**NISTIR 8011**
**Volume 4**

# Automation Support for Security Control Assessments:

*Software Vulnerability Management*

Kelley Dempsey
Paul Eavy
George Moore
Eduardo Takamura

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8011-4

**National Institute of Standards and Technology**
U.S. Department of Commerce

**NISTIR 8011**
**Volume 4**

# Automation Support for Security Control Assessments:

*Software Vulnerability Management*

Kelley Dempsey
Eduardo Takamura
*Computer Security Division*
*Information Technology Laboratory*

Paul Eavy
*Cybersecurity Division*
*Cybersecurity and Infrastructure Security Agency*
*Department of Homeland Security*

George Moore

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8011-4

April 2020

U.S. Department of Commerce
*Wilbur L. Ross, Jr., Secretary*

National Institute of Standards and Technology
*Walter Copan, NIST Director and Under Secretary of Commerce for Standards and Technology*

National Institute of Standards and Technology Interagency or Internal Report 8011 Volume 4
112 pages (April 2020)

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8011-4

**Comments on this publication may be submitted to:**

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: sec-cert@nist.gov

## Reports on Computer Systems Technology

## Abstract

The NISTIR 8011 capability-specific volumes focus on the automation of security control assessment within each individual information security capability. The capability-specific volumes add tangible detail to the more general overview given in NISTIR 8011 Volume 1, providing a template for transition to a detailed, NIST standards-compliant automated assessment. This document, Volume 4 of NISTIR 8011, addresses automating the assessment of security controls that support the software vulnerability management security capability to facilitate the management of risk created by defects present in software on the network. Software vulnerability management, in the scope of this document, focuses on known defects that have been discovered in software in use on a system. The Common Weakness Enumeration (CWE) provides identifiers for weaknesses that result from poor coding practices and have the potential to result in software vulnerabilities. The Common Vulnerabilities and Exposures (CVEs) program provides a list of many known vulnerabilities. Together, CVE and CWE are used to identify software defects and the weaknesses that cause a given defect. Vulnerable software is a key target that attackers use to initiate an attack internally and to expand control.

## Keywords

actual state; assessment; authorization boundary; automation; capability; Common Vulnerability and Exposure (CVE); Common Weakness Enumeration (CWE); dashboard; defect; desired state specification; dynamic code analyzer; Information Security Continuous Monitoring (ISCM); malicious code; malware; mitigation; ongoing assessment; patch management; root cause analysis; security capability; security control item; security control; software file; Software Identification (SWID) tag; software injection; software product; software vulnerability; software weakness; software; static code analyzer

## Acknowledgments

The authors—Kelley Dempsey and Eduardo Takamura of the National Institute of Standards and Technology (NIST), Paul Eavy of the Department of Homeland Security, and Dr. George Moore—wish to thank their colleagues who reviewed drafts and assisted in the production of this document, including Nathan Aileo, Ujwala Arikatla, Jim Foti, Ned Goren, John Groenveld, Frank Husson, Jaime Miller, Susan Pagan, Ron Ross, David Waltermire, Kimberly Watson, Isabel Van Wyk, Brett Kreider, Nate Lee, Jim Finegan, and Nolda Dooley. The authors also gratefully acknowledge and appreciate the comments and contributions made by government agencies, private organizations, and individuals in providing direction and assistance in the development of this document.

## Document Conventions

The terms "shall" and "shall not" indicate requirements to be followed strictly in order to conform to the publication and from which no deviation is permitted.

The terms "should" and "should not" indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is discouraged but not prohibited.

The terms "may" and "need not" indicate a course of action permissible within the limits of the publication.

The terms "can" and "cannot" indicate a possibility and capability, whether material, physical or causal.

The security capability designation "VULN" was utilized in previous volumes of NISTIR 8011 in reference to the Software Vulnerability Management security capability. For greater consistency with industry, academia, and other federal agencies, the more widely used designation for the Software Vulnerability Management security capability, "VUL," replaces VULN herein and in future NISTIR 8011 volumes when referring to the Software Vulnerability Management security capability.

## Patent Disclosure Notice

## Executive Summary

The National Institute of Standards and Technology (NIST) and the Department of Homeland Security (DHS) have collaborated on the development of a process that automates the test assessment method described in NIST Special Publication (SP) 800-53A for the security controls catalogued in NIST SP 800-53. This process is consistent with the Risk Management Framework described in NIST SP 800-37 and the Information Security Continuous Monitoring (ISCM) guidance in NIST SP 800-137. The multi-volume NIST Interagency Report 8011 (NISTIR 8011) has been developed to provide information on automation support for ongoing assessments. NISTIR 8011 describes how ISCM facilitates automated, ongoing assessment to provide near-real-time security-related information to organizational officials on the security posture of individual systems and the organization as a whole.

NISTIR 8011, Volume 1 includes a description of *ISCM Security Capabilities*—groups of security controls working together to achieve a common purpose. The subsequent NISTIR 8011 volumes are capability-specific. Each volume focuses on one specific ISCM information security capability in order to (a) add tangible detail to the more general overview given in NISTIR 8011 Volume 1 and (b) provide a template for the transition to detailed, standards-compliant automated assessments. NISTIR 8011, Volume 4 assumes the reader is familiar with the concepts and ideas presented in the Overview (NISTIR 8011, Volume 1) as well as concepts and terms from the NIST Risk Management Framework (RMF) [SP800-37]. Many terms used herein are defined in NIST SP 800-37 or in the NISTIR 8011 Volume 1 glossary.

This publication, Volume 4 of NISTIR 8011, immediately follows Hardware Asset Management (HWAM) and Software Asset Management (SWAM) capabilities published as Volumes 2 and 3 respectively. Although it was planned to be released as Volume 5, following Configuration Settings Management (CSM), the VUL capability is being addressed as Volume 4 in the NISTIR 8011 series due to its close relationship with software. Its objective is to address the management of risk created by defects present in software on the network. A *software vulnerability* is caused by one or more known defects that have been discovered in software, and that can be exploited to affect an adverse security or privacy outcome.[1] *Vulnerable software* is software in use on a system that has a software vulnerability but has not yet been patched or otherwise mitigated. The Common Weakness Enumeration (CWE) provides identifiers for weaknesses that result from poor design or coding practices and *have the potential* to result in software vulnerabilities. The Common Vulnerabilities and Exposures (CVEs) program works with software providers, vulnerability coordinators, bug bounty programs, and vulnerability researchers to provide a list of publicly disclosed vulnerabilities. Together, CVE and CWE are used to identify software defects and the weaknesses that caused a given defect respectively. Vulnerable software is a key target that attackers use to initiate an attack and to expand control within a system. Patching vulnerabilities discovered in existing software and improving coding practices for future releases of software are two ways to limit the success of attacks.

[1] *Hardware* vulnerabilities are often mitigated through software, such as applying a firmware or operating system patch that controls hardware access or turns off a hardware feature. In NISTIR 8011, software vulnerabilities are inclusive of hardware vulnerabilities mitigated through software. Hardware vulnerabilities that cannot be mitigated through software require physical changes to or replacement of hardware, and are outside the scope of NISTIR 8011, Volume 4.

The term *vulnerability* is used herein to denote *software* vulnerability as opposed to the more general use of the term *vulnerability*. See glossary for the distinction.

Known vulnerabilities are the most likely flaws to be exploited. Risk from **known** vulnerabilities is reduced by implementing the software vulnerability management (VUL) capability. The VUL capability focuses on managing known vulnerabilities *and* common sources of software flaws known to produce vulnerabilities.

When known software vulnerabilities are unmanaged, uncorrected, or undetected, software is left open to exploitation. As a result, vulnerable software is a key target that attackers use to initiate an attack on an organization's network and expand control to attack other components on that network. A well-designed vulnerability management capability helps prevent software with vulnerabilities from being installed on a network, detect software with vulnerabilities already installed on a network, and respond to the vulnerabilities detected (e.g., by patching the vulnerabilities or through other mitigations). Automated assessment of known software vulnerabilities and weaknesses helps verify that the software vulnerability management capability is working. When known vulnerabilities are managed, the level of effort needed to initiate an attack and expand control to other components on the network is increased since the attacker must identify another method of attack.

Risk from **unknown** vulnerabilities is reduced primarily by implementing the software asset management capability (whitelisting) [IR8011-3], and by limiting the use of software to an organizationally approved list. When software whitelisting is effective, unauthorized software is blocked, thereby limiting vulnerabilities to only those remaining in the organization's *authorized* software. Additionally, the VUL capability can potentially address *unknown* vulnerabilities by scanning for poor coding practices (e.g., CWEs). Scanning for CWEs can make unknown vulnerabilities known so they can be addressed. Thus, while the primary focus of the VUL capability is on known vulnerabilities (e.g., CVEs), risk associated with unknown vulnerabilities is also addressed by the focus on common sources of software flaws (e.g., CWEs).

NISTIR 8011, Volume 4 outlines detailed, step-by-step processes to automate the assessment of security controls that support vulnerability management implemented for a given assessment boundary (target network) and to apply the results to the assessment of all authorization boundaries within that network. A process is also provided to implement the assessment (diagnosis) and response to a discovered vulnerability. Automated testing related to the controls for the VUL capability, as outlined herein, is consistent with other NIST guidance.

NISTIR 8011, Volume 4 documents a detailed assessment plan to evaluate the effectiveness of controls related to vulnerability management. Included are specific tests that form the basis for such a plan, how the tests apply to specific controls, and the resources needed to operate and use the assessment to mitigate defects found. For the VUL capability, it can be shown that the assessment of 87.5 %[2] of determination statements for controls in the NIST SP 800-53 Low-

[2] Derived from the Control Allocation Tables (CAT) in this volume. With respect to security controls selected in the NIST SP 800-53 [SP800-53] Low-Medium-High baselines that support the VUL capability, 42 of 48 determination statements (87.5 %) can be fully or partially automated.

Medium-High baselines can be fully or partially automated.

The methods outlined here are designed to provide objective, timely, and complete identification of defects in the effectiveness of security controls supporting the VUL capability, facilitating risk management at a lower cost than manual assessment methods. Using security control defect information can drive the most efficient and effective responses to the security defects found.

## Table of Contents

## List of Appendices

## List of Figures

## List of Tables

# 1 Introduction

## 1.1 Purpose and Scope

The purpose of the National Institute of Standards and Technology (NIST) Interagency Report (NISTIR) 8011, Volume 4 is to provide an operational approach for automating the assessment of NIST SP 800-53 [SP800-53] security controls related to the Information Security Continuous Monitoring (ISCM) security capability of software vulnerability management (VUL). The VUL capability is consistent with the principles outlined in NISTIR 8011, Volume 1 [IR8011-1].

The scope of this report is limited to the assessment of security controls/control items that are implemented for managing software security vulnerabilities and coding weaknesses, also referred to as *flaws*, as defined in NIST SP 800-53.

## 1.2 Target Audience

Because it is focused on the VUL capability, NISTIR 8011, Volume 4 is of special relevance to those who authorize, download, install, and/or execute software—particularly software patches. In addition, NISTIR 8011, Volume 4 is relevant to those who design, code, and test software, and those who wish to understand the risks that software might impose on non-software assets.

## 1.3 Organization of this Volume

Section 2 provides an overview of the VUL capability to clarify both scope and purpose and provides links to additional information specific to the VUL capability. Section 3 provides detailed information on the VUL defect checks and how the defect checks are used to automate assessment of the effectiveness of NIST SP 800-53 security controls and control items that support the VUL capability. Section 3 also provides artifacts that can be used by an organization to produce an automated security control assessment plan for most of the control items supporting software vulnerability management.

