Guide*

Term	Definition	Source
Container runtime	The environment for each container; comprised of binaries coordinating multiple operating system components that isolate resources and resource usage for running containers.	NIST Glossary
	Software that coordinates multiple operating system components to isolate resource and resource usage for running containers and managing container images on a node.	FFIEC Adapted for Supervisory Purposes
Continuous integration/ continuous delivery or deployment (CI/CD) pipeline	The set of tools and the associated process workflows to achieve CI and CD with build, test, security, and release delivery activities, which are steered by a CI/CD orchestrator and automated as much as practice allows.	*DOD Enterprise DevSecOps Fundamentals*
Contract	A binding agreement between two or more parties, which may indicate prices and goods and services to be provided.	*Merriam-Webster*
Copyright	A type of intellectual property that protects original works of authorship as soon as an author fixes the work in a tangible form of expression.	U.S. Copyright Office
Counterfeit item	A counterfeit item is a suspect item that is a copy or substitute without legal right or authority to do so or one whose materials, performance, or characteristics are knowingly misrepresented by the vendor, supplier, distributor, or manufacturer. An item that does not conform to established requirements is not normally considered an S/CI if the nonconformity results from one or more of the following conditions, which should be controlled by site procedures as nonconforming items: defects resulting from inadequate design or production quality control; damage during shipping, handling, or storage; improper installation; deterioration during service; degradation during removal; failure resulting from aging or misapplication; or other controllable causes.	U.S. Department of Energy, *Suspect/ Counterfeit Items Awareness Training*
Cyber risk	Risk of financial loss, operational disruption, or damage from the failure of the digital technologies employed for informational and/or operational functions introduced to a manufacturing system via electronic means from the unauthorized access, use, disclosure, disruption, modification, or destruction of the manufacturing system.	NIST Glossary
	Risk of financial loss, operational disruption, or damage from the failure of the digital technologies employed for informational and/or operational functions introduced to a system via electronic means from the unauthorized access, use, disclosure, disruption, modification, or destruction of the system.	FFIEC Adapted for Supervisory Purposes
Cybersecurity	The process of protecting consumer and bank information by preventing, detecting, and responding to attacks.	NIST Glossary
	D	
Dashboard	A tool that consolidates and communicates information relevant to the entity in near real-time. It is generally visual and often uses a variety of charts.	NIST SP 800-137, *Information Security Continuous Monitoring for Federal Information Systems and Organizations*
Data	A representation of information as stored or transmitted.	NIST Glossary
	A physical or digital representation of information processed, stored (at rest), or transmitted (in transit).	FFIEC Adapted for Supervisory Purposes
Data analytics	The systematic process of evaluating and organizing data sets to draw insights, make predictions, and reveal trends using logical analysis.	FFIEC Developed for Supervisory Purposes
Data center	A facility that houses virtual and/or physical information technology infrastructure(s) (e.g., computer, server, and networking systems and components) designed to store, process, and serve large amounts of data in support of an entity's strategic and business	FFIEC Developed for Supervisory Purposes

Term	Definition	Source
	objectives. A data center may be a dedicated facility or an area or room that contains computer, server, and networking systems and components, and may be private or shared (e.g., a co-location facility).	
Data classification	Categorizing data based on its level of sensitivity (e.g., confidentiality, integrity, or availability), value, and criticality to the entity.	FFIEC Developed for Supervisory Purposes
Data communications	The transfer of data over networks using a combination of telecommunication services and network devices.	FFIEC Developed for Supervisory Purposes
Data governance	A set of processes that ensures that data assets are formally managed throughout the enterprise. A data governance model establishes authority and management and decision-making parameters related to the data produced or managed by the enterprise.	NIST Glossary
Data management	The practice of putting into place policies, procedures and best practices to ensure that data are understandable, trusted, visible, accessible and interoperable.	*DHS Lexicon Terms and Definitions*
	The practice of putting into place policies, procedures, and best practices to ensure that data are understandable, trusted, visible, accessible, and interoperable to ensure that user needs are met.	FFIEC Adapted for Supervisory Purposes
Database	A repository of information or data, which may or may not be a traditional relational database system.	NIST Glossary
	A repository of information or data organized to be accessed, managed, and updated.	FFIEC Adapted for Supervisory Purposes
Developer	A person or group that designs and/or builds, and/or documents and/or configures the hardware and/or software of computerized systems	ISACA Glossary
	A person or group that designs, builds, documents, and configures the hardware and software of systems.	FFIEC Adapted for Supervisory Purposes
Development	The systematic application of knowledge toward the production of useful materials, devices, and systems or methods that leverage the results of applied research activities; includes validation and demonstration of a chosen technology in laboratory, representative and operational environments, improving on research prototypes, integration into systems and subsystems, addressing manufacturing, producibility and sustainability needs, and independent operational test and evaluation.	*DHS Lexicon Terms and Definitions*
	The systematic application of knowledge toward the production of useful materials, devices, and systems or processes of defining, designing, testing, and implementing systems or components. Development includes validation and demonstration of a chosen technology, use of test and production environments, improvement of developed prototypes, integration into systems and subsystems, and inclusion of hardware builds.	FFIEC Adapted for Supervisory Purposes
Device	A piece of equipment or a mechanism designed to serve a special purpose or perform a special function.	*Merriam-Webster*
DevOps	A set of practices for automating the processes between software development and information technology operations teams so that they can build, test, and release software faster and more reliably. The goal is to shorten the SDLC and improve reliability while delivering features, fixes, and updates frequently in close alignment with business objectives.	NIST Glossary

Term	Definition	Source
DevSecOps	An organizational software engineering culture and practice that aims at unifying software development (Dev), security (Sec) and operations (Ops).	NIST SP 800-204C, *Implementation of DevSecOps for a Microservices-Based Application With Services Mesh*
	A software engineering culture and practice that aims at unifying software development (Dev), security (Sec), and operations (Ops). The main characteristic of DevSecOps is to automate, monitor, and apply security at all phases of the SDLC.	FFIEC Adapted for Supervisory Purposes
Digital signature	An electronic analogue of a written signature, a digital signature provides assurance that the claimed signatory signed the information, and the information was not modified after signature generation.	NIST Computer Security Resource Center
Dynamic analysis	Testing that operates by executing a program using a set of input use-cases and analyzing the program's runtime behavior.	*OWASP Vulnerability Management Guide (OVMG)*
	E	
Encryption	Any procedure used in cryptography to convert plain text into cipher text to prevent anyone but the intended recipient from reading that data.	NIST Glossary
End-of-life (EOL)	With respect to technology, a time frame usually defined by a technology vendor to describe when an asset has reached the end of its useful life cycle and when the vendor no longer maintains and supports the asset or continues to sell or license it.	FFIEC Developed for Supervisory Purposes
	F	
Feasibility study	An analysis of the known or anticipated need for a product, system or component to assess the degree to which the requirements, designs or plans can be implemented.	ISACA Glossary
	An analysis of a known problem or anticipated need and its potential solutions that considers all relevant factors—including economic, technical, legal, and scheduling considerations—to determine a project's viability, costs, and benefits.	FFIEC Adapted for Supervisory Purposes
Firewall	A gateway that limits access between networks in accordance with local security policy.	NIST Glossary
Firmware	Memory chips with embedded program code that hold their content when power is turned off.	ISACA Glossary
	A type of software, often included in read-only memory, that is embedded directly in a piece of hardware to make the hardware work as intended. Software is retained even when power is turned off.	FFIEC Adapted for Supervisory Purposes
Fourth party	Fourth (or nth) parties provide services to support the operations of third parties, which, in turn, provide services to the primary enterprise. Also referred to as subcontractor.	ISACA
Functional testing	Testing that verifies that an implementation of some function operates correctly.	NIST Glossary
Fuzz testing	A black box software testing technique, which basically consists in finding implementation bugs using malformed/semi-malformed data injection in an automated fashion by tools referred to as "fuzzers," which are programs or scripts that submit some combination of inputs to the test target to reveal how it responds.	OWASP, *Code Review Guide 2.0*
	A black box software testing technique, which basically consists in finding implementation bugs using malformed/semi-malformed data injection in an automated fashion.	FFIEC Adapted for Supervisory Purposes
	G	
Gateway	An intermediate system (interface, relay) that attaches to two (or more) computer networks that have similar functions but dissimilar implementations and that enables either one-way or two-way communication between the networks.	NIST Glossary