## 1.4 Interaction with Other Volumes in this NISTIR

Volume 1 of this NISTIR (Overview) provides a conceptual synopsis of using automation to support security control assessment as well as definitions and background information that facilitate understanding of the information in this and subsequent volumes.[3] NISTIR 8011, Volume 4 assumes that the reader is familiar with the information in Volume 1 as well as concepts and terms from the NIST Risk Management Framework [SP800-37].

The VUL capability detects vulnerable software that has been loaded on or is being executed within the target network, and responds in accordance with organizational policy. Identifying vulnerable software allows vulnerabilities to be mitigated. The VUL capability depends on the

[3] NISTIR 8011 Volume 1 (June 2017) lists the VUL capability to be published as NISTIR 8011, Volume 5. However, it was later determined that the listed order for publishing NISTIR 8011 capability volumes was not optimal, so the VUL capability is published as NISTIR 8011, Volume 4. The listed order for publishing NISTIR 8011 capability volumes will be revised in an errata version of Volume 1.

Software Asset Management (SWAM) capability [IR8011-3] to provide an inventory of installed software. The inventory is then examined to detect the presence of known vulnerabilities and poor coding practices. Changing configuration settings (the subject of the Configuration Setting Management (CSM) capability in a future NISTIR 8011 volume) can sometimes be used to mitigate vulnerabilities by disabling or otherwise protecting vulnerable software features, especially when patches are not available, thereby supporting software vulnerability management.

In practice, vulnerability scanning software is often used to find vulnerable software. If the metadata used to guide software scanning is organized appropriately[4], the same digital fingerprints used for whitelisting [IR8011-3] can be used to accurately and reliably identify vulnerable code as further discussed in Section 2.5.2.3.

[4] Organized appropriately means that it is known which software code files (identified by their digital fingerprint) are vulnerable, and that software whitelisting is used to determine whether or not files with vulnerabilities are present.

# 2 Software Vulnerability Management (VUL) Capability Definition, Overview, and Scope

Software vulnerability management recognizes that even authorized software—software that has been assessed and approved by the organization for execution on a system—can have known vulnerabilities and (presumably) unknown instances of coding weaknesses that result in vulnerabilities. Networked devices with coding defects in authorized software are also exploitable. A key attack vector for external and internal attackers is to exploit software defects, either for what the software itself can offer or as a platform from which to attack other assets. Attacks can make use of previously unknown software vulnerabilities (often referred to as zero-day vulnerabilities), although attacks against known vulnerabilities are more likely to be attempted. By assigning software with flaws to a person or team for response, the VUL capability helps reduce the probability that attackers find and exploit software weaknesses and vulnerabilities.

## 2.1 Find Defects/Prioritize Response

The VUL capability gives an organization visibility into the vulnerabilities in software authorized to operate—or being considered for authorization to operate – on its network(s)[5]. Visibility into the vulnerabilities allows the organization to manage and defend itself in an appropriate manner. The VUL capability also provides a view of software management responsibility that helps prioritize identified defects and facilitate risk response decisions (e.g., mitigation or acceptance) by the assigned managers.

The VUL capability identifies software that is present on the network (the *actual* state) and compares it with the *desired* state software inventory to determine if there are less vulnerable (usually newer) versions of software that can be deployed or if non-patch-related mitigation strategies are needed.[6] The VUL capability is focused on ensuring that all software operating on the target network poses as little risk from known vulnerabilities as possible, and that an effective patching and response policy[7] is applied.

Note that Volume 3 of NISTIR 8011 defines software to include firmware. The same definition is used in this volume.

Software (code), as used here, includes a range of assets that might not always be thought of as

[5] Specific software products are authorized to operate as components of a system that may or may not be connected to organizational networks. Conversely, to be effective and efficient, automated assessment of systems and system components requires network connectivity. Standalone systems/system components do not lend themselves to automated assessment. See NISTIR 8011 Volume 1 for information on authorization boundaries (systems) versus automated assessment boundaries (networks).

[6] See sections 2.5.1 and 2.5.2 for discussion on actual and desired states.

[7] Patching and response policy may be addressed in the organization's vulnerability management policy.

software. Such software assets include:

- Installed software files and products listed in the operating system software database (e.g., Windows Registry, Linux package manager);
- Software files and products residing on a hard drive but not listed in the operating system database;
- Mobile code;
- Firmware, if it can be modified (usually includes the BIOS); and
- Code in memory (which could be modified in place).

## 2.2 VUL Attack Scenarios and Desired Result

NISTIR 8011 uses an attack step model to summarize the six primary steps of cyber-attacks that NIST SP 800-53 controls work together to block or delay. The *VUL* security capability is intended to block or delay attacks only at the attack steps addressed in Figure 1 and Table 1.

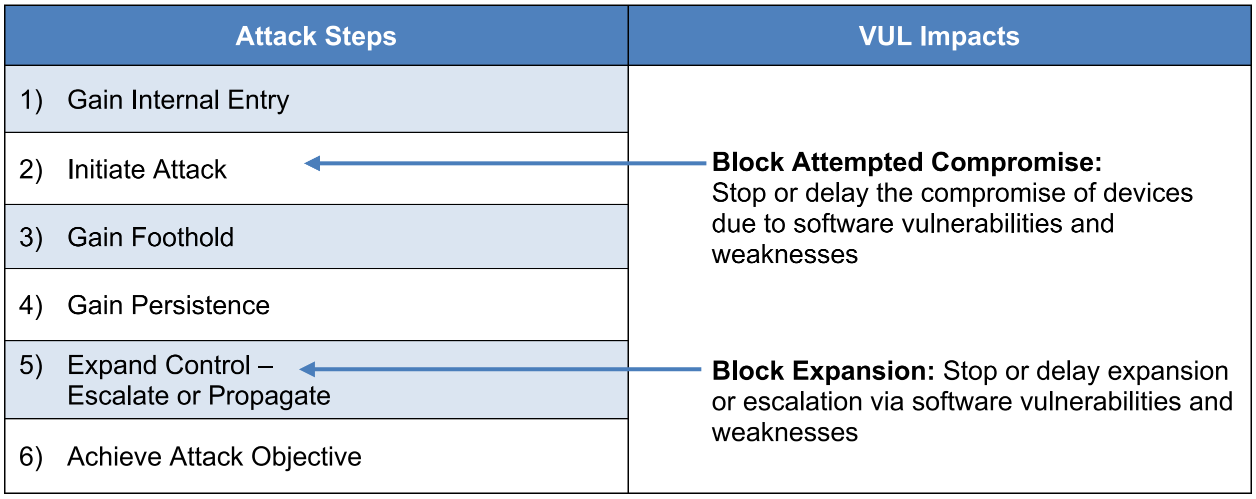
Attack Steps
VUL Impacts
1) Gain Internal Entry
2) Initiate Attack
3) Gain Foothold
4) Gain Persistence
5) Expand Control – Escalate or Propagate
6) Achieve Attack Objective
Block Attempted Compromise: Stop or delay the compromise of devices due to software vulnerabilities and weaknesses
Block Expansion: Stop or delay expansion or escalation via software vulnerabilities and weaknesses

**Figure 1: VUL Impact on an Attack Step Model**

### Notes on Figure 1

The attack steps shown in Figure 1 apply only to adversarial attacks. (See NISTIR 8011, Volume 1, Section 3.2.)

If the initiated attack succeeds in Step 2, the normal attack progression is that the attacker immediately gains a foothold on the affected device (via the software) in Step 3. Step 5 (propagation, expansion of control) includes a loop back to Step 2 on a different device from the one compromised in Step 5.

**Table 1: VUL Impact on an Attack Step Model**

| Attack Step Name | Attack Step Purpose (General) | Capability-Specific Defense |
|---|---|---|
| 2) Initiate Attack Internally | The attacker is inside the boundary and initiates an attack on some assessment object inside the boundary.<br><br>General examples (not limited to the VUL capability) include but are not limited to: user opens spear phishing email and/or clicks on attachment, laptop lost or stolen, user installs unauthorized software and/or hardware, unauthorized personnel gain physical access to restricted facility. | Block Attempted Compromise: Stop or delay the compromise of devices due to software vulnerabilities.<br><br>Specific examples (for the VUL capability) include but are not limited to: unauthorized software, weak setting configuration, and incomplete patching. |
| 5) Expand Control - Escalate or Propagate | The attacker has persistence on the assessment object and seeks to expand control by escalation of privileges on the assessment object or propagation to another assessment object.<br><br>General examples (not limited to the VUL capability) include but are not limited to: administrator privileges hijacked and/or stolen, administrator's password used by unauthorized party, secure configuration is changed and/or audit function is disabled, authorized users access resources the users do not need to perform job, process or program that runs as root is compromised and/or hijacked. | Block Expansion: Stop or delay expansion or escalation via software vulnerabilities.<br><br>Specific examples (for the VUL capability) include but are not limited to: unauthorized software, weak setting configuration, and incomplete patching. |

**Other examples of traceability among requirement levels**. While Table 1 shows software vulnerability management impacts on example attack steps, it is frequently useful to observe traceability among other sets of requirements. To examine such traceability, use Table 2 to reveal traceability from one requirement type to another by looking up the cell in the matching row and column of interest, and clicking on the link. For example, clicking on the link to Table 6 in the "Example Attack Steps" column reveals traceability between the Attack Step and the Sub-Capability/Defect Check ID, Name, and Purpose.

Full traceability from controls to attack steps requires the following links in a path:

1. Control-to-control item (provided by control item nomenclature)
2. Control item-to-determination statement (see Section 3.3)
3. Determination statement-to-defect check (aka, *sub-capability*; see Section 3.3)
4. Defect check (aka, *sub-capability*)-to-capability (provided by sub-capability nomenclature)
5. Defect check (and thus *capability*)-to-attack steps (see Table 6)
6. Capability-to-attack steps (see Figure 1 which is a summary of Table 6)

**Table 2: Traceability Among Requirement Levels**

| | Example Attack Steps | Capability | Sub-Capability/ Defect Check | Control Items |
|---|---|---|---|---|
| **Example Attack Steps** | | Figure 1<br>Table 1 | Table 6 | |
| **Capability** | Figure 1<br>Table 1 | | Table 6 | Section 3.3[a] |
| **Sub-Capability/ Defect Check** | Table 6 | Table 6 | | Section 3.3[b] |
| **Control Items** | | Section 3.3[a] | Section 3.3[b] | |

[a] Each level-four section (e.g., 3.3.1.1) is a control item that supports this capability.
[b] Refer to the table under the heading *Supporting Control Items* within each defect check.

## 2.3 Assessment Objects Managed and Assessed by VUL

The objects managed and assessed by the VUL capability include *software flaws* and *hardware flaws mitigated through software*.[8] Two kinds of software flaws are directly managed and assessed by the VUL capability: (1) **Common Vulnerabilities and Exposures (CVEs)** [CVE] identified, analyzed, and proven to exist in specific versions and patch levels of software files in use on devices; and (2) poor programming practices, called **Common Weakness Enumerations (CWEs)** [CWE], revealed in software code of software products and files in use on devices. Devices are protected when levels of risk arising from CVEs and CWEs contained in the software running on them are kept within organizational risk tolerances.

The number of software flaws present on a system rises and falls over time. The number increases as flaws are discovered and decreases as flaws are mitigated. Assessments need to be

[8] Hardware vulnerabilities are mitigated through software, such as applying a firmware or operating system patch that controls hardware access or turns off a hardware feature. In NISTIR 8011, software vulnerabilities are inclusive of hardware vulnerabilities mitigated through software. Hardware vulnerabilities that cannot be mitigated through software require physical changes to or replacement of hardware and are outside the scope of NISTIR 8011, Volume 4.

periodically repeated to maintain currency of information.