Term	Definition	Source
H		
Hardening	A process to eliminate as many security risks as possible by removing all nonessential software programs, protocols, services, and utilities from the system. (Referred to as system hardening)	ISACA Glossary
	A process intended to eliminate as many security risks as possible by implementing security controls (e.g., changing default passwords, enabling security settings, and protecting privileged accounts), patching vulnerabilities, turning off nonessential services, and removing all nonessential software programs, protocols, and utilities from the system.	FFIEC Adapted for Supervisory Purposes
Hardware	The physical components of an information system.	NIST Glossary
Help desk	A service offered via telephone/Internet by an enterprise to its clients or employees that provides information, assistance and troubleshooting advice regarding software, hardware, or networks.	ISACA Glossary
Heterogeneity	Consisting of dissimilar or diverse elements.	*Merriam-Webster*
Heuristics	Involving or serving as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods.	*Merriam-Webster*
Host	Any hardware device that has the capability of permitting access to a network via a user interface, specialized software, network address, protocol stack, or any other means. Some examples include computers, personal electronic devices, thin clients, and multifunctional devices.	NIST Glossary
Human-readable code	Code that includes source code, scripts, and any other form of code that an organization deems human-readable.	NIST SP 800-218, *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities*
Hypertext transfer protocol (HTTP)	A standard method for communication between clients and web servers.	NIST Glossary
Hypertext transfer protocol secure (HTTPS)	A protocol for accessing a secure web server, whereby all data transferred are encrypted. Standard port number is 443.	ISACA Glossary
Hypervisor	The virtualization component that manages the guest operating systems (OS) on a host and controls the flow of instructions between the guest OSs and the physical hardware.	NIST Glossary
I		
Identity and access management (IAM)	Encapsulates people, processes, and products to identify and manage the data used in an information system to authenticate users and grant or deny access rights to data and system resources.	ISACA Glossary
Impact analysis	The analysis conducted by an organizational official to determine the extent to which a change to the information system have affected the security state of the system. Referred to as a security impact analysis.	NIST SP 800-128, *Guide for Security-Focused Configuration Management of Information Systems*
	The analysis conducted by qualified staff in an organization to determine the extent to which changes to the system affect the security posture of the system. This may involve security, resilience, and compliance.	FFIEC Adapted for Supervisory Purposes
Incident	An occurrence that actually or potentially jeopardizes the confidentiality, integrity, or availability of a system or the information the system processes, stores, or transmits or that constitutes a violation or imminent threat of violation of security policies, security procedures, or acceptable use policies.	NIST Glossary

Term	Definition	Source
Incident management	The process of identifying, analyzing, and correcting disruptions to operations and preventing future recurrences. The goal of incident management is to limit the disruption and restore operations as quickly as possible.	FFIEC Developed for Supervisory Purposes
Incident response	Responds to crises or urgent situations within the pertinent domain to mitigate immediate and potential threats. Uses mitigation, preparedness, and response and recovery approaches, as needed, to maximize survival of life, preservation of property, and information security. Investigates and analyzes all relevant response activities.	DOD Cyber Exchange
Information security	The protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide confidentiality, integrity, and availability.	NIST Glossary
Information technology asset management (ITAM)	Refers to a set of policies and procedures that an organization uses to track, audit, and monitor the state of its IT assets, and maintain system configurations.	NIST SP 1800-5, *IT Asset Management*
Infrastructure	System of facilities, equipment, and services needed for the operation of an organization.	ISO Online Browsing Platform, Terms & Definitions
	The physical elements, products, and services necessary to provide and maintain ongoing operations to support business activity, including the maintenance of physical facilities.	FFIEC Adapted for Supervisory Purposes
Infrastructure as a service (IaaS)	IaaS provides entities with the ability to provision processing, storage, networks, and other fundamental computing resources where the entity is able to deploy and run software, which can include operating systems and applications. The entity does not manage or control the underlying cloud infrastructure; however, it has control over operating systems, storage, and deployed applications. Entities have the maximum flexibility to customize their cloud services and user interfaces.	NIST SP 500-316, *Framework for Cloud Usability*
Integration testing	An orderly progression of testing in which software elements, hardware elements or both are combined and tested to evaluate their interactions, until the entire system has been integrated.	ISACA Glossary
Integrity	A property whereby data has not been altered in an unauthorized manner since it was created, transmitted or stored.	NIST Glossary
Interdependencies	When two or more departments, processes, functions, or third-party providers interact to successfully complete a task, business function, or process.	FFIEC Developed for Supervisory Purposes
Internet of things (IoT)	Refers to the collection of technologies that allow information to be sent to and received from physical devices (e.g., security systems, HVAC systems, intelligent personal assistants, and kitchen appliances), that were not traditionally thought of as IT assets, using the internet. These devices have the ability to send and receive data over a network without necessarily requiring human-to-human or human-to-computer interaction using embedded computing capability and network connectivity and unique identifiers (e.g., internet protocol address).	FFIEC Developed for Supervisory Purposes
Intrusion detection system (IDS)	A security service that monitors and analyzes network or system events for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorized manner.	NIST Glossary
Intrusion prevention system (IPS)	A system that can detect an intrusive activity and can also attempt to stop the activity, ideally before it reaches its targets.	NIST Glossary
	K	
Kernel	Core that provides basic services for all other parts of the OS.	FFIEC Developed for Supervisory Purposes

Term	Definition	Source
L		
Legacy	A custom environment containing older systems or applications that may need to be secured to meet today's threats, but often use older, less secure communication mechanisms and need to be able to communicate with other systems (aka legacy environment).	NIST Glossary
	Relates to older, outdated technology systems, components, or applications that may be critical to day-to-day operations. Often, they may need to be secured, updated, or replaced to meet current threats.	FFIEC Adapted for Supervisory Purposes
License	A permission granted by competent authority to engage in a business or occupation or in an activity otherwise unlawful.	*Merriam-Webster*
Log	A record of events occurring within an entity's systems and networks.	NIST Glossary
Loose coupling	An approach to designing systems so that linked components of networks, software, and services can be scaled to operate as independently as possible to avoid issues with one component adversely affecting others.	FFIEC Developed for Supervisory Purposes
M		
Machine code	Also referred to as "code" or "machine language." Computer instructions and definitions expressed in a form (binary code) that can be recognized by the CPU of a computer. All source code, regardless of the language in which it was programmed, is eventually converted to machine code.	ISACA Glossary
Machine language	See *machine code*.	
Machine learning (ML)	The process of using an algorithm(s) to help computers learn without being explicitly programmed and identify patterns in data. Those patterns are then used to create a data model that can make predictions.	FFIEC Developed for Supervisory Purposes
Mainframe	A large, high-speed computer, especially one supporting numerous workstations or peripherals.	ISACA Glossary
Maintenance	Any act that either prevents the failure or malfunction of equipment or restores its operating capability.	NIST Glossary
Malicious code	Unwanted files or programs that can cause harm to a computer or compromise data stored on a computer. Various classifications of malicious code include viruses, worms, and Trojan horses.	CISA, "Protecting Against Malicious Code"
Malware	A program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim's data, applications, or operating system or of otherwise annoying or disrupting the victim.	NIST Glossary
Markup language	A system (such as HTML or SGML) for marking or tagging a document that indicates its logical structure (such as paragraphs) and gives instructions for its layout on the page especially for electronic transmission and display.	*Merriam-Webster*
	A type of language used to annotate text and embed tags in a text document to control its structure, formatting, display, or the relationship between its parts to facilitate automated processing and use by humans.	FFIEC Adapted for Supervisory Purposes
Masking	The process of systematically removing a field or replacing it with a value in a way that does not preserve the analytic utility of the value, such as replacing a phone number with asterisks or a randomly generated pseudonym.	NIST Glossary
Master services agreement (MSA)	One legal document that consolidates separate but related agreements between the same signing parties. Also known as master contract.	*Black's Law Dictionary*
	Also known as a "master contract," it is a legal document that consolidates and governs multiple service agreements or transactions between the same signing parties or entities, and often addresses the terms that will govern future transactions or agreements.	FFIEC Adapted for Supervisory Purposes

Term	Definition	Source
Metadata	Data about data. For file systems, metadata are data that provide information about a file's contents.	NIST Glossary
	Data about data. Examples of metadata include purpose of the data, creator or owner of the data, file size, location where the data were created, and source of the data.	FFIEC Adapted for Supervisory Purposes
Microchip	A very small piece of silicon inside a computer. It has electronic circuits on it and can hold large quantities of information or perform mathematical and logical operations.	*Collins Dictionary*
Microcontroller	An integrated circuit that contains a microprocessor along with memory and associated circuits and that controls some or all of the functions of an electronic device (such as a home appliance) or system.	*Merriam-Webster*
Microservices	A set of containers that work together to compose an application.	NIST Glossary
Middleware	Middleware is a term used for a computer program that functions as a go between for two other programs.	DHS, *Access Control Technologies Handbook*
Misdirection	The process of maintaining and employing deception resources or environments and directing adversary activities to those resources or environments.	NIST Glossary
Mobile device	A portable computing device that: (i) has a small form factor such that it can easily be carried by a single individual; (ii) is designed to operate without a physical connection (e.g., wirelessly transmit or receive information); (iii) possesses local, non-removable data storage; and (iv) is powered-on for extended periods of time with a self-contained power source.	NIST Glossary
Multifactor authentication (MFA)	Authentication using two or more factors to achieve authentication. Factors include: (i) something you know (e.g., password or personal identification number (PIN)); (ii) something you have (e.g., cryptographic identification device or token); or (iii) something you are (e.g., biometric).	NIST Glossary
	N	
Network	A system implemented with a collection of interconnected components. Such components may include routers, hubs, cabling, telecommunications controllers, key distribution centers, and technical control devices.	NIST Glossary
Network diagram	A network diagram (also referred to as a network map or network topology) is a visual representation of nodes and connections in a computer network.	FFIEC Developed for Supervisory Purposes
Networked	A database organized according to ownership of records, allowing records to have multiple owners and thus providing multiples access paths to the data. Database management systems (DBMSs) providing such capabilities are also referred to as CODASYL (Conference on Data Systems Languages) DBMSs.	Gartner Glossary
Network operations center (NOC)	The central location or department responsible for monitoring the health and performance of the network, including analyzing and maintaining network traffic, telecommunications, and network disruptions.	FFIEC Developed for Supervisory Purposes
Nonproduction environment	Systems (e.g., applications, infrastructure, networks, and operating systems) that are not used for production purposes. For example, systems that are used as development or test environments for new software or technologies or changes to existing software or technologies.	FFIEC Developed for Supervisory Purposes
	O	
Object code	A code expressed in machine language ("1"s and "0"s), which is normally an output of a given translation" process that is ready to be executed by a computer.	U.S. Food and Drug Administration, Glossary of Computer System Software

Term	Definition	Source
		Development Terminology
Open-source software	Software released under a license that allows the software and its source code to be accessed, used, modified, and shared by anyone.	NIST S 6106.01, "Open Source Code"
Operating system (OS)	The software "master control application" that runs the computer. It is the first program loaded when the computer is turned on, and its main component, the kernel, resides in memory at all times. The operating system sets the standards for all application programs (such as the Web server) that run in the computer. The applications communicate with the operating system for most user interface and file management operations.	NIST Glossary
Operations	The performance of activities comprising methods, principles, processes, procedures, and services that support business functions.	FFIEC Developed for Supervisory Purposes
Operations management	The process of overseeing the methods, activities, or performance of practical work, and application of principles, processes, procedures, and services of an entity, using business resources.	FFIEC Developed for Supervisory Purposes
Orchestrator	A tool that enables DevOps personas or automation working on their behalf to pull images from registries, deploy those images into containers, and manage the running containers. Orchestrators are responsible for monitoring container resource consumption, job execution, and machine health across hosts.	NIST Glossary
Original equipment manufacturer	Technology providers (e.g., IT hardware vendors, hardware component makers, software vendors, resellers, and distributors) that distribute output devices and services produced by another company under their own brand name.	FFIEC Developed for Supervisory Purposes
Out-of-Band	Communication between parties utilizing a means or method that differs from the current method of communication.	NIST Glossary
	P	
Patch	Fixes to software programming errors and vulnerabilities.	ISACA Glossary
Patch management	The systematic notification, identification, deployment, installation, and verification of operating system and application software code revisions. These revisions are referred to as patches, hot fixes, and service packs.	NIST Glossary
Penetration testing	A test methodology in which assessors, using all available documentation (e.g., system design, source code, manuals) and working under specific constraints, attempt to circumvent the security features of an information system.	NIST Glossary
	A test methodology in which assessors, typically working under specific constraints and using all available documentation, attempt to circumvent or defeat the security features of a system, such as the network, application, its data, or its environment resources. Penetration tests often involve analyzing for individual and combinations of vulnerabilities on a system or multiple systems that can be used to gain access beyond what is achieved through a single vulnerability.	FFIEC Adapted for Supervisory Purposes
Planned obsolescence	The practice of making or designing something in such a way that it will only be usable for a short time so that people will have to buy another one.	*Merriam-Webster*
	A system disposal strategy developed and implemented, as needed, to address technology refresh in budget planning to limit the use of obsolete systems that present security or reliability risks.	FFIEC Adapted for Supervisory Purposes
Platform	A computer or hardware device and/or associated operating system, or a virtual environment, on which software can be installed or run.	NIST Glossary
Platform as a service (PaaS)	The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or	NIST Glossary

Term	Definition	Source
	control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.	
Policies	Statements, rules, or assertions that specify the correct or expected behavior of an entity.	NIST Glossary
Post-implementation review (PIR)	The process for reviewing IT projects in order to learn from past investments and initiatives by comparing actual results to estimates. PIRs also serve as vehicles for evaluating the entire IT investment management process.	U.S. Government Accountability Office, *Information Technology Investment Management*
	An evaluation of projects that provides a feedback mechanism to management that can measure project results, comparing actual results to estimates, and gather data necessary to analyze what corrective action may be required, if necessary.	FFIEC Adapted for Supervisory Purposes
Problem	In IT, the unknown underlying cause of one or more incidents.	ISACA Glossary
Procedures	A document containing a detailed description of the steps necessary to perform specific operations in conformance with applicable standards. Procedures are defined as part of processes.	ISACA Glossary
Procurement	The process of obtaining a system, product, or service.	NIST Glossary
	The corporate processes and functions that involve governance over purchasing decisions for obtaining a system, product, or service.	FFIEC Adapted for Supervisory Purposes
Program	An activity that focuses on the coordination of a number of related projects and other activities, over time, to deliver benefits to the organization.	PMI
Project	An activity chartered to create a specified deliverable as efficiently as possible.	PMI
Project management office (PMO)	Organizational unit that supports the management of projects and project-based organizations.	PMI, "The PMO: Your Key to Strategy Execution and Results Delivery"
Protocol	A set of rules (i.e., formats and procedures) to implement and control some type of association (e.g., communication) between systems.	NIST Glossary
Prototyping	The process of quickly putting together a working model (a prototype) to test various aspects of a design, illustrate ideas or features, and gather early user feedback. Prototyping uses programmed simulation techniques to represent a model of the final system to the user for advisement and critique. The emphasis is on end-user screens and reports. Internal controls are not a priority item since this is only a model.	ISACA Glossary
Provenance	Origin or source.	*Merriam-Webster*
Provisioning	The activity of obtaining the equipment and resources you need for a particular activity.	*Cambridge Dictionary*
	The activity of obtaining, modifying, and making available the equipment, resources, software, or services a user needs to carry out a particular activity.	FFIEC Adapted for Supervisory Purposes
	Q	
Quality assurance (QA)	A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements	ISACA Glossary
	A planned and systematic pattern of all actions necessary to provide adequate confidence that a system, component, or facility conforms to established requirements.	FFIEC Adapted for Supervisory Purposes
Quality assurance/ quality control (QA/QC)	Part of quality management focused on providing confidence that quality requirements will be fulfilled.	NIST Glossary

Term	Definition	Source
Quality control (QC)	The operational techniques and procedures used to achieve quality requirements.	ISACA Glossary
	The operational techniques and procedures (such as design analysis and inspection for defects) used to achieve quality requirements.	FFIEC Adapted for Supervisory Purposes
Quality of service	The measurable end-to-end performance properties of a network service, which can be guaranteed in advance by a Service Level Agreement between a user and a service provider, so as to satisfy specific customer application requirements. Properties may include throughput (bandwidth), transit delay (latency), error rates, priority, security, packet loss, and packet jitter.	Committee on National Security Systems Glossary
Quality management	Coordinated activities to direct and control an organization with regard to quality.	NIST Glossary
Query	In databases, a request for data or information from a table or combination of tables.	FFIEC Developed for Supervisory Purposes
R		
Radio frequency identification (RFID)	A form of automatic identification and data capture technology that uses electric or magnetic fields at radio frequencies to transmit information.	NIST SP 800-98, *Guidelines for Securing Radio Frequency Identification (RFID) Systems*
Random access memory (RAM)	The computer's primary working memory. Each byte of RAM can be accessed randomly regardless of adjacent bytes.	ISACA Glossary
	The computer's primary working memory, which is kept where each byte of RAM can be quickly accessed by the device's processor.	FFIEC Adapted for Supervisory Purposes
Registry	A service that allows developers to easily store images as they are created; tag and catalog images for identification and version control to aid in discovery and reuse; and find and download images that others have created.	NIST SP 800-190, *Application Container Security Guide*
Regression	Defect or bug (in terms of code).	OWASP, *Code Review Guide 2.0*
Release	A collection of new and/or changed configuration items, which are tested and introduced into a production environment together.	NIST Glossary
Relational database	A database organized according to ownership of records, allowing records to have multiple owners and thus providing multiple access paths to the data. Database management systems (DBMSs) providing such capabilities are also referred to as CODASYL (Conference on Data Systems Languages) DBMSs.	Gartner Glossary
Remote access	Access to an organizational information system by a user (or an information system) communicating through an external, nonorganization-controlled network (e.g., the Internet).	NIST Glossary
Report	A detailed account or statement. For purposes of IT, the report provides analysis that supports informed decision-making.	*Merriam-Webster*
Request for information (RFI)	A market research tool used to obtain price, delivery, capabilities, and interest for planning purposes. Also referred to as a sources-sought notice.	GSA, "RFP, RFI, and RFQ: Understanding the Difference"
Request for proposal (RFP)	A solicitation method which communicates the [entity's] requirements and requests proposals.	GSA, "RFP, RFI, and RFQ: Understanding the Difference"
Request for quote (RFQ)	A solicitation method used to obtain price, cost, delivery, and related information from suppliers.	GSA, "RFP, RFI, and RFQ: Understanding the Difference"