The VUL capability concentrates on protecting from *known* vulnerabilities for which potential attackers can easily and cheaply obtain knowledge and tools to guide their exploits. For most known vulnerabilities, patches exist to repair the vulnerabilities (if a patch does not yet exist, the vulnerability is considered to be a zero-day vulnerability, since the vulnerability has been disclosed but no mitigation is available; see Section 2.3.1). Unfortunately, *known* vulnerabilities are not always patched in a timely manner, which means that at any point in time some systems and system components likely have unpatched, exploitable software vulnerabilities.

An effective vulnerability management program—even one that is concentrating only on *known* vulnerabilities—is still useful in defending against well-funded, highly motivated, or capable attackers. Sophisticated attackers spend significant resources to find, weaponize, and conceal *unknown* vulnerabilities. They are frugal in deploying the weaponized *unknown* vulnerabilities, because the act of using the vulnerability risks revealing the vulnerability (i.e., taking it from unknown to known) and, once known, could lead to mitigation and neutralization by defenders. Well-funded and highly capable/motivated attackers, therefore, often prefer to exploit *known* vulnerabilities because known vulnerabilities are very cost-effective to attack and using them does not require spending precious *unknown* vulnerabilities to achieve the attack objectives. As such, if software is protected against *known* vulnerabilities, it raises the cost for even sophisticated attackers to succeed.

The VUL capability may also potentially address *unknown* vulnerabilities, for example, by scanning for poor coding practices (e.g., CWEs). Scanning for CWEs helps make *unknown* vulnerabilities *known* so they can be addressed. Thus, while the primary focus of the VUL capability is on *known* vulnerabilities (e.g., CVEs), risk associated with *unknown* vulnerabilities is also addressed by the focus on common sources of software flaws (e.g., CWEs).

### 2.3.1 Common Vulnerabilities and Exposures (CVEs)

The Common Vulnerabilities and Exposures (CVE) Program [CVE] provides a list of entries—each of which contains a unique identification number, a description, and at least one public reference—for publicly disclosed cybersecurity vulnerabilities that have been found in specific software and reported (to https://cve.mitre.org).[9] Important characteristics of CVEs for the purposes of automated assessment are:

- A CVE is a standard way of describing publicly disclosed cybersecurity vulnerabilities found in software;
- The CVE list is a dictionary with one entry per vulnerability or exposure;

[9] At the time of this publication, the CVE Program is in transition and the organizations managing the CVE Program and associated websites may change.

- The unique identifier of a CVE is designed to be interoperable with software systems across the industry; and

- A CVE is designed to convey the same meaning across products, tools, and services.

Once a vulnerability is disclosed and listed as a CVE, the vendor organization[10] maintaining the software begins work on creating a patch to close the vulnerability. The intent of patching and alternative methods to fix coding flaws is to discover and mitigate issues before an attacker can find and exploit them. The challenge for the defender is to stay one step ahead of the attacker while managing the increasing complexity of the code.

From the time that a vulnerability is discovered (by someone) until the organization controlling the software learns of it and provides a patch, the vulnerability is known as a zero-day vulnerability. The software is exposed during that interval and until a patch is released and applied. During this period of exposure, the options for defense from attack are limited to whitelisting,[11] applying common secure configurations, isolation, or removal.

When assessing risk, it is important to understand that the number of *reported* software vulnerabilities is not necessarily proportional to the *actual* number of software vulnerabilities across software vendors and products. Software that is implemented across platforms (e.g., Acrobat and Java) or implemented on the most widely used platforms (e.g., Microsoft or Cisco) typically presents the most attractive investments of time for attackers looking to cost-effectively exploit vulnerabilities. Consequently, code on widely used platforms report the most vulnerabilities.

The higher volume of reported vulnerabilities might be due to the increased focus of vulnerability research and reporting on more widely used software. However, a larger number of publicly disclosed vulnerabilities over a series of software releases typically indicates a higher degree of software *provider* maturity. It is not unusual for the providers of software platforms to have robust vulnerability disclosure, reporting, and management programs, all positive indicators of good risk management practices by the software provider. Conversely, it cannot be assumed that products with no *reported* vulnerabilities have no vulnerabilities.

The National Vulnerability Database [NVD] publishes CVE information to the public in a standard, machine-readable format. The NVD is one of the most comprehensive U.S. Government *open* sources of information on known and analyzed software vulnerabilities. The CVE Program also maintains a CVE dataset which might contain additional vulnerabilities[12] but provides less information about each vulnerability. Throughout this document the term NVD is

[10] The vendor organization as defined in Section 2.3.3.

[11] Note that while *malware* cannot execute in a whitelisted environment because it is unauthorized, attackers can still gain entry to an environment via *unmitigated vulnerabilities* in the whitelisted software itself. Consequently, software vulnerability management is of high priority even in a whitelisted software environment.

[12] The CVEs listed at [CVE] have been described less completely as stated on the MITRE website, which links to the NVD for more detailed descriptions. At the time of this publication, the CVE Program is in transition, and the organizations managing the CVE Program and associated websites may change.

used to refer to both nvd.nist.gov and cve.mitre.org. On occasion, industry is aware of publicly disclosed vulnerabilities not yet catalogued in NVD, but such sources are generally proprietary, not open.

Vulnerabilities are typically identified in the following ways:

- Each CVE entry in the NVD [NVD] is identified by CVE [CVE] numbering authorities [CNA]. A CNA may be a vendor, open source provider, or a researcher. The NVD receives a data feed from the CVE.

- Reputable software manufacturers with a mature and robust vulnerability management program report verified vulnerabilities.

- Vulnerabilities are reported by third-party ethical hackers.

Some vulnerabilities discovered in code that *can* be exploited as vulnerabilities are not publicly reported through the CVE Program and are therefore not listed in the NVD as CVEs.[13] There are several reasons a known vulnerability might not be publicly disclosed, including but not limited to the following:

- The vulnerability may have been discovered only by criminals and/or intelligence services who plan to exploit the vulnerability at some point and thus do not want it disclosed.

- The vulnerability might exist only in custom software and/or industrial control systems. Because of the limited number of users—and the potential sensitivity of the systems involved—such vulnerabilities might not be listed in the NVD because disclosing them might increase the risk of attack more than it would protect the affected systems.

- The vulnerability might exist in commercial off-the-shelf (COTS) software but might not be announced until a patch is available because disclosing it is thought to increase the risk of attack more than it would protect systems.

- The vulnerability might have been discovered by a vulnerability scanning provider *before* a CNA had assigned it a CVE ID.

Because of variations in vendor and attacker efforts to expose vulnerabilities as well as attacker efforts to conceal unreported vulnerabilities they have discovered, the number of CVEs listed for a software product is not necessarily reflective of the number of vulnerabilities *actually* present in the product.

[13] Some vendor organizations choose not to report vulnerabilities through the CVE Program and maintain vulnerability information and provide patches for their products via proprietary processes.

### 2.3.2 Common Weakness Enumerations (CWEs)

The Common Weakness Enumeration (CWE) is a taxonomy of well-known poor coding practices that are observed to manifest themselves in production software [CWE]. An example of a poor coding practice is *Improper Input Validation*, which is described on the CWE website:

> "When software does not validate input properly, an attacker is able to craft the input in a form that is not expected by the rest of the application. This will lead to parts of the system receiving unintended input, which may result in altered control flow, arbitrary control of a resource, or arbitrary code execution." [CWE]

Without adequate validation, an attacker may be able to change the program flow in an unintended way that creates a security vulnerability. See [CWE] for more examples.

Important characteristics of common weaknesses relevant to automated assessment include:

- Code analyzers are typically either static or dynamic. Static code analyzers are used to review bodies of source code (at the programming language level) or compiled code (at the machine language level). Dynamic code analyzers are used for observing code behavior *as it executes,* probing the application, and analyzing the application responses.

- While a CVE entry in the NVD often conveys information about the poor coding practice (i.e., the CWE) that resulted in the CVE, there is no guarantee that a CWE will result in a CVE. If the code is not analyzed, probed, or the weakness is not detected, then perhaps the flaw may not be noticed. In such a case, the flaw may eventually become a CVE.

- Even if the code is analyzed, and a piece of code is tagged as a CWE, it still may not actually result in a CVE because the code analyzers employed to detect poor coding practices produce many *false positive* results (i.e., the code analyzers identify code as containing poor coding practices when it does not).

- A code analyzer-identified poor coding practice that has not yet been verified to be a false positive is treated as if it were a software vulnerability. Because of the frequent occurrence of false positives in reports from code analyzers, remediation efforts often involve independent validation and verification of the identified poor coding practice. The additional analysis is needed to decide whether specific reported instances of poor coding practices are ignored (because they are false positives) or acted upon (because they are confirmed true positives) with subsequent appropriate response or reporting.

- CWEs are primarily of interest to parties who have *control* over and maintain source code—developers or testers in an organization that creates COTS, government off-the-shelf (GOTS), or custom code. However, CWEs are also of interest to organizations requiring verification of the security worthiness (i.e., the need for additional software security assurance) of software before deploying that software in a production environment.

There are three primary methods employed to ensure that code does not contain instances of CWEs. In order of effectiveness, the methods are:

1. Acquiring developers experienced with secure coding practices, and ensuring that existing developers are trained in secure coding practices;

2. Adopting processes to ensure that code is independently reviewed by a team of programmers experienced with secure coding practices; and

3. Using code analyzers, which can frequently find poor coding practices in code after it has been written or compiled; code analyzers automate review of applications.

### 2.3.3 Roles for Mitigation of CVEs and CWEs

To understand the NISTIR 8011-defined operational roles in vulnerability management, one must first consider the broader organizational roles. The Carnegie Mellon CERT special report, *The CERT Guide to Coordinated Vulnerability Disclosure* [SEI], describes vulnerability management roles as shown in the following diagram:

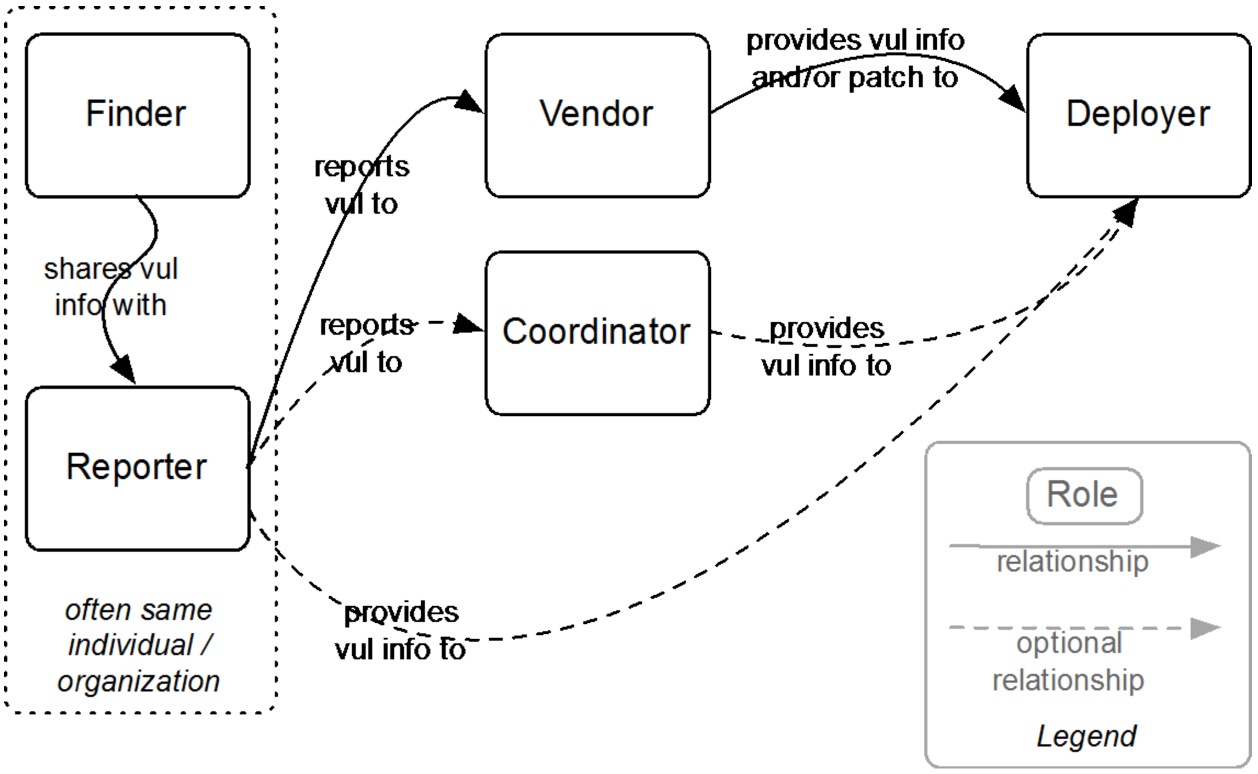

**Figure 2: Organizational Roles in Vulnerability Information Disclosure [SEI]**

Sometimes the roles are organizations, and sometimes the roles are individuals.