Term	Definition	Source
Resilience	The ability to prepare for and adapt to changing conditions and withstand and recover rapidly from disruptions. Resilience includes the ability to withstand and recover from deliberate attacks, accidents, or naturally occurring threats or incidents.	NIST Glossary
Role-based access	A model for controlling access to resources where permitted actions on resources are identified with roles rather than with individual subject identities.	NIST Glossary
Rollback	Updates to earlier firmware versions may provide a means to recover from a firmware update that is not functioning correctly.	NIST SP 800-193, *Platform Firmware Resiliency Guidelines*
	A means to recover from an update that is not functioning correctly or restore to a previous secure and functional version that may require the physical presence of a user.	FFIEC Adapted for Supervisory Purposes
Runtime	The period during which a computer program is executing.	NIST Glossary
	S	
Sanitization	Actions taken to render data written on media unrecoverable by ordinary and, for some forms of sanitization, extraordinary means.	NIST Glossary
Scalability	Refers to how well a hardware and software system can adapt to increased demands. For example, a scalable network system would be one that can start with just a few nodes but can easily expand to thousands of nodes. Scalability can be a very important feature because it means the entity can invest in a system with confidence that the entity will not quickly outgrow it.	FFIEC Developed for Supervisory Purposes
Scheduling	A method used in the information processing facility to determine and establish the sequence of computer job processing.	ISACA Glossary
Scope creep	Also referred to as requirement creep, this refers to uncontrolled changes in a project's scope. Scope creep can occur when the scope of a project is not properly defined, documented, and controlled. Typically, the scope increase consists of either new products or new features of already approved products. Hence, the project team drifts away from its original purpose. Because of one's tendency to focus on only one dimension of a project, scope creep can also result in a project team overrunning its original budget and schedule. For example, scope creep can be a result of poor change control, lack of proper identification of what products and features are required to bring about the achievement of project objectives in the first place, or a weak project manager or executive sponsor.	ISACA Glossary
	Scope creep refers to uncontrolled changes in a project's scope, adding features and functionality without addressing the effects on time, costs, and resources. These changes may result from a lack of effective project change management, without approval (e.g., management or customer), or when the scope is not properly defined, documented, and controlled.	FFIEC Adapted for Supervisory Purposes
Secure code review	The process of auditing the source code of an application to verify that the proper security and logical controls are present, that they work as intended, and that they have been invoked in the right places. Code review aims to identify security flaws in the application related to its features and design, along with the exact root causes.	OWASP, *Code Review Guide 2.0*
	The process of auditing the source code of an application to verify that the proper security and logical controls are present, work as intended, and have been invoked in the right places.	FFIEC Adapted for Supervisory Purposes
Secure development	A hardware and software development approach with an objective of building systems and components while minimizing the potential for vulnerabilities and reducing the attack surface. This is done through measures such as secure coding practices, comprehensive development testing, and built-in security mechanisms (e.g., access control, authentication, and logging capabilities).	FFIEC Developed for Supervisory Purposes
Security	The state in which the integrity, confidentiality, and accessibility of information, service or network entity is assured.	NIST Glossary

Term	Definition	Source
Security impact analysis	The analysis conducted by qualified staff in an organization to determine the extent to which changes to the system affect the security posture of the system.	NIST 800-128, *Guide for Security-Focused Configuration Management of Information Systems*
Security testing	Testing that attempts to verify that an implementation protects data and maintains functionality as intended.	NIST Glossary
	Testing that attempts to verify that a modified or new system implementation includes appropriate controls and does not introduce any security holes that might compromise other systems or allow for any misuse of the system or its information and protects data and maintains functionality as intended.	FFIEC Adapted for Supervisory Purposes
Server	A computer or device on a network that manages network resources. Examples include file servers (to store files), print servers (to manage one or more printers), network servers (to manage network traffic), and database servers (to process database queries).	NIST Glossary
Service deregistration	Removal of a service from the service registry.	FFIEC Developed for Supervisory Purposes
Service discovery	The process by which an application learns what services are available on the network, and by which the network learns what services the application can provide.	Science Direct Topics in Computer Science
Service improvement	The actions taken to identify and execute methods to improve an entity's services and align them with its business objectives.	FFIEC Developed for Supervisory Purposes
Service level agreement (SLA)	Defines the specific responsibilities of the service provider and sets the customer expectations.	NIST Glossary
	A formal agreement between two parties that records a common understanding about products or services to be delivered, priorities, responsibilities, guarantees, and warranties between the parties. In addition, the agreement describes the nature, quality, security, availability, scope, and timeliness of delivery and response of the parties, the point(s) of contact for end-user problems, and the metrics by which the effectiveness of the process is monitored and approved and may include other measurable objectives. The agreement should cover not only expected day-to-day situations, but also unexpected or adverse events, as the need for the service may vary.	FFIEC Adapted Definition for Supervisory Purposes
Service mesh	A distributed computing middleware that optimizes communications between application services.	NIST SP 800-204A, *Building Secure Microservices-Based Applications Using Service-Mesh Architecture*
Service registration	The service registry service is used by microservices that are coming online to publish their locations in a process referred to as service registration and is used by microservices seeking to discover registered services.	NIST Glossary
	The process of microservices using the registry service to publish their locations when they come online. It is used by microservices seeking to discover other registered services.	FFIEC Adapted for Supervisory Purposes
Service registry	A directory where new service instances created for the microservices-based application register themselves while service instances going offline are deleted from it.	NIST SP 800-204A, *Building Secure Microservices-Based Applications*

Term	Definition	Source
		Using Service-Mesh Architecture
Shadow IT	Refers to unauthorized hardware and other devices, software, or services operating in an entity's IT environment.	FFIEC Developed for Supervisory Purposes
Signature	A recognizable, distinguishing pattern associated with an attack, such as a binary string in a virus or a particular set of keystrokes used to gain unauthorized access to a system.	NIST Glossary
Signature-based detection	This intrusion detection method compares network traffic to a set of pre-defined signatures and triggers an alert when a match is detected. Signatures are derived from numerous sources and are specific machine-readable patterns of network traffic that affect the integrity, confidentiality, or availability of computer networks, systems, and information.	DHS, *Privacy Impact Assessment for the National Cybersecurity Protection System (NCPS) - Intrusion Detection*
Single point of failure	An element in the design, configuration, or implementation of a system that can cause the entire system to fail if it stops working.	FFIEC Developed for Supervisory Purposes
Social engineering	The act of deceiving an individual into revealing sensitive information, obtaining unauthorized access, or committing fraud by associating with the individual to gain confidence and trust.	NIST Glossary
Software	Computer programs (which are stored in and executed by computer hardware) and associated data (which also is stored in the hardware) that may be dynamically written or modified during execution.	NIST Glossary
Software as a service (SaaS)	The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.	NIST Glossary
Software bill of material (SBOM)	A formal record containing the details and supply chain relationships of various components used in building software. Software developers and vendors often create products by assembling existing open-source and commercial software components. The SBOM enumerates these components in a product.	NIST Glossary
Source code	Computer commands written in a computer programming language that is meant to be read by people.	U.S. Department of Commerce, "Open Source Code"
	Computer commands written in a computer programming language that is meant to be read by people. Source code is not executable by the computer directly. It must first be converted (e.g., via compiler, assembler) into a machine language.	FFIEC Adapted for Supervisory Purposes
Spiral model	The "Spiral" model divides a software development project into several smaller projects that address the major risks first.	*NIST IR 7499, Guidelines for Planning and Development of Software for Buildings and Building Systems*
	A risk-driven software development process model that divides a software development project into several smaller projects that address the major risks first. The model illustrates activities (e.g., requirements analysis, preliminary and detailed design, coding, integration, and testing) that are performed iteratively until development is complete.	FFIEC Adapted for Supervisory Purposes

Term	Definition	Source
Standard image	The approved set of server configurations, applications, and systems, which can be used to deploy servers consistently and rebuild them more easily and quickly, when necessary.	FFIEC Developed for Supervisory Purposes
Standards	Rules, conditions, or requirements describing the following information for products, systems, services or practices: (i) Classification of components. (ii) Specification of materials, performance, or operations; or (iii) Delineation of procedures.	NIST Glossary
State	A condition or mode of existence in which a system, component or simulation may be, e.g., the preflight state of an aircraft navigation program or the input state of a given channel.	ISACA Glossary
	A condition or mode of existence in which a system, component, or simulation (including stored data and inputs) may be at a given point in time.	FFIEC Adapted for Supervisory Purposes
Stateful	Refers to a data representation or a process that is dependent on an external data store.	NIST Glossary
	Refers to a data representation or a process that is dependent on some external information or other process and has the ability to retain or store the data for reuse.	FFIEC Adapted for Supervisory Purposes
Stateful inspection	Packet filtering that tracks the state of connections and blocks packets that deviate from the expected state.	NIST Glossary
Stateful protocol analysis	A firewalling capability that improves upon standard stateful inspection by adding basic intrusion detection technology. This technology consists of an inspection engine that analyzes protocols at the application layer to compare vendor-developed profiles of benign protocol activity against observed events to identify deviations, allowing a firewall to allow or deny access based on how an application is running over a network.	NIST Glossary
Stateless	Refers to a data representation or a process that is self-contained and does not depend on any external data store.	NIST Glossary
	Refers to a data representation or a process that is self-contained and does not depend on any external information or other process and it does not store or reuse data.	FFIEC Adapted for Supervisory Purposes
Statement of objective (SOO)	When the government is open to a wide range of solutions to meet their objective(s), it will use a SOO. You will need to develop and include in your offer a proposed PWS (i.e., your solutions), performance metrics, a measurement plan, and a quality assurance plan.	GSA
	When an entity is open to a wide range of solutions to meet their objective(s), it may use a SOO. You will need to develop and include in your offer a proposed PWS (i.e., your solutions), performance metrics, a measurement plan, and a quality assurance plan.	FFIEC Adapted for Supervisory Purposes
Statement of work (SOW)	A document that details the developer's responsibilities in the performance of the contract. Documentation developed under the contract, for example, is specified in the SOW. Security assurance requirements, which detail many aspects of the processes the developer follows and what evidence must be provided to assure the organization that the processes have been conducted correctly and completely, may be specified in the SOW.	NIST IR 7622, *Notional Supply Chain Risk Management Practices for Federal Information Systems*
Static analysis	Detecting software vulnerabilities by examining the app source code and binary code and attempting to reason over all possible behaviors that might arise at runtime. Also referred to as static testing.	NIST 800-163, rev. 1, *Vetting the Security of Mobile Applications*
Structured data	Data that has a predefined data model or is organized in a predefined way.	NIST SP 1500-1r2, *NIST Big Data Interoperability Framework: Vol. 1, Definitions*
Subcontractor	An individual or business firm contracting to perform part or all of another's contract.	*Merriam-Webster*

Term	Definition	Source
Supplier	Organization or an individual that enters into an agreement with the acquirer for the supply of a product or service.	NIST Glossary
	External or internal organization, business unit, or individual that enters into an agreement with the acquirer for the supply of a product, component, or service. This includes suppliers, developers, and manufacturers in the supply chain.	FFIEC Adapted for Supervisory Purposes
Supply chain	A system of organizations, people, activities, information, and resources, possibly international in scope, that provides products or services to consumers.	NIST Glossary
Supply chain risk management (SCRM)	The process of identifying, assessing, and mitigating the risks associated with the global and distributed nature of information and communications technology product and service supply chains.	NIST Glossary
	The process of identifying, assessing, and mitigating the risks interconnected throughout the supply chains and their components, such as susceptibilities, vulnerabilities, and threats, associated with the global and distributed nature of information and communications technology product and service supply chains.	FFIEC Adapted for Supervisory Purposes
System	A combination of interacting elements organized to achieve one or more stated purposes.	NIST Glossary
System and organization controls (SOC)	The suite of services CPAs may provide relating to system-level controls of a service organization and system- or entity-level controls of other organizations. Formerly, SOC referred to service organization controls. By redefining that acronym, the American Institute of Certified Public Accountants [AICPA] enables the introduction of new internal control examinations that may be performed (a) for other types of organizations, in addition to service organizations, and (b) on either system-level or entity-level controls of such organizations.	AICPA, *System and Organization Controls: SOC Suite of Services*
System development life cycle (SDLC)	The scope of activities associated with a system, encompassing the system's initiation, development and acquisition, implementation, operation and maintenance, and ultimately its disposal that instigates another system initiation.	NIST Glossary
System test	A test performed on a complete system to evaluate its compliance with specified requirements.	NIST Glossary
	T	
Telecommunications	The transmission, between or among points specified by the user, of information of the user's choosing, without change in the form or content of the information as sent and received.	NIST Glossary
Term of support	The length of time for which the device will be supported by the manufacturer or supporting parties for such actions and materials as part replacements, software updates, vulnerability notices, technical support questions, etc.	NIST Glossary
	The length of time for which the software, hardware, or service will be supported by the manufacturer or supporting parties for such actions and materials as part replacements, software updates, vulnerability notices, and technical support questions.	FFIEC Adapted for Supervisory Purposes
Third-party service provider	Any independent party to whom an entity outsources activities that the entity itself is authorized to perform, including a technology service provider.	FFIEC Developed for Supervisory Purposes
Threat intelligence	Threat information that has been aggregated, transformed, analyzed, interpreted, or enriched to provide the necessary context for decision-making processes.	NIST Glossary
Threat model	A form of risk assessment that models aspects of the attack and defense sides of a logical entity, such as a piece of data, an application, a host, a system, or an environment.	NIST Glossary
Transmission	An act, process, or instance of transmitting.	*Merriam-Webster*
	The act of sending or conveying data, voice, audio, or video from one person or place to another.	FFIEC Adapted for Supervisory Purposes

Term	Definition	Source
	U	
Unit testing	A testing technique that is used to test program logic in a particular program or module. The purpose of the test is to ensure that the internal operation of the program performs according to specification. It uses a set of test cases that focus on the control structure of the procedural design.	ISACA Glossary
Unstructured data	Data that does not have a predefined data model or is not organized in a predefined way.	NIST SP 1500-1r2, *NIST Big Data Interoperability Framework: Vol. 1, Definitions*
Useful life	The normal expected operating life of an asset.	Internal Revenue Service
User acceptance testing (UAT)	Testing that involves taking use cases or procedures for how the system was designed to perform and ensuring that someone who follows the procedure gets the intended result; however, it does not necessarily demonstrate how well the system supports users in performing those functions.	NIST IR 7741, *NIST Guide to the Processes Approach for Improving the Usability of Electronic Health Records*
	V	
Virtualization	The simulation of the software and/or hardware upon which other software runs.	NIST Glossary
Virtual machine (VM)	Software that allows a single host to run one or more guest operating systems.	NIST Glossary
	A simulated environment created by virtualization using software that allows a single host to run one or more guest operating systems.	FFIEC Adapted for Supervisory Purposes
Vulnerability	Weakness in system security procedures, design, implementation, internal controls, etc., that could be accidentally triggered or intentionally exploited and result in a violation of the system's security policy.	NIST Glossary
Vulnerability assessment	Systematic examination of an information system or product to determine the adequacy of security and privacy measures, identify security and privacy deficiencies, provide data from which to predict the effectiveness of proposed security and privacy measures, and confirm the adequacy of such measures after implementation.	NIST Glossary
Vulnerability management	The practice of identification, classification, remediation, and mitigation of vulnerabilities in systems, subsystems, and system components.	*DoD Instruction 8531.01; DoD Vulnerability Management*
Vulnerability scanning	A technique used to identify hosts/host attributes and associated vulnerabilities.	NIST Glossary
	W	
Workstation	A computer used for tasks such as programming, engineering, and design.	NIST Glossary
	Z	
Zero-day exploit	Referred to as zero-day attack; an attack that exploits a previously unknown hardware, firmware, or software vulnerability.	NIST Glossary
	An exploit that exploits a previously unknown hardware, firmware, or software vulnerability.	FFIEC Adapted for Supervisory Purposes

APPENDIX C: ABBREVIATIONS

AI	artificial intelligence
AICPA	American Institute of Certified Public Accountants
API	application programming interface
ATM	automated teller machine
CA	Consumer Affairs (for FRB SR Letters)
CCB	configuration or change control board
CD	continuous delivery
CFPB	Consumer Financial Protection Bureau
CFR	Code of Federal Regulations
CI	continuous integration
CIO	chief information officer
CISA	Cybersecurity and Infrastructure Security Agency
CISO	chief information security officer
COTS	commercial off-the-shelf
CPU	central processing unit
DevOps	development and operations
DevSecOps	development security operations
DOD	Department of Defense
EOL	end-of-life
EULA	end user license agreement
FDIC	Federal Deposit Insurance Corporation
FFIEC	Federal Financial Institutions Examination Council
Fintech	financial technology
FOSS	free and open-source software
FIL	Financial Institution Letter
FRB	Board of Governors of the Federal Reserve System
FS-ISAC	Financial Services Information Sharing and Analysis Center
FTC	Federal Trade Commission
GLBA	Gramm–Leach–Bliley Act
GSA	U.S. General Services Administration
HVAC	heating, ventilation, and air conditioning
IaaS	infrastructure-as-a-service
IoT	internet of things
IR	Internal Report
ISACA	Information Systems Audit and Control Association
ISAC	information sharing and analysis center
IT	information technology
ITAM	IT asset management
ITL	Information Technology Laboratory
ITRM	IT risk management
FFIEC IT Handbook	FFIEC Information Technology Examination Handbook
MFA	multifactor authentication
MSA	master services agreement
NCUA	National Credit Union Administration

NIST	National Institute of Standards and Technology
NTIA	National Telecommunications and Information Administration
OCC	Office of the Comptroller of the Currency
OEM	original equipment manufacturer
OFAC	Office of Foreign Assets Control
OWASP	Open Worldwide Application Security Project
OS	operating system
PaaS	platform-as-a-service
PIR	post-implementation review
PMI	Project Management Institute
PMO	program management office
PWS	performance work statement
QA	quality assurance
QC	quality control
RAM	random access memory
RFI	request for information
RFID	radio frequency identification
RFP	request for proposal
RFQ	request for quote
SaaS	software-as-a-service
SBOM	software bill of material
SCRM	supply chain risk management
SDLC	system development life cycle
SLA	service-level agreement
SLC	State Liaison Committee
SOC	security operations center
SOC	system and organization control
SOW	statement of work
SP	Special Publication
SQL	Structured Query Language
SR	Supervision and Regulation Letter
UAT	user acceptance testing
USB	universal serial bus
USC	U.S. Code