- Roles that are most frequently individuals include: *finder*, *reporter*, and *deployer* (e.g., home users, organizational users, system administrators).
- Roles that are most frequently organizations include: *vendor* and *coordinator* as well as *deployer* when the software is implemented in a government or business organization (e.g., software flaw manager, patch manager).

The role name *vendor* requires additional explanation. The relevant entity is the party that currently maintains the software that has the vulnerability. As the Carnegie Mellon CERT report states, "The vendor is the party responsible for updating the product containing the vulnerability." Thus, a *vendor*, in the context of vulnerability management, is not a third-party vendor from which the deployer (individual user or organization) purchased the software.

Note that for custom or in-house software code, the "party responsible for updating the product" may be an employee or group within the deployer organization or a service provider contracted by the deployer organization.

The *coordinator* role depicted in Figure 2 is normally external to the assessment organization and is not addressed in NISTIR 8011. Typically, the coordinator role is performed by groups such as SEI, the vulnerability scanner vendor, and others listed in the source document.

For maintained and authorized software, the NISTIR 8011-defined roles involved in the mitigation of CWEs and CVEs are the roles of Software Flaw Manager (SWFM), which corresponds to the CERT-defined *vendor* role, and Patch Manager (PatMan) which corresponds to the CERT-defined *deployer* role.[14] NISTIR 8011-defined roles for mitigation of CWEs and CVEs are depicted in Figure 3. Note that for *unauthorized* software, no patch is generated for a CVE, and there is likely to be *no* mitigation short of isolation or removal.

[14] For all roles supporting the VUL capability, see Section 2.7.

**Software Flaw Manager (SWFM)**
**(Vendor Organization Role)**[15]

*For software maintained by the vendor organization:*

- Creates patches for CVEs on software products maintained by the vendor organization (e.g., COTS and GOTS, software developed for others, or custom software developed for the organization)
- Finds CWEs on software maintained by the vendor organization and remediates
- Sometimes finds CWEs on COTS and GOTS developed by others[16]

*For software no longer maintained:*

- No patches are developed (unsupported)

**Patch Manager (PatMan)**
**(Deployer Organization Role)**

*For authorized software:*

- Finds devices with software needing application of patches created by the SWFM (i.e., installed software with CVEs)
- Applies patches to repair CVEs on installed software products.

*For unauthorized software:*

- Implements mitigation for unauthorized software (e.g., removal, isolation)

**Figure 3: CVE and CWE Mitigation Roles**

The SWFM and PatMan roles do not necessarily correspond to a person or people assigned exclusively to the tasks described in Figure 3. Rather, the tasks associated with the SWFM and PatMan roles may be duties included within broader roles.

#### 2.3.3.1 Software Flaw Manager (SWFM)

The SWFM is a vendor organization role. As such, it may be the responsibility of a third-party vendor organization over which the software-owning organization has little control (e.g., for most COTS software), or it may be the responsibility of the software-owning organization itself (e.g., for custom software developed in-house).

When a new vulnerability is discovered and confirmed to exist for software maintained by the vendor organization, the vulnerability is reported to the CVE Program so it can be listed as a

[15] Note that the SWFM role is typically a sub-role of software developer/maintainer or software development/maintenance manager rather than a completely separate role. The SWFM role is specified here because it is instrumental in mitigating software vulnerabilities as part of the VUL capability.

[16] Finding CWEs in software developed by others occurs primarily when the deployment organization has an internal SWFM and employs dynamic code scanners to look for use of CWE-listed coding practices in COTS or GOTS products developed by others.

CVE (or a decision is made not to, or when to, report the vulnerability).[17] The Software Flaw Manager (SWFM) within the vendor organization (i.e., the organization maintaining the software source code) then creates a new patch to mitigate the vulnerability.

- For COTS or externally maintained GOTS software, the patch is developed by the SWFM of the external vendor organization.
- For custom or internally maintained GOTS software, the patch is developed by the SWFM of the internal organization (vendor and deployer).

In either case – CVE or CWE mitigation – the SWFM is responsible for reporting poor coding practices and vulnerabilities within maintained software when discovered internal to the vendor organization, assessing the extent of code repairs required, making the necessary repairs, preparing a patch, performing integration, testing of the patch, preparing documentation, and distributing the finished patch to the deployer organization(s).

#### 2.3.3.2 Patch Manager (PatMan)

The Patch Manager (PatMan) is a deployer organization role that exists in the organization that has authorized and implemented the software (deployer organization).

The PatMan is responsible for detecting instances of CVEs present on devices and authorized software where available patches or a workaround solution needs to be applied. Software (i.e., code), as used here, is typically managed at the following levels of analysis:

- Software files (identified by digital fingerprint)
- Software source code (i.e., content of the software file[s] at the version/release/patch level)
- Software products (at the version/release/patch level)
- Firmware, if it can be modified (usually includes the BIOS, at the version/release/patch level)

The importance of accurately detecting the particular version/release and patch level of software cannot be overstated with respect to vulnerability management. Accurate version/release and patch level detection is important because variations of a software version/release and its corresponding patch level present different vulnerabilities depending on which patches have already been applied to that version/release. Digital fingerprints uniquely identify a particular version/release and patch level of a software file.

The primary tools employed by the PatMan in detecting CVEs present on a system are commercial vulnerability scanners. Vulnerability scanners automate the identification of CVEs

[17] For example, the vulnerability might not be reported until a patch is developed to prevent exploitation in the interim.

and the associated patches needed for each software file installed on each device in a system. Patches, in turn, contain information on the respective CVE(s) they are mitigating.

The PatMan is responsible for receiving patches from ***internal*** or ***external*** development organizations (i.e., vendor organizations), testing patch interoperability on the local system, and applying patches to devices in the production environment. Some CVEs can be mitigated by means other than patching before a patch becomes available. If so, the PatMan is responsible for applying any workaround mitigations in the interim period.

Patches are typically applied via a package management system which automates the steps of installation, upgrade, configuration, and removal of software files.[18] Alternatively, patches can be applied manually.

Some software products have patches that must be applied in a sequential order, in which case it is reasonable to refer to a patch *level*. Other products allow the selective application of patches in various orders. In such cases, the use of the expression *patch level* is more accurately denoted by the term *patch set.*[19] Patch sets are inherently more complex than *patch levels* because of the large number of combinations possible for the allowable order in which patches are applied. In this document, the term *patch level* refers to *whichever* patch level or patch set is applicable.

**Patching complexity introduced by shared code**. Some executables are *shared* by several software products. Dynamic Linked Library (DLL) executable files are prominent examples of shared software. In the case of DLL patching, one product may either protect or expose another product, depending on the vulnerabilities in the latest patch of the DLL installed and how the dependent software makes use of the library. For example, the "Heartbleed" vulnerability was found in the OpenSSL cryptography library but affected only the Transport Layer Security (TLS) implementation provided by OpenSSL. At the same time, OpenSSL cryptographic algorithm application programming interfaces (APIs) were not vulnerable. Thus, OpenSSL implementations of TLS exposed the Heartbleed vulnerability while OpenSSL implementations of only the cryptographic functions did not. Therefore, the shared nature of some software products is therefore a factor which complicates software vulnerability management.

**Patches on top of patches**. Unfortunately, it is still possible for a patch itself to contain *additional* software flaws that may be discovered later. Even if a given patch is free of *known* flaws, it is possible and even likely that new or different poor coding practices will be subsequently discovered that create new CVE entries in the NVD or result in new zero-day attacks to be exploited by adversaries.

[18] Examples of package management systems include but are not limited to Microsoft Windows Store, Linux Red Hat RPM Package Manager, Apple Mac App Store, Debian DPKG, and Comprehensive Perl Archive Network.

[19] Where patch sets are specified, the order of patch application can still affect the outcome (for example, when two or more patches change the same file).

## 2.4 Example VUL Data Requirements[20]

The desired state for the VUL capability is that the list of known vulnerabilities is up to date, accurate, and complete; and software products installed on all devices are free of known vulnerabilities.[21] Examples of data requirements for the VUL capability actual state are in Table 3. Examples of data requirements for the VUL capability desired state are in Table 4.

**Table 3: Example VUL Actual State Data Requirements**

| Data Item | Justification |
|---|---|
| The vulnerable software installed on every device is identified | To identify software flaws |
| Device software that is compliant with *alternative* mitigation specifications (to include the corresponding CVEs or local identifiers for flaws that are appropriately mitigated) | To prevent appropriately mitigated flaws from appearing in the results |
| Data necessary to determine how long the flaw has been present on a device; at a minimum: <br>• Date/time flaw was first discovered on the device <br>• Date/time flaw was last seen on the device | To determine how long vulnerabilities have been present on a device |

[20] Specific data required to support the VUL capability is variable based on organizational platforms, tools, configurations, etc.

[21] It is rarely possible or feasible to have *no* known vulnerabilities present (e.g., when a patch is not yet available or when a low risk vulnerability has not yet been patched), so the goal is to *minimize* the presence of known vulnerabilities in the environment.

**Table 4: Example VUL Desired State Data Requirements**

| Data Item | Justification |
|---|---|
| Authorized Hardware Inventory | To identify what devices to check |
| Associated value for every device attribute[a] | To prioritize defects associated with devices |
| A version-controlled, dated listing of all software products that have at least one known flaw, to include:<br>• Vulnerable software product in same format as the Authorized Software Inventory (Common Platform Enumeration [CPE] or Software Identification [SWID] [IR8060] equivalent)<br>• All CVEs associated with that software product<br>• All CWEs associated with that software product<br><br>For every locally defined[b] known vulnerability, maintain a version-controlled, dated listing to include:<br>• Vulnerable software product in same format as the Authorized Software Inventory (CPE or SWID equivalent)<br>• Identifier of all local vulnerabilities associated with that software product (e.g., CWE or other local identifier)<br>• Severity for each local vulnerability (e.g., Common Vulnerability Scoring System [CVSS] score equivalent) | To report on known flaws present on the system |
| Alternative mitigation specification[c] for any known vulnerability where the source vendor provides a mitigation option that can be implemented instead of patching/re-versioning the software to include:<br>• CVE or local identifier<br>• Associated system attributes<br>• Required/acceptable values | To prevent reporting on flaws mitigated by alternative methods for which the mitigation can be automatically checked[d] |
| Compliance definition (desired state specification criteria) | To determine compliance with each specific check |

[a] This value is defined by the organization based on the value it assigns to assets. See [IR8011-1] for an explanation of device attributes.
[b] Organizations can define data requirements and associated defects for their local environment.
[c] Some known vulnerabilities can be effectively mitigated by not installing sections of code, executables, or via configuration options.
[d] If the check that determines implementation of the alternative mitigation method can be verified by checking registry settings, executable hashes, or configuration settings, then a specification can be defined to automatically determine presence of the vulnerability.

## 2.5 VUL Concept of Operational Implementation

VUL identifies software (including on/in virtual machines) that is actually present on network devices (the actual state) and compares it with the desired state inventory to determine what known vulnerabilities (or weaknesses) are present on the software and deploy patching (or alternative methods of mitigation) to reduce the exploitability of the system.

The software vulnerability management capability concept of operations (CONOPS) illustrates how the VUL capability might be implemented. The CONOPS is central to the automated assessment process. (See Figure 4)

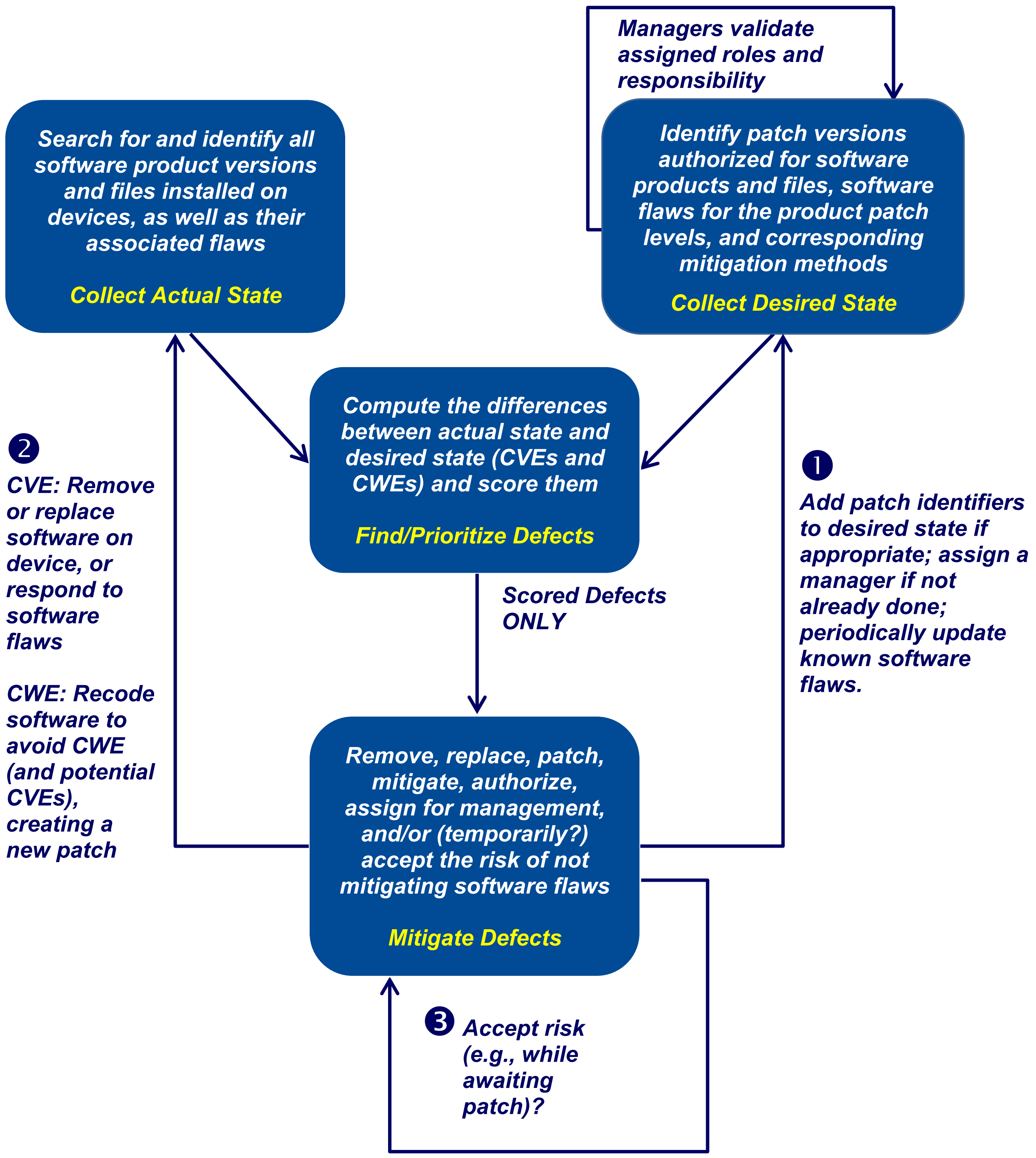
Managers validate assigned roles and responsibility
Search for and identify all software product versions and files installed on devices, as well as their associated flaws
Collect Actual State
Identify patch versions authorized for software products and files, software flaws for the product patch levels, and corresponding mitigation methods
Collect Desired State
Compute the differences between actual state and desired state (CVEs and CWEs) and score them
Find/Prioritize Defects
2
CVE: Remove or replace software on device, or respond to software flaws
CWE: Recode software to avoid CWE (and potential CVEs), creating a new patch
1
Add patch identifiers to desired state if appropriate; assign a manager if not already done; periodically update known software flaws.
Scored Defects ONLY
Remove, replace, patch, mitigate, authorize, assign for management, and/or (temporarily?) accept the risk of not mitigating software flaws
Mitigate Defects
3
Accept risk (e.g., while awaiting patch)?

**Figure 4: VUL Concept of Operations (CONOPS)**

### 2.5.1 Collect Actual State

The ISCM data collection process uses tools to identify the software files (and products) on network devices at the patch level, including software residing on mass storage and in firmware. The tools further provide the information required to compare the actual software and patch levels discovered (actual state) with the authorized patch levels (desired state). Examples of methods used to identify actual and desired patch levels are described in this section.

The ISCM data collection process also identifies how much of the target network is being monitored and how frequently in order to finalize the completeness and timeliness metrics. Devices might not be monitored on a specific scan because the device is not connected; the device is turned off; there is an error with the scanning process; the device is in a protected enclave not available to scanning; or the device is in an unexpected Internet Protocol (IP) address range (if the scanner is programmed for specific ranges); etc. Note that the inventory from the Hardware Asset Management (HWAM) capability can also be used as a check on what should be scanned if the quality of inventory data is acceptable.

The actual state data for all capabilities requires effective configuration management. Appendix G specifies how configuration management of the actual state is to be performed. The controls listed in Appendix G are metacontrols for the VUL capability assessment process.

#### 2.5.1.1 Actual State Data from the Operating System Software Database[22]

Some organizations use the operating system software database (OSSD) as a source for actual state data on the software versions present. However, OSSDs have several operating characteristics that may lead to errors in identifying software versions as described below:

- **Software is missing in the OSSD**. Some software on the device can run *without* having an OSSD entry (i.e., the OSSD might not be able to identify some software because there is no OSSD entry for the software).

- **Entry in the OSSD does not completely identify the software installed**. Different instances of installation media for a particular product version might install slightly different executables and thus might therefore have a different set of vulnerabilities. The OSSD might not detect this.

- **Uninstall processes for a product might remove the entry for a software file in the OSSD but not remove all of the code**. Problems with the uninstall process may leave open the possibility that vulnerable code remains on the device that is not identified in the OSSD and can therefore be exploited but is not identified in the OSSD.

[22] For example, the Windows registry or Linux package manager.

- **OSSD does not contain shared code**. Use of the OSSD as a source does not address shared code, which might be changed in the process of patching any of the programs that use the shared code. See Section 2.5.2.6.

#### 2.5.1.2 Actual State Data from Vulnerability Scanners

Use of vulnerability scanners is one of the most common ways to find CVEs in the actual state. Vulnerability scanners compare a list of software file versions known to contain vulnerabilities to the actual software file versions present on system devices. To ensure risk is accurately portrayed, verification of vulnerability scanner functionality is advisable before trusting results from a scanner. Vulnerability scanner verification includes the following:

- Ensure the vulnerability scanner is programmed by the organization to check for a high percentage of known vulnerabilities. If not, it might report fewer vulnerabilities than those that actually exist. The organization verifies the percentage of known vulnerabilities addressed by the scanner by comparing what the scanner checks for with the NVD and accepts the percentage addressed as part of the acquisition process for the scanner.

- Ensure that the false positive and false negative rates of the scanner are acceptable. No test is 100 % reliable. The tests used by the scanner to identify a vulnerability can report vulnerabilities when none exist (false positives), or the tests can fail to report vulnerabilities that do exist (false negatives). The false positive and false negative rates of the scanner are assessed as part of the acquisition process. Typically, there is an inverse relationship between false positive and false negative frequencies—as one goes up, the other goes down. There is a need to balance the two (i.e., balancing the risk of allowing excessive reporting of vulnerabilities that are not actual vulnerabilities [false positives] against the risk of too frequently failing to catch vulnerabilities that are actually present [false negatives]).

- Ensure that the vulnerability scanner vendor provides timely updates when new vulnerabilities are found and that the scanner can be updated quickly[23] with new detection code. Note that implementation of both detection (scanning) and response (patching) are necessary for vulnerability management to be effective.

#### 2.5.1.3 Actual State Data from Software Whitelisting Inventory

To the extent that the digital fingerprint for a software file with a vulnerability is known, it can be reliably and correctly found by inventorying software files on a device by their digital fingerprints. See more in Section 2.5.2.3.

The main problem with data from a software whitelisting inventory is that, *at the time of this*

[23] *Quickly,* here, is defined by the organization considering the expected speed with which adversaries are likely to exploit an undetected vulnerability.

*writing,* neither the NVD nor vendors report the digital fingerprint(s) of the software files carrying specific known vulnerabilities.[24]

#### 2.5.1.4 Actual State Data from Code Analyzers

Both dynamic and static code analyzers (see Glossary) are used to identify coding weaknesses that might materialize as vulnerabilities. Code analyzers are usually deployed *prior* to moving software to the operational state (i.e., in the earlier phases of the system engineering/system development life cycle) because the weaknesses found are cheaper to fix at the early stages of development.

In cases where the organization does not control the source code but desires to assess whether acquired products (or products whose acquisition is under consideration) have been engineered securely, *dynamic* code analyzers are frequently deployed to identify and diagnose security weaknesses. The organization deploys the acquired code in a production-like test environment, preferably *before* final purchasing decisions are made, and assesses whether weaknesses are at an acceptable level considering organizational risk tolerances.

### 2.5.2 Collect Desired State

The desired state for the VUL capability is the list or inventory of acceptable software file versions that limit *known* flaws in software installed on the network to within organizational risk tolerances. Thus, defining the desired state requires knowing how to identify—for all software files on the network—the optimal versions (i.e., patch levels) which contain the fewest known flaws. As indicated in the discussion of data collection methods below, identifying the desired state is a continually evolving process of incorporating and integrating information from multiple sources and, in some cases applying organizational risk tolerances to specific cases.

The desired state data for all capabilities requires effective configuration management. Appendix G specifies how configuration management of the desired state is to be performed. The controls in Appendix G are metacontrols for the VUL capability assessment process.

#### 2.5.2.1 Desired State Data from the National Vulnerability Database (NVD)

Since the desired state for the VUL capability, with respect to CVEs, is to have the most flaw-free software available, the NVD is an important source of information about CVEs to be minimized in the desired state. Each CVE has a unique identifier, and the NVD is the authoritative source of CVEs. Since NVD data is available to the public in digital form, many parties engaged in vulnerability identification and remediation download the NVD data and then integrate it with additional data, such as signatures for software files containing the CVE, articles written about the CVE, or identifiers for patches to the CVE.