APPENDIX D: REFERENCES

Laws

11 USC 101–112, “Bankruptcy” and “Office of the Law Revision Counsel: U.S. Code; Title 11”
12 USC 226, “Title X of the Financial Institutions Regulatory and Interest Rate Control Act of 1978”
12 USC 1461–1468, “Home Owners’ Loan Act”
12 USC 1813.3(u), “Institution-Affiliated Party”
12 USC 1841–1852, “Bank Holding Company Act”
12 USC 1861–1867, “Bank Service Company Act”
12 USC 1882, “Bank Protection Act of 1968”
12 USC 5481(14) and (26), 5514, 5515, and 5531, “Dodd–Frank Wall Street Reform and Consumer Protection Act”
15 USC 1681w, “Fair and Accurate Credit Transactions Act”
15 USC 6801–6809 and 6821–6827, “Financial Modernization Act of 1999” (“Gramm–Leach–Bliley Act” [GLBA])
17 USC 1–8, 10–12, “Copyright Law of the United States”
18 USC 1030, “Fraud and Related Activity in Connection With Computers”
29 USC 794d, “Rehabilitation Act of 1973, Section 508”

Consumer Financial Protection Bureau

Regulations

12 CFR 1005, “Electronic Fund Transfers (Regulation E)”
12 CFR 1016, “Privacy of Consumer Financial Information (Regulation P)”
12 CFR 1022, “Fair Credit Reporting Act (Regulation V)”

Guidance

CFPB Compliance Bulletin and Policy Guidance 2016-02, “Service Providers” (October 2016)

Federal Deposit Insurance Corporation

Regulations

12 CFR 304.3(d), “Notification of Performance of Bank Services, Form FDIC 6120/06”
12 CFR 326, subpart A, “Minimum Security Procedures”
12 CFR 332, “Privacy of Consumer Financial Information”
12 CFR 364, appendix A, “Interagency Guidelines Establishing Standards for Safety and Soundness”

12 CFR 364, appendix B, "Interagency Guidelines Establishing Information Security Standards"
12 CFR 364, supplement A to appendix B, "Interagency Guidance on Response Programs for Unauthorized Access to Customer Information and Customer Notice"

Guidance

FIL-29-2023, "Interagency Guidance on Third-Party Relationships: Risk Management" (June 6, 2023)
FIL-10-2023, "Financial Institutions Are Required to Meet Contractual Obligations With Bridge Banks" (March 14, 2023)
FIL-08-2023, "Joint Statement on Liquidity Risks to Banking Organizations Resulting From Crypto-Asset Market Vulnerabilities" (February 23, 2023)
FIL-01-2023, "Joint Statement on Crypto-Asset Risks to Banking Organizations" (January 5, 2023)
FIL-35-2022, "Advisory to FDIC-Insured Institutions Regarding Deposit Insurance and Dealings With Crypto Companies" (July 29, 2022)
FIL-30-2022, "FDIC Updates on Brokered Deposits" (July 15, 2022)
FIL-16-2022, "Notification of Engaging in Crypto-Related Activities" (April 7, 2022)
FIL-12-2022, "Computer-Security Incident Notification Implementation" (March 29, 2022)
FIL-74-2021, "Computer-Security Incident Notification Final Rule" (November 18, 2021)
FIL-59-2021, "Conducting Due Diligence on Financial Technology Companies: A Guide for Community Banks" (August 27, 2021)
FIL-55-2021, "Authentication and Access to Financial Institution Services and Systems" (August 11, 2021)
FIL-27-2021, "Bank Secrecy Act: Agencies Address Model Risk Management for Bank Models and Systems Supporting Bank Secrecy Act/Anti-Money Laundering and Office of Foreign Assets Control Compliance" (April 9, 2021)
FIL-103-2020, "The FDIC Publishes Sound Practices to Strengthen Operational Resilience" (November 2, 2020)
FIL-52-2020, "FFIEC Joint Statement on Risk Management for Cloud Computing Services" (April 30, 2020)
FIL-14-2020, "Interagency Statement on Pandemic Planning" (March 6, 2020)
FIL-19-2019, "Technology Service Provider Contracts" (April 2, 2019)
FIL-16-2018, "FFIEC Issues Joint Statement: Cyber Insurance and Its Potential Role in Risk Management Programs" (April 10, 2018)
FIL-68-2016, "FFIEC Cybersecurity Assessment Tool: Frequently Asked Questions" (October 18, 2016)
FIL-43-2016, "Information Technology Risk Examination (InTREx) Program" (June 30, 2016)
FIL-37-2016, "FFIEC Joint Statement on Cybersecurity of Interbank Messaging and Wholesale Payment Networks" (June 7, 2016)
FIL-28-2015, "Cybersecurity Assessment Tool" (July 2, 2015)
FIL-13-2015, "FFIEC Joint Statements on Destructive Malware and Compromised Credentials" (March 30, 2015)

FIL-49-2014, "Technology Alert: GNU Bourne-Again Shell (Bash) Vulnerability" (September 29, 2014)
FIL-41-2014, "FDIC Clarifying Supervisory Approach to Institutions Establishing Account Relationships with Third-Party Payment Processors" (July 28, 2014)
FIL-16-2014, "Technology Alert: OpenSSL "Heartbleed" Vulnerability" (April 11, 2014)
FIL-13-2014, "Technology Outsourcing: Informational Tools for Community Bankers" (April 7, 2014)
FIL-3-2012, "Payment Processor Relationships Revised Guidance (Revised July 2014)" (January 31, 2012)
FIL-4-2009, "Risk Management of Remote Deposit Capture" (January 14, 2009)
FIL-127-2008, "Guidance on Payment Processor Relationships" (November 7, 2008)
FIL-44-2008, "Third-Party Risk: Guidance for Managing Third-Party Risk" (June 6, 2008)
FIL-77-2006, "Authentication in an Internet Banking Environment: Frequently Asked Questions" (August 21, 2006)
FIL-52-2006, "Foreign-Based Third-Party Service Providers: Guidance on Managing Risks in These Outsourcing Relationships" (June 21, 2006)
FIL-69-2005, "Voice Over Internet Protocol: Guidance on the Security Risks of VoIP" (July 27, 2005)
FIL-66-2005, "Spyware: Guidance on Mitigating Risks From Spyware" (July 22, 2005)
FIL-121-2004, "Computer Software Due Diligence Guidance on Developing an Effective Computer Software Evaluation Program to Assure Quality and Regulatory Compliance" (November 16, 2004)
FIL-114-2004, "Risk Management of Free and Open Source Software" (October 21, 2004)
FIL-84-2004, "Guidance on Instant Messaging" (July 21, 2004)
FIL-27-2004, "Guidance on Safeguarding Customers Against E-Mail and Internet-Related Fraudulent Schemes" (March 12, 2004)
FIL-43-2003, "Guidance on Developing an Effective Software Patch Management Program" (May 29, 2003)
FIL-30-2003, "Federal Bank and Credit Union Regulatory Agencies Jointly Issue Guidance on the Risks Associated With Weblinking" (April 23, 2003)
FIL-8-2002, "Guidance on Managing Risks Associated With Wireless Networks and Customer Access" (February 1, 2002)
FIL-50-2001, "Bank Technology Bulletin on Outsourcing" (June 4, 2001)
FIL-81-2000, "FFIEC Guidance on Managing Risks Associated With Outsourcing Technology Services" (November 29, 2000)
FIL-49-1999, "Bank Service Company Act" (June 3, 1999)
FIL-12-99, "FFIEC Adopts Updated Uniform Rating System for Information Technology" (February 5, 1999)
FIL-82-96, "Interagency Statement on the Risks to Financial Institutions Involving Client/Server Computer Systems" (October 8, 1996)
FIL-46-95, "Minimum Security Devices and Procedures" (July 7, 1995)

Note: While listed FIL announcements may be noted as inactive, the underlying guidance remains in effect, unless specifically superseded or rescinded.