[24] Requiring vendors to report data using digital fingerprints to reliably detect vulnerabilities would be a significant improvement to the vulnerability detection process.

#### 2.5.2.2 Desired State Data from Vulnerability Scanners

In addition to providing actual state data (as described in Section 2.5.1.2), vulnerability scanners are also a source of desired state data. Vulnerability scanners attempt to find known vulnerabilities in software on networked devices on a system by taking the CVE information from the NVD, linking the CVEs to identifiers for the software known to contain the CVEs, and then checking for the existence of the CVE-mitigating software patches on networked devices. The desired state, from the perspective of any given scan, is to have no CVEs present in software.[25]

*Note:* Since any given vulnerability scanner might only check for a portion of known vulnerabilities, each scanner defines the desired state differently.

#### 2.5.2.3 Desired State Data from Developer Package Manifests

One reason that vulnerability scanners are commercially viable is that they provide an acceptable approximation—within tolerable ranges of precision—of the specific instances of code on a device matching code known to contain CVEs. Package manifests provide an even more reliable option for identifying CVEs and their patches if they also contain digital fingerprints of each file.[26] Now, developers can (and frequently do) provide the following patch level file manifest information about each version:

- CVEs in that version
- An enumeration of the software files that contain each vulnerability, files that contain the fix for the vulnerability, and the respective digital fingerprint for each

When patch level manifest information is provided, scanners can provide very precise descriptions of the actual state (what CVEs are present) and desired state (what precise files *should* be there and at what patch level) for vulnerabilities on devices. When vendor-provided manifests at the patch level are used, the potential to limit error rates in scanning for vulnerabilities—both false positives and false negatives—is highest. Patch level manifests could come from SWID tags.

#### 2.5.2.4 Desired State Data from Approved Patch Level List

Some organizations simply develop an approved (and required) patch list. The approved patch list becomes the desired state. Any software without the required patches and/or other mitigations is tagged as vulnerable. The organizationally approved patch list is based on risk tolerance and is manually managed.

[25] Stated more precisely, the desired state is to have all of the software patched to the level consistent with organizational risk tolerances. For example, some organizations can tolerate CVEs considered by the organization to be low risk, for example.

[26] Package manifests enumerate the files contained in a patch distribution. If the manifest also contains a digital fingerprint for each file, then the entire contents of the patch can be validated for integrity/authenticity. Therefore, a more reliable approach to identifying CVEs is to require software vendors to provide package manifests that include a digital fingerprint for each file.

#### 2.5.2.5 Desired State Data from CWE (Coding Weakness) Information

The desired state for the VUL capability with respect to CWEs is that software exhibits no CWEs inconsistent with the organization's risk tolerance. Collecting and responding to CWE information is an important part of the process for custom software development. CWE information is also important for commercial software that organizations plan to deploy where the vendor is not yet trusted to find and report software vulnerabilities. Examples of tools for discovery of the actual and desired states for CWEs are discussed in Section 2.3.2.

#### 2.5.2.6 Desired State Data from Shared Code

Risk from software vulnerabilities may be further reduced by addressing shared code in the desired state. It is possible for an organization to identify software files updated by different products and compare the identified software files to the vulnerability list for the product or products using the shared code in order to identify whether a shared code file included in a patch is in the desired state.

### 2.5.3 Find/Prioritize Defects

The VUL capability focuses on comparing the versions of software objects discovered inside the assessment boundary (actual state) with an up-to-date list of the versions of software objects which *should* be there (desired state) and prioritizing a response (usually patching the vulnerable software). The desired state software objects are the versions selected for lowest risk of unpatched vulnerabilities. While the comparison of actual and desired state is most frequently performed with the assistance of commercial vulnerability scanners using vulnerability and patch information often derived from CVEs, other defects related to vulnerability management—such as CWEs the organization determines must be fixed—might be identified for unknown vulnerabilities with code analyzers. In any case, after the actual state to desired state comparison is completed, identified defects are prioritized[27] so that the appropriate response action (i.e., higher risk problems addressed first) can be taken.

## 2.6 NIST SP 800-53 Controls and Control Items that Support VUL

This section describes how the NIST SP 800-53 controls and control items needed to support the VUL capability were identified as well as the nomenclature used to clarify each control or control item's focus on software vulnerabilities.

### 2.6.1 Process for Identifying Needed Controls

The process used to determine the NIST SP 800-53 controls and control items needed to support a capability is described in detail in Volume 1 of this NISTIR, Section 3.5.2, Tracing Security Control Items to Capabilities. In short, the two steps are:

[27] Risk prioritization methods, which are necessary to score or prioritize defects, are out of scope for this publication. See [SP800-30] and [SP800-39] for a discussion on risk management, risk assessment, and risk prioritization.

1. Use a keyword search of the NIST SP 800-53 control text to identify controls or control items that might support the capability. See keyword rules in Appendix B.

2. Manually identify the NIST SP 800-53 controls or control items that *do* support the capability (true positives) and ignore those that do not (false positives).

The two steps above subsequently produce three sets of NIST SP 800-53 controls/control items:

1. Controls/control items in the low, moderate, and high baselines that support the VUL capability (listed in Section 3.3 and Section 3.4).

2. Controls/control items in the low, moderate, and high baselines that were selected by the keyword search but were manually determined to be false positives (listed in Appendix C).

3. Controls which were not in a baseline, and not analyzed further after the keyword search as follows:
   a. Program management (PM) controls, because PM controls do not apply to individual systems;
   b. Not selected controls—controls that are in NIST SP 800-53 but are not assigned to (selected in) a baseline; and
   c. Privacy controls.

   The unanalyzed controls/control items are listed in Appendix D, in case the organization wants to develop automated tests.

### 2.6.2 Control Item Nomenclature

Many control items that support the VUL capability also support several other capabilities. For example, the hardware asset management, software asset management, and configuration settings management capabilities can benefit from *configuration management* controls.

To clarify the scope of control items that support multiple capabilities as they relate to the VUL capability, expressions in the control item text are enclosed in curly brackets, e.g., {…software…}, to denote that a particular control item supports the VUL capability and focuses on—*and only on*—what is inside the curly brackets.

## 2.7 VUL-specific Roles and Responsibilities

Table 5 describes VUL-specific roles and their corresponding responsibilities. Figure 5 shows how the roles integrate with the concept of operations. An organization implementing automated assessment can customize its approach by assigning (allocating) the responsibilities to persons in existing roles.

**Table 5: Operational and Managerial Roles for VUL**[28]

| Role Code | Role Title | Role Description | Role Type |
|---|---|---|---|
| DSM | Desired State Managers (DSM) | Desired state managers are needed for both the ISCM Target Network and each assessment object. The desired state managers ensure that data specifying the desired state of the relevant capability is entered into the ISCM system's desired state data and is available to guide the actual state collection subsystem and identify defects. The DSM for the ISCM Target Network also resolves any ambiguity about which system authorization boundary has defects (if any).<br><br>Authorizers share some of the responsibilities by authorizing specific items (e.g., devices, software, or settings) and thus defining the desired state as delegated by the DSM. The DSM oversees and organizes this activity. | Operational |
| ISCM-Ops | ISCM Operators (ISCM-Ops) | ISCM operators are responsible for operating the ISCM system (see ISCM-Sys). | Operational |
| ISCM-Sys (Automated) | The automated system that collects, analyzes, and displays ISCM security-related information | ISCM-Sys is an automated role. It is included here to show tasks automated by the approach presented in this NISTIR.<br><br>The ISCM system: a) collects the desired state specification, b) collects security-related information from sensors (e.g., scanners, agents, training applications), and c) processes that information into a useful form.<br><br>To support task C, the system conducts specified defect check(s) and sends defect information to an ISCM dashboard covering the relevant system(s). The ISCM system is responsible for the assessment of most NIST SP 800-53 security controls. | Operational (Automated) |
| MAN | Manual Assessors | Assessments not automated by the ISCM system are conducted by human assessors using manual/procedural methods. Manual/procedural assessments might also be conducted to verify the automated security-related information collected by the ISCM system when there is a concern about data quality. | Operational |
| PatMan | Patch Manager (PatMan) | Assigned to a specific device or group of devices, patch managers are responsible for patching software products on affected devices. The patch managers are specified in the desired state specification. The patch manager may be a person or a group. If a group, a group manager is designated.<br><br>*Note*: The patch manager *role* might be performed by the device manager from the HWAM capability or the SWMan from the SWAM capability, depending on the volume of patching required. The role might also be performed by an automated central process managed by a centralized or distributed patch management team. | Operational |

[28] The role is filled by a person or a team unless stated that it is automated.

| Role Code | Role Title | Role Description | Role Type |
|---|---|---|---|
| RskEx | Risk Executive, System Owner, and/or Authorizing Official (RskEx) | Defined in SP 800-37 [SP800-37] and SP 800-39 [SP800-39] | Managerial |
| SWFM | Software Flaw Manager (SWFM) | Assigned to a specific software product or group of software products, software flaw managers are responsible for ensuring that CWE instances resulting in vulnerabilities in authorized software are found and corrected. As such, the SWFM is usually part of the development/maintenance team. The SWFMs are specified in the desired state specification for software products. The SWFM may be a person or a group. If a group, a group manager is designated.<br><br>*Note*: Most SWFM activities occur during systems engineering, but the process produces data to ensure that flaws are scored for software in production on the target network. Many (but not all) COTS software manufacturers track and score flaws independently.<br><br>The SWFM supports the desired state manager to ensure that risks from poor coding are tracked for custom software and software for which the manufacturer does not track security flaws. | Operational |
| SWMan | Software Manager | Software managers are assigned to specific devices and responsible for installing and/or removing software from the device. The key aspects of the software manager's responsibility are to ONLY install authorized software and to promptly remove ALL unauthorized software found. The software manager is also responsible for ensuring that software media is available to support the roll back of changes and restoration of software to prior states.<br><br>This role might be performed by the DM (device manager) and/or the PatMan (patch manager).<br><br>If users are authorized to install software, they are also SWMans (software managers) for the relevant devices. | Operational |