Board of Governors of the Federal Reserve System

Regulations

Regulation H

12 CFR 208, appendix D-1, "Interagency Guidelines Establishing Standards for Safety and Soundness"
12 CFR 208, appendix D-2, "Interagency Guidelines Establishing Information Security Standards"
12 CFR 208.61, "Bank Security Procedures"

Regulation K

12 CFR 211.5 and 211.24 (i), "Protection of Customer and Consumer Information"

Regulation Y

12 CFR 225, appendix F, "Interagency Guidelines Establishing Information Security Standards"

Guidance

SR Letter 23-4, "Interagency Guidance on Third-Party Relationships: Risk Management" (June 7, 2023)
SR Letter 22-4/CA 22-3, "Contact Information in Relation to Computer-Security Incident Notification Requirements" (March 29, 2022)
SR Letter 21-14, "Authentication and Access to Financial Institution Services and Systems" (August 11, 2021)
SR Letter 20-24, "Interagency Paper on Sound Practices to Strengthen Operational Resilience" (November 2, 2020)
SR Letter 20-3/CA 20-2, "Interagency Statement on Pandemic Planning" (March 10, 2020)
SR Letter 16-11, "Supervisory Guidance for Assessing Risk Management at Supervised Institutions With Total Consolidated Assets Less than $100 Billion" (June 8, 2016, revised February 17, 2021)
SR Letter 15-9, "FFIEC Cybersecurity Assessment Tool for Chief Executive Officers and Boards of Directors" (July 2, 2015)
SR Letter 12-17/CA 12-14, "Consolidated Supervision Framework for Large Financial Institutions" (December 17, 2012)
SR Letter 11-7, "Guidance on Model Risk Management" (April 4, 2011)
SR Letter 05-23/CA 05-10, "Interagency Guidance on Response Programs for Unauthorized Access to Customer Information and Customer Notice" (December 1, 2005)
SR Letter 04-17, "FFIEC Guidance on the Use of Free and Open Source Software" (December 6, 2004)
SR Letter 01-15, "Standards for Safeguarding Customer Information" (May 31, 2001)
SR Letter 99-8, "Uniform Rating System for Information Technology" (March 31, 1999)
SR Letter 98-9, "Assessment of Information Technology in the Risk-Focused Frameworks for the Supervision of Community Banks and Large Complex Banking Organizations" (April 20, 1998, revised February 26, 2021)

SR Letter 96-14, "Risk-Focused Safety and Soundness Examinations and Inspections" (May 24, 1996)

National Credit Union Administration

Regulations

12 CFR 748, "Security Program, Suspicious Transactions, Catastrophic Acts, Cyber Incidents, and Bank Secrecy Act Compliance"
12 CFR 748, appendix A, "Guidelines for Safeguarding Member Information"
12 CFR 749, "Records Preservation Program and Appendices—Record Retention Guidelines; Catastrophic Act Preparedness Guidelines"
12 CFR 749, appendix A, "Record Retention Guidelines"

Guidance

Letter to Credit Unions 22-CU-07, "Federally Insured Credit Union Use of Distributed Ledger Technologies" (May 2022)
Letter to Credit Unions 07-CU-13, "Evaluating Third Party Relationships" (December 2007)
Letter to Credit Unions 06-CU-07, "IT Security Compliance Guide for Credit Unions" (April 2006)
Letter to Credit Unions 03-CU-14, "Computer Software Patch Management" (September 2003)
Letter to Credit Unions 01-CU-20, "Due Diligence Over Third Party Service Providers" (November 2001)
Letter to Credit Unions 00-CU-11, "Risk Management of Outsourced Technology Services" (December 2000)

Office of the Comptroller of the Currency

Regulations

12 CFR 5.30, "Establishment, Acquisition, and Relocation of a Branch of a National Bank"
12 CFR 5.31, "Establishment, Acquisition, and Relocation of a Branch and Establishment of an Agency Office of a Federal Savings Association"
12 CFR 30, appendix A, "Interagency Guidelines Establishing Standards for Safety and Soundness"
12 CFR 30, appendix B, "Interagency Guidelines Establishing Information Security Standards"
12 CFR 30, appendix D, "OCC Guidelines Establishing Heightened Standards for Certain Large Insured National Banks, Insured Federal Savings Associations, and Insured Federal Branches"
12 CFR 30, appendix E, "OCC Guidelines Establishing Standards for Recovery Planning by Certain Large Insured National Banks, Insured Federal Savings Associations, and Insured Federal Branches"
12 CFR 41.83, "Proper Disposal of Records Containing Consumer Information"

Guidance

OCC Bulletin 2023-22, "Cybersecurity: Cybersecurity Supervision Work Program" (June 26, 2023)

OCC Bulletin 2023-17, "Third-Party Relationships: Interagency Guidance on Risk Management" (June 6, 2023)

OCC Bulletin 2023-1, "Crypto-Assets: Joint Statement on Crypto-Asset Risks to Banking Organizations" (January 3, 2023)

OCC Bulletin 2022-22, "Cybersecurity: 2022 Cybersecurity Resource Guide for Financial Institutions" (October 6, 2022)

OCC Bulletin 2022-8, "Information Technology: OCC Points of Contact for Banks' Computer-Security Incident Notifications" (March 29, 2022)

OCC Bulletin 2021-55, "Computer-Security Incident Notification: Final Rule" (November 23, 2021)

OCC Bulletin 2021-40, "Third-Party Relationships: Conducting Due Diligence on Financial Technology Companies: A Guide for Community Banks" (August 27, 2021)

OCC Bulletin 2021-36, "Information Security: FFIEC Statement on Authentication and Access to Financial Institution Services and Systems" (August 11, 2021)

OCC Bulletin 2021-30, "FFIEC Information Technology Examination Handbook: New Architecture, Infrastructure, and Operations Booklet" (June 30, 2021)

OCC Bulletin 2021-17, "Artificial Intelligence: Request for Information on Financial Institutions' Use of Artificial Intelligence, Including Machine Learning" (March 31, 2021)

OCC Bulletin 2020-94, "Operational Risk: Sound Practices to Strengthen Operational Resilience" (October 30, 2020)

OCC Bulletin 2020-46, "Cybersecurity: Joint Statement on Security in a Cloud Computing Environment" (April 30, 2020)

OCC Bulletin 2020-23, "Pandemic Planning: Essential Critical Infrastructure Workers in the Financial Services Sector" (March 25, 2020)

OCC Bulletin 2020-13, "Pandemic Planning: Updated FFIEC Guidance" (March 6, 2020)

OCC Bulletin 2020-5, "Cybersecurity: Joint Statement on Heightened Cybersecurity Risk" (January 16, 2020)

OCC Bulletin 2019-57, "FFIEC Information Technology Examination Handbook: Revised Business Continuity Management Booklet" (November 14, 2019)

OCC Bulletin 2019-37, "Operational Risk: Fraud Risk Management Principles" (July 24, 2019)

OCC Bulletin 2019-13, "Recovery Planning: Updated Comptroller's Handbook Booklet and Rescissions" (March 15, 2019)

OCC Bulletin 2018-40, "Cybersecurity: Cyber-Related Sanctions" (November 5, 2018)

OCC Bulletin 2017-43, "New, Modified, or Expanded Bank Products and Services: Risk Management Principles" (October 20, 2017)

OCC Bulletin 2017-7, "Third-Party Relationships: Supplemental Examination Procedures" (January 24, 2017)

OCC Bulletin 2016-34, "Cybersecurity: Frequently Asked Questions on the FFIEC Cybersecurity Assessment Tool" (October 17, 2916)

OCC Bulletin 2016-18, "Cybersecurity of Interbank Messaging and Wholesale Payment Networks: FFIEC Statement" (June 7, 2016)
OCC Bulletin 2015-44, "FFIEC Information Technology Examination Handbook: Revised Management Booklet" (November 10, 2015)
OCC Bulletin 2015-40, "Cybersecurity: Joint Statement on Cyber Attacks Involving Extortion" (November 3, 2015)
OCC Bulletin 2015-31, "Cybersecurity: FFIEC Cybersecurity Assessment Tool" (June 30, 2015)
OCC Bulletin 2015-20, "Cybersecurity: Destructive Malware Joint Statement" (March 30, 2015)
OCC Bulletin 2015-19, "Cybersecurity: Cyber Attacks Compromising Credentials Joint Statement" (March 30, 2015)
OCC Bulletin 2014-48, "Bourne-Again Shell (Bash) 'Shellshock' Vulnerability: FFIEC Alert" (September 26, 2014)
OCC Bulletin 2014-17, "Information Security Vulnerability in OpenSSL Encryption Tool (Heartbleed): Joint Statement" (April 25, 2014)
OCC Bulletin 2014-13, "Cyber Attacks on Financial Institutions' Automated Teller Machine and Card Authorization Systems: Joint Statement" (April 2, 2014)
OCC Bulletin 2011-12, "Sound Practices for Model Risk Management: Supervisory Guidance on Model Risk Management" (April 4, 2011)
OCC Bulletin 2008-16, "Information Security: Application Security" (May 8, 2008)
OCC Bulletin 2006-35, "Authentication in an Internet Banking Environment: Frequently Asked Questions" (August 15, 2006)
OCC Bulletin 2006-31, "FFIEC Information Security Booklet: Information Security Guidance" (July 27, 2006)
OCC Bulletin 2005-13, "Response Programs for Unauthorized Access to Customer Information and Customer Notice - Final Guidance: Interagency Guidance" (April 14, 2005)
OCC Bulletin 2004-47, "FFIEC Guidance: Risk Management for the Use of Free and Open Source Software" (October 27, 2004)
OCC Bulletin 2004-32, "FFIEC Information Technology Examination Handbook: FFIEC IT Booklets on Outsourcing Technology Services and Management" (July 15, 2004)
OCC Bulletin 2002-16, "Bank Use of Foreign-Based Third-Party Service Providers: Risk Management Guidance" (May 15, 2002)
OCC Bulletin 2002-10, "Country Risk: Sound Risk Management Practices" (March 11, 2002)
OCC Bulletin 2001-35, "Examination Procedures to Evaluate Compliance With the Guidelines to Safeguard Customer Information: Examination Procedures" (July 18, 2001)
OCC Bulletin 2001-12, "Bank-Provided Account Aggregation Services: Guidance to Banks" (February 28, 2001)
OCC Bulletin 1998-31, "Guidance on Electronic Financial Services and Consumer Compliance: FFIEC Guidance" (July 30, 1998)
OCC Bulletin 1998-3, "Technology Risk Management: Guidance for Bankers and Examiners" (February 4, 1998)