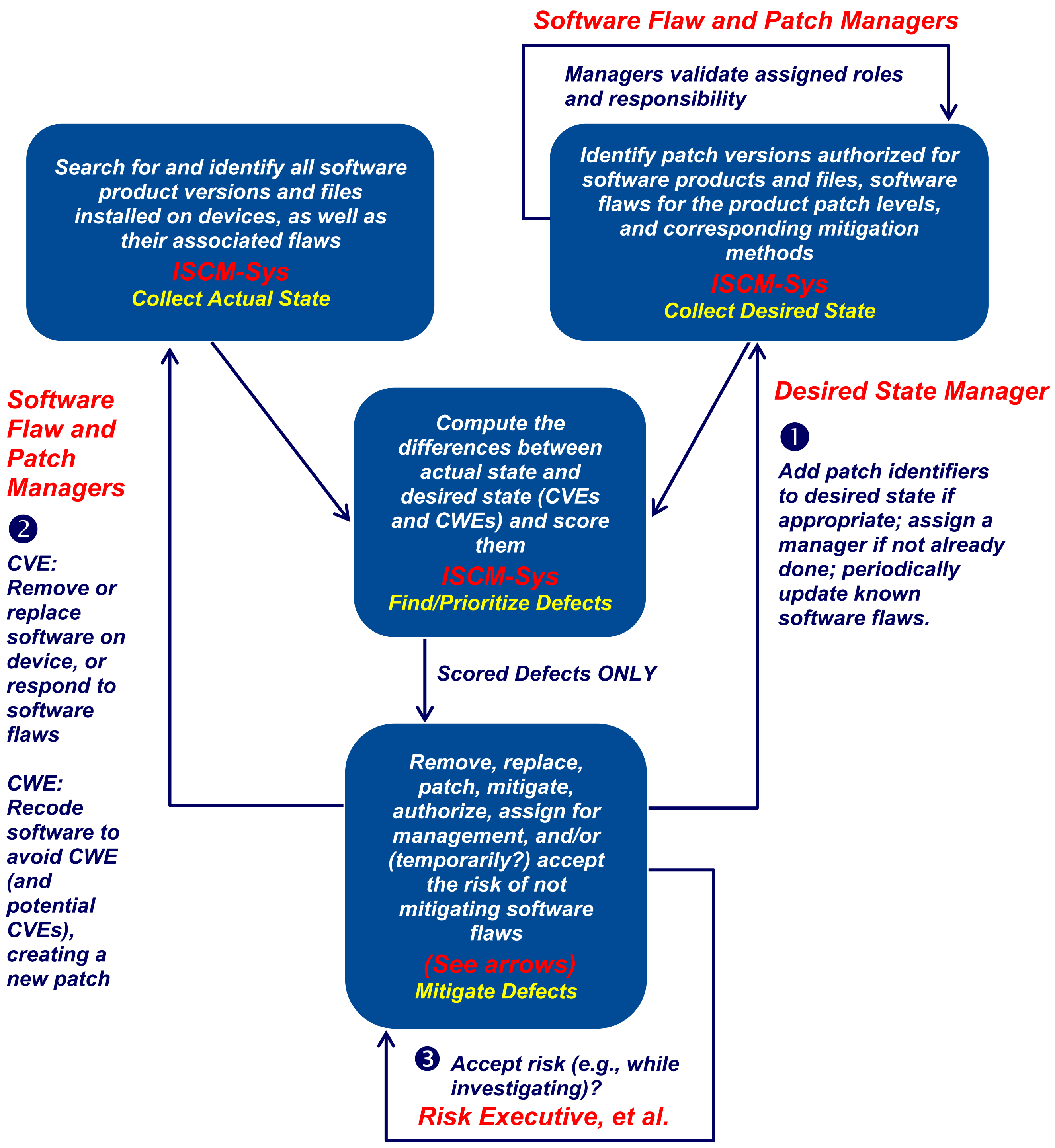
Software Flaw and Patch Managers
Managers validate assigned roles and responsibility
Search for and identify all software product versions and files installed on devices, as well as their associated flaws
ISCM-Sys
Collect Actual State
Identify patch versions authorized for software products and files, software flaws for the product patch levels, and corresponding mitigation methods
ISCM-Sys
Collect Desired State
Desired State Manager
❶
Add patch identifiers to desired state if appropriate; assign a manager if not already done; periodically update known software flaws.
Software Flaw and Patch Managers
❷
CVE: Remove or replace software on device, or respond to software flaws
CWE: Recode software to avoid CWE (and potential CVEs), creating a new patch
Compute the differences between actual state and desired state (CVEs and CWEs) and score them
ISCM-Sys
Find/Prioritize Defects
Scored Defects ONLY
Remove, replace, patch, mitigate, authorize, assign for management, and/or (temporarily?) accept the risk of not mitigating software flaws
(See arrows)
Mitigate Defects
❸ Accept risk (e.g., while investigating)?
Risk Executive, et al.

**Figure 5: Primary Roles in Automated Assessment of VUL**

## 2.8 VUL Assessment Boundary

The assessment boundary is all software on an entire *network* of computers from the innermost enclave out to where the network either ends in an airgap or interconnects to other network(s), typically the internet or the network(s) of a partner or partners. For the VUL capability, the boundary includes software on all devices, including software on removable devices found at the time of the scan. For more detail and definitions of some of the terms applicable to the assessment boundary, see Section 4.3.2 in Volume 1 of this NISTIR.

## 2.9 VUL Actual State and Desired State Specification

For information on the actual state and desired state specification for the VUL capability, see the assessment criteria notes section of the defect check tables in Section 3.2.

Note that many controls that support the VUL capability refer to a developed and updated inventory of software on devices (or other inventories). Software inventory is addressed in the SWAM capability. Note also that per the NIST SP 800-53A [SP800-53A] definition of *test*, testing of the VUL controls implies the need for specification of both an actual state inventory and a desired state inventory, allowing the test to compare the two inventories. The details of the comparison are described in the defect check tables in Section 3.2.

## 2.10 VUL Authorization Boundary and Inheritance

See Section 4.3.1 of Volume 1 of this NISTIR for information on how authorization boundaries are addressed in automated assessments. In short, for the VUL capability, software on each device is assigned to one and only one authorization (system) boundary per NIST SP 800-53, CM-08(5), *Information System Component Inventory | No Duplicate Accounting of Components.* The ISCM dashboard can include a mechanism for recording the assignment of software to authorization boundaries, making sure all software are assigned to at least one authorization boundary and that no software product is assigned to more than one authorization boundary.

For information on how inheritance of common controls is managed, see Section 4.3.3 of Volume 1 of this NISTIR. For VUL, many utilities, database management software products, web server software objects, and parts of the operating system provide inheritable support and/or controls for other systems. The ISCM dashboard can include a mechanism to record information about inheritance and use it to assess the system's overall risk.

## 2.11 VUL Assessment Criteria Recommended Scores and Risk-Acceptance Thresholds

General guidance on options for risk scores[29] to be used to set thresholds is outside of the scope of this NISTIR and is being developed elsewhere. For the VUL capability, organizations are encouraged to use metrics that consider both the average risk score and maximum risk score per device. Note that vulnerability scanning tools may perform risk scoring in their assessments of

[29] In the context of VUL, a risk score (also called a *defect score*) is a measure of how exploitable a defect is.

detected vulnerabilities.

## 2.12 VUL Assessment Criteria Device Groupings to Consider

To support automated assessment and ongoing authorization, software is clearly grouped by authorization boundary (see Control Items CM-8(a) and CM-8(5) in NIST SP 800-53). Software is also clearly organized by the role of the persons—device managers, patch managers, software managers, and software flaw managers—performing software vulnerability management on specific devices (see Control Item CM-8(4) in NIST SP 800-53). In addition to these two important groupings, the organization may want to use other groupings for risk analysis as discussed in Section 5.6 of Volume 1 of this NISTIR.

# 3 VUL Security Assessment Plan Documentation Template

## 3.1 Introduction and Steps for Adapting This Plan

This section provides templates for the security assessment plan in accordance with NIST SP 800-37 and NIST SP 800-53A. The documentation elements are described in Section 6 of Volume 1 of this NISTIR. Section 9 of the same volume specifically describes how the templates and documentation relate to the assessment tasks and work products defined in NIST SP 800-37 and NIST SP 800-53A.

Figure 6 shows the suggested steps to adapt the security assessment plan to the organization's needs and implement automated monitoring. The steps are expanded to more detail in the following three sections.

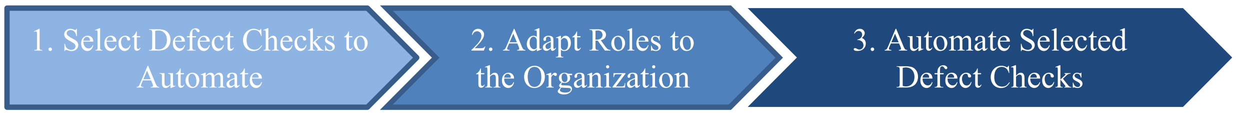
1. Select Defect Checks to Automate
2. Adapt Roles to the Organization
3. Automate Selected Defect Checks

**Figure 6: Main Steps in Adapting the Plan Template**

### 3.1.1 Select Defect Checks to Automate

The sub-steps for selecting defect checks to automate are described in this section.

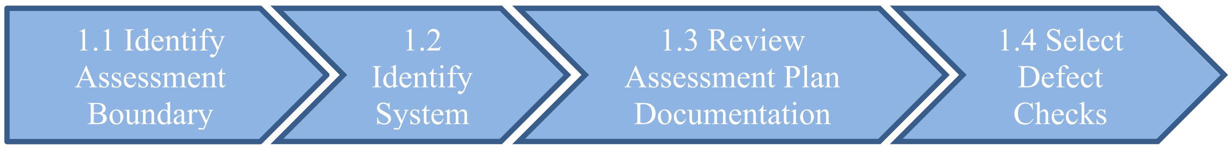
1.1 Identify Assessment Boundary
1.2 Identify System
1.3 Review Assessment Plan Documentation
1.4 Select Defect Checks

**Figure 7: Sub-Steps to Select Defect Checks to Automate**

Take the following sub-steps, shown in Figure 7, to select which defect checks to automate:

**Sub-step 1.1 Identify Assessment Boundary:** Identify the assessment boundary to be covered. (See Section 4.3 of Volume 1 of this NISTIR.)

**Sub-step 1.2 Identify System Impact:** Identify the Federal Information Processing Standard (FIPS) 199-defined impact level (high water mark) for the assessment boundary identified in Sub-step 1.1 [FIPS199]. (See [SP 800-60-v1] and/or organizational categorization records.)

**Sub-step 1.3 Review Security Assessment Plan Documentation:**

- Review the defect checks documented in Section 3.2 to get an initial sense of the proposed items to be tested.
- Review the security assessment plan narratives in Section 3.2 to understand how the defect checks apply to the controls that support vulnerability management.

**Sub-step 1.4 Select Defect Checks:**

- Based on Sub-steps 1.1, 1.2, and 1.3 as well as an understanding of the organization's risk tolerance, use Table 6 in Section 3.2.3 to identify the defect checks necessary to assess the effectiveness of controls implemented in accordance with the system impact level and organizational risk tolerance.

- Mark the defect checks necessary as selected in Section 3.2.2. The organization is not required to use automation, but automation of control assessment adds value to the extent that it:

    1. Produces assessment results timely enough to better defend against attacks and/or
    2. Reduces the cost of assessment over the long term.

### 3.1.2 Adapt Roles to the Organization

The sub-steps for adapting roles to the organization are described in this section.

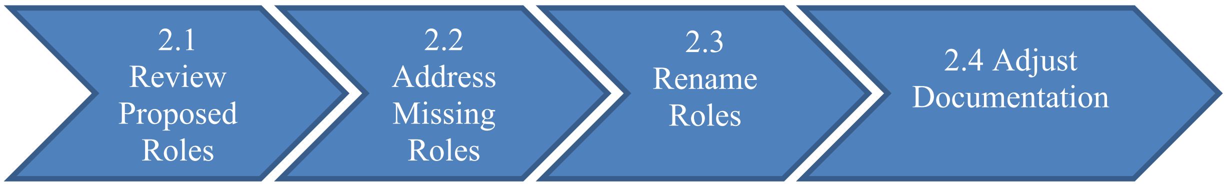

**Figure 8: Sub-Steps to Adapt Roles to the Organization**

Take the following sub-steps, shown in Figure 8, to adapt the roles to the organization.

**Sub-step 2.1 Review Proposed Roles**: Proposed roles are described in Section 2.7, VUL Specific Roles and Responsibilities (Illustrative).

**Sub-step 2.2 Address Missing Roles:** Identify any required roles not currently assigned in the organization. Determine how to assign the unassigned roles.

**Sub-step 2.3 Rename Roles:** Identify the organization-specific names that match each role. (Note that more than one proposed role might be performed by the same organizational role.)

**Sub-step 2.4 Adjust Documentation:** Map the organization-specific roles to the roles proposed herein, in one of two ways (either may be acceptable):

- Add a column to the table in Section 2.7 for the organization-specific role and list the organization-specific role names there

- Use global replace to change the role names throughout the documentation from the names proposed in this NISTIR to the organization-specific names.

### 3.1.3 Automate Selected Defect Checks

The sub-steps for automating selected defect checks are described in this section.

3.1 Add Defect Checks
3.2 Adjust Data Collection
3.3 Operate the ISCM System
3.4 Use the Results to Manage Risk

**Figure 9: Sub-Steps to Automate Selected Defect Checks**

Take the following sub-steps, shown in Figure 9, to implement automation defect checks.

**Sub-step 3.1 Add Defect Checks:** Review the defect check definition and add checks as needed based on organizational risk tolerance and expected attack types. [Role: DSM (See Section 2.7.)]

**Sub-step 3.2 Adjust Data Collection:**

- Review the actual state information needed and configure automated sensors to collect the required information. [Role: ISCM-Sys (See Section 2.7.)]
- Review the matching desired state specification that was specified or add additional specifications to match the added actual state to be checked. Configure the collection system to receive and store the desired state specification in a form that can be compared automatically to the actual state data. [Role: ISCM-Sys (See Section 2.7.)]

**Sub-step 3.3 Operate the ISCM System:**

- Operate the collection system to identify both security and data quality defects.
- Configure the collection system to send security and data quality information to the defect management dashboard.

**Sub-step 3.4 Use the Results to Manage Risk:** Use the results to respond to higher risk findings first and measure potential residual risk to inform aggregate risk acceptance decisions. If risk is determined to be too great for acceptance, the results may also be used to help prioritize further mitigation actions.[30]

## 3.2 VUL Sub-Capabilities and Defect Check Tables and Template

This section describes the specific test templates that are proposed and considered adequate to

[30] Risk is determined based on threats, vulnerabilities, likelihoods, and impacts. See NIST SP 800-30 [SP800-30] and NIST SP 800-39 [SP800-39] for more information on risk management, risk assessment, and risk prioritization. Automated vulnerability scanning tools may also provide information on risk and risk prioritization for identified software vulnerabilities.

assess the control items that support the VUL capability. See Section 5 of Volume 1 of this NISTIR for an overview of defect checks and Section 4.1 of Volume 1 for an overview of the actual state and desired state specifications discussed in the Assessment Criteria Notes for each defect check. Sections 3.2.1, 3.2.2, and 3.2.3 of this document describe the foundational, data quality, and local defect checks, respectively. The *Supporting Control Item(s)* data in Sections 3.2.1, 3.2.2, and 3.2.3 specify which controls, when ineffective, might cause a particular defect check to fail. The association between control items and defect checks provides further documentation on why the check (test) might be needed. Refer to Section 3.1 on how to adapt the defect checks (and roles specified therein) to the organization.

Data found in this section can be used in both defect check selection and root cause analysis. Section 3.2.4 documents how each sub-capability (tested by a defect check) serves to support the overall capability by addressing certain example attack steps and/or data quality issues. Appendix G can also be used to support root cause analysis.

The Defect Check Templates are organized as follows:

- In the section beginning with "*The purpose of this sub-capability…*," the sub-capability being tested by the defect check is defined and assessment criteria described. How the sub-capabilities block or delay certain example attack steps is described in Section 3.2.4.

- In the section beginning "*The defect check to assess…*," the defect check name and the assessment criteria to be used to assess sub-capability effectiveness in achieving its purpose are described.

- In the section beginning "*Example Responses*," examples of potential responses when the check finds a defect and what role is likely responsible are described. Potential responses (with example primary responsibility assignments) are common actions and are appropriate when defects are discovered in a given sub-capability. The example primary responsibility assignments do not change the overall management responsibilities defined in other NIST guidance. Moreover, the response actions and responsibilities can be customized by each organization to best adapt to local circumstances.

- Finally, in the section beginning "*Supporting Control Items,*" the control items that work together to support the sub-capability are listed. Identification of the supporting control items is based on the mapping of defect checks to control items in Section 3.3. Each sub-capability is supported by a set of control items. Thus, if any of the listed supporting controls fail, the defect check fails, and overall risk is likely to increase.

As noted in Section 3.1, this material is designed to be customized and adapted to become part of an organization's security assessment plan.

### 3.2.1 Foundational Sub-Capabilities and Corresponding Defect Checks

NISTIR 8011, Volume 4 proposes one foundational security-oriented defect check for the VUL capability. The foundational check is designated VUL-F01.

Defect checks may be computed for individual checks (e.g., foundational, data quality, or local) or summarized for various groupings of devices (e.g., device manager, device owner, system) out to the full assessment boundary. The foundational defect check was selected for its value for summary reporting. The *Selected* column indicates whether the check is to be implemented.

#### 3.2.1.1 Reduce Software Vulnerabilities Sub-Capability and Defect Check VUL-F01

The purpose of this sub-capability is defined as follows:

| Sub-Capability Name | Sub-Capability Purpose |
|---|---|
| Reduce software vulnerabilities | Reduce the presence of software vulnerabilities (CVEs) listed in the reference defect list (e.g., National Vulnerability Database [NVD]). |

The defect check to assess whether this sub-capability is operating effectively is defined as follows:

| Defect Check ID | Defect Check Name | Assessment Criteria Notes | Selected |
|---|---|---|---|
| VUL-F01 | Vulnerable Software | 1) The actual state is the list (inventory) of software product, version, release, and patch levels present on the device.<br>2) The desired state specification is to have minimal (i.e., acceptable) risk from CVEs or equivalent.<br>3) A defect is the presence of an unacceptable software vulnerability (CVE or equivalent) as listed in the reference defect list (i.e., National Vulnerability Database [NVD] or other vulnerability dataset accepted for use by the organization). | Yes |

Example Responses:

| Defect Check ID | Potential Response Action | Primary Responsibility |
|---|---|---|
| VUL-F01 | Patch the software | PatMan |
| VUL-F01 | Remove the software | SWMan |
| VUL-F01 | Assess as false positive | RskEx |
| VUL-F01 | Reduce false positives | ISCM-Ops |
| VUL-F01 | Apply workaround mitigation | PatMan |
| VUL-F01 | Accept risk | RskEx |
| VUL-F01 | Oversee and coordinate response | DSM |

Supporting Control Items:

| Defect Check ID | Baseline | NIST SP 800-53 Control Item Code |
|---|---|---|
| VUL-F01 | Low | RA-5(a) |
| VUL-F01 | Low | RA-5(b) |
| VUL-F01 | Low | RA-5(c) |
| VUL-F01 | Low | RA-5(d) |
| VUL-F01 | Low | RA-5(e) |
| VUL-F01 | Low | SI-2(a) |
| VUL-F01 | Low | SI-2(c) |
| VUL-F01 | Low | SI-2(d) |
| VUL-F01 | Moderate | SA-11(d) |
| VUL-F01 | High | SI-2(1) |

### 3.2.2 Data Quality Sub-Capabilities and Corresponding Defect Checks

NISTIR 8011, Volume 4 proposes four *data quality* defect checks, designated VUL-Q01 through VUL-Q04. The data quality defect checks are important because they provide the information necessary to determine how reliable the overall assessment automation process is— information which can be used to decide how much to trust the other defect check data (i.e., provide greater assurance about security control effectiveness). The data quality defect checks were selected for their value in summary reporting and are not associated with specific control items. The *Selected* column indicates which of the checks is implemented by the organization. Data quality checks are described more completely in NISTIR 8011, Volume 1, Overview, Section 5.5., "Data Quality Measures."

#### 3.2.2.1 Ensure Completeness of Device-Level Reporting Sub-Capability and Defect Check VUL-Q01

The purpose of this sub-capability is defined as follows:

| Sub-Capability Name | Sub-Capability Purpose |
|---|---|
| Ensure completeness of device-level reporting | Ensure that devices expected to report VUL information to the actual state inventory have reported to prevent CVEs and CWEs from going undetected. |

The defect check to assess whether this sub-capability is operating effectively is defined as follows:

| Defect Check ID | Defect Check Name | Assessment Criteria Notes | Selected |
|---|---|---|---|
| VUL-Q01 | Non-reporting devices | 1) The actual state is the list of devices in the desired state in HWAM-F01 that report software vulnerabilities (CVEs or equivalent, and CWEs)<br>2) The desired state is the list of actual devices detected in HWAM-F01, whether authorized or not.<br>3) A defect occurs when a device in the desired state has not been detected as recently as expected in the actual state. Criteria are developed to define the threshold for “as recently as expected” for each device or device type based on the same considerations listed in HWAM-Q01. | Yes |

Example Responses:

| Defect Check ID | Potential Response Action | Primary Responsibility |
|---|---|---|
| VUL-Q01 | Restore device reporting | ISCM-Ops |
| VUL-Q01 | Declare device missing | DM |
| VUL-Q01 | Accept risk | RskEx |
| VUL-Q01 | Oversee and coordinate response | RskEx |

Supporting Control Items:

| Defect Check ID | Baseline | NIST SP 800-53 Control Item Code |
|---|---|---|
| VUL-Q01 | Low | RA-5(a) |
| VUL-Q01 | Low | RA-5(c) |
| VUL-Q01 | Low | SI-2(a) |
| VUL-Q01 | Low | SI-2(b) |
| VUL-Q01 | High | SI-2(1) |

### 3.2.2.2 Ensure Completeness of Defect Check-Level Reporting Sub-Capability and Defect Check VUL-Q02

The purpose of this sub-capability is defined as follows:

| Sub-Capability Name | Sub-Capability Purpose |
|---|---|
| Ensure completeness of defect check-level reporting | Ensure that defect check information is correctly reported in the actual state inventory to prevent systematic inability to check any applicable defect on any device. |

The defect check to assess whether this sub-capability is operating effectively is defined as follows:

| Defect Check ID | Defect Check Name | Assessment Criteria Notes | Selected |
|---|---|---|---|
| VUL-Q02 | Non-reporting applicable defect checks | 1) The actual state is the set of vulnerabilities that was tested and collected in each collection cycle for each device.<br>2) The desired state is the set of vulnerabilities that are defined as applicable for that device and that *should* therefore have been tested and collected.<br>3) A defect is any vulnerability for a device from the desired state that was not tested and collected in the actual state. The defects may be of two types:<br>a. The collection system does not test and collect data for the defect on *any* applicable device; or<br>b. The collection system only tests and collects data for the defect on *some* of the applicable devices.<br><br>Notes on root cause:<br>• Item 3a) is usually a systematic error of the collection system.<br>• Item 3b) may be a related to the interaction of the device and the collection system; either the device or the collection system may be the root cause. | Yes |

Example Responses:

| Defect Check ID | Potential Response Action | Primary Responsibility |
|---|---|---|
| VUL-Q02 | Restore defect check reporting | ISCM-Ops |
| VUL-Q02 | Accept risk | RskEx |
| VUL-Q02 | Oversee and coordinate response | RskEx |

Supporting Control Items:

| Defect Check ID | Baseline | NIST SP 800-53 Control Item Code |
|---|---|---|
| VUL-Q02 | Low | RA-5(a) |
| VUL-Q02 | Low | RA-5(b) |
| VUL-Q02 | Low | RA-5(c) |
| VUL-Q02 | Low | SI-2(a) |
| VUL-Q02 | Low | SI-2(b) |
| VUL-Q02 | Moderate | RA-5(1) |
| VUL-Q02 | Moderate | RA-5(2) |
| VUL-Q02 | High | SI-2(1) |

#### 3.2.2.3 Ensure Overall Defect Check Reporting Completeness Sub-Capability and Defect Check VUL-Q03

The purpose of this sub-capability is defined as follows:

| Sub-Capability Name | Sub-Capability Purpose |
|---|---|
| Ensure overall defect check reporting completeness | Ensure that data for as many defect checks as possible are correctly reported in the actual