Other References

Federal Regulations

16 CFR 314 (FTC), "Standards for Safeguarding Customer Information"
16 CFR 314.4(j) (FTC), "Amendment to Standards for Safeguarding Customer Information"

National Institute of Standards and Technology (NIST)

NIST, "Automated Identification Technologies for Forensic Science"
NIST Glossary
National Checklist Program (security configurations)
National Vulnerability Database
NIST Advanced Manufacturing Series 100-40, *Agile for Model-Based-Standards Development*
NIST SP 500-292, *NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology*
NIST SP 500-316, *Framework for Cloud Usability*
NIST SP 800-28, version 2, *Guidelines on Active Content and Mobile Code*
NIST SP 800-37, rev. 2, *Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy*
NIST SP 800-47, rev. 1, *Managing the Security of Information Exchanges*
NIST SP 800-53, rev. 5, *Security and Privacy Controls for Information Systems and Organizations*
NIST SP 800-53, rev. 5, SR-3, *Supply Chain Controls and Processes*
NIST SP 800-58, *Security Considerations for Voice Over IP Systems*
NIST SP 800-63-3, *Digital Identity Guidelines*
NIST SP 800-82, rev. 3, *Guide to Operational Technology (OT) Security*
NIST 800-98, *Guidelines for Securing Radio Frequency Identification (RFID) Systems*
NIST SP 800-115, *Technical Guide to Information Security Testing and Assessment*
NIST SP 800-125, *Guide to Security for Full Virtualization Technologies*
NIST SP 800-128, *Guide for Security-Focused Configuration Management of Information Systems*
NIST SP 800-137, *Information Security Continuous Monitoring (ISCM) for Federal Systems and Organizations*
NIST SP 800-145, *The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology*
NIST SP 800-146, *Cloud Computing Synopsis and Recommendations*
NIST SP 800-160, vol. 1, rev. 1, *Engineering Trustworthy Secure Systems*
NIST SP 800-160, vol. 2, rev. 1, *Developing Cyber Resilient Systems: A Systems Security Engineering Approach*
NIST SP 800-161r1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*
NIST SP 800-163, rev. 1, *Vetting the Security of Mobile Applications*
NIST SP 800-190, *Application Container Security Guide*
NIST SP 800-193, *Platform Firmware Resiliency Guidelines*

NIST SP 800-204, *Security Strategies for Microservices-Based Application Systems*
NIST SP 800-204A, *Building Secure Microservices-Based Applications Using Service-Mesh Architecture*
NIST SP 800-204C, *Implementation of DevSecOps for a Microservices-Based Application With Service Mesh*
NIST SP 800-207, *Zero Trust Architecture*
NIST SP 800-210, *General Access Control Guidance for Cloud Systems*
NIST SP 800-218, *Secure Software Development Framework (SSDF) Version 1: Recommendations for Mitigating the Risks of Software Vulnerabilities*
NIST SP 1500-1r2, *NIST Big Data Interoperability Framework: Volume 1, Definitions*
NIST SP 1800-5, *IT Asset Management*
NIST SP 1800-16, *Securing Web Transactions: TLS Server Certificate Management*.
NIST Response to Executive Order 14028, *Software Security in Supply Chains*
NIST Information Technology Laboratory (ITL) Bulletin, "Security Considerations for Exchanging Files Over the Internet" (August 2020)
NIST ITL Bulletin, "Security for Enterprise Telework, Remote Access, and Bring Your Own Device (BYOD) Solutions"(March 2020)
NIST ITL Bulletin, "The System Development Life Cycle (SDLC)" (April 2009)
NIST National Construction Safety Team Act Report (NCSTAR) 1-4B, *Fire Suppression Systems*
NIST Directive S 6106.01, "Open Source Code"
NIST Internal Report (IR) 7499, *Guidelines for Planning and Development of Software for Buildings and Building Systems*
NIST IR 7622, *Notional Supply Chain Risk Management for Federal Information Systems*
NIST IR 7741, *NIST Guide to the Processes Approach for Improving the Usability of Electronic Health Records*
NIST IR 8228, *Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks*
NIST IR 8259A, *IoT Device Cybersecurity Capability Core Baseline*
NIST IR 8397, *Guidelines on Minimum Standards for Developer Verification of Software*
NIST IR 8419, *Block Chain and Related Technologies to Support Manufacturing Supply Chain Traceability: Needs and Industry Perspectives*
NIST IR 8425, *Profile of the IoT Core Baseline for Consumer IoT Products*

Documents referenced by NIST

Gallagher, Leonard, and Jeff Offutt, "Test Sequence Generation for Integration Testing of Component Software," *Computer Journal*, 2005
Martzloff, François, *A New IEC Standard on the Measurement of Power Quality Parameters*
Rapp, Paul, *Copyright Law*

Other Federal Government

Bureau of Labor Statistics

Occupational Outlook Handbook

"Customer Service Representatives"

"Purchasing Managers, Buyers, and Purchasing Agents"

"Software Developers, Quality Assurance Analysts, and Testers"

Centers for Disease Control

"Implementation Phase"

Cybersecurity and Infrastructure Security Agency (CISA)

"CISA Insights - Cyber: Remediate Vulnerabilities for Internet Accessible Systems"

CRR Supplemental Resource Guide, Volume 3: Configuration and Change Management, Version 1.1

CRR Supplemental Resource Guide, Volume 5: Incident Management Resource Guide, Version 1.1

"Infrastructure Dependency Primer"

"Protecting Against Malicious Code"

Securing the Software Supply Chain: Recommended Practices Guide for Developers

Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Security-by-Design and -Default

Department of Defense (DOD)

DOD Enterprise DevSecOps Fundamentals

DoD Enterprise DevSecOps Reference Design: Version 1.0

Securing Defense-Critical Supply Chains

"DoD Open Source Software FAQ"

DOD Instruction 8531.01: DoD Vulnerability Management

Documents referenced by DOD

Davis, Noopur, *Secure Software Development Life Cycle Processes: A Technology Scouting Report*, *Carnegie Mellon University Software Engineering Institute* (Defense Technical Information Center)

Food and Drug Administration

Glossary of Computer System Software Development Terminology

National Aeronautics and Space Administration

A Software Development Simulation Model of a Spiral Process (NASA Technical Reports Server)

National Telecommunications and Information Administration (NTIA)

Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM)

"SBOM at a Glance"

"SBOM Myths vs. Facts"

U.S. Department of Commerce

"Open Source Code"

U.S. Copyright Office

Copyright Law of the United States (Title 17)

"Copyright in General"

"What Is Copyright?"

U.S. Department of Energy

Suspect/Counterfeit Items Awareness Training

U.S. Department of Homeland Security

Access Control Technologies Handbook

DHS Lexicon Terms and Definitions

Privacy Impact Assessment for the National Cybersecurity Protection System (NCPS) – Intrusion Detection

Systems Engineering Life Cycle

U.S. Department of Justice

Systems Development Life Cycle Guidance, chapter 12, "Disposition Phase"

U.S. Government Accountability Office

Information Technology Investment Management

The Post-Implementation Review

U.S. General Services Administration (GSA)

"Performance Work Statement"

"PWS, SOO, SOW - Finding the Best Fit"

"Respond to a Solicitation"

"RFP, RFI, and RFQ: Understanding the Difference"

U.S. Patent and Trademark Office

"Copyright Basics"

"Trademark, Patent, or Copyright"

State Government

New York State Office of Information Technology Services

NYS Project Management Guidebook Release 2

System Implementation

Industry References

American Institute of Certified Public Accountants (AICPA)

System and Organization Controls: SOC Suite of Services

"What Are You Doing to Prevent Cyberattacks? SOC 2 and SOC for Cybersecurity: How They're Different and How They Can Help"

Center for Internet Security

"CIS Critical Security Control 7: Continuous Vulnerability Management"

Cloud Security Alliance

"Understanding the OWASP API Security Top 10"

Creative Commons

CC0

Financial Accounting Standards Board

"Post-Implementation Review"

Financial Services Information Sharing and Analysis Center (FS-ISAC)

Information Systems Audit and Control Association (ISACA)

ISACA Glossary

International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)

ISO Online Browsing Platform, Terms & Definitions

Internet Engineering Task Force (IETF)

"OAuth 2.0 Dynamic Client Registration Protocol"

The MITRE Corporation (MITRE)

Improperly Controlled Modification of Dynamically-Determined Object Attributes

The National Automated Clearinghouse Association (Nacha)

National Council of ISACs

Open Worldwide Application Security Project (OWASP)

"OWASP API Security Project"

Component Analysis

Fuzzing

OWASP Vulnerability Management Guide (OVMG)

Code Review Guide 2.0

Payments Card Industry (PCI) Security Standards Council

Project Management Institute (PMI)

A Guide to the Project Management Body of Knowledge, fifth edition

"Running IS Maintenance as a Project"

"Statement of Work: The Foundation for Delivering Successful Service Projects"

"What Is Project Management?"

"Is This Really Worth the Effort? The Need for a Business Case"

Society for Human Resource Management

"Top Database Security Threats and How to Mitigate Them"

Other References

Black's Law Dictionary

Law Insider Dictionary

"IT Security Health Check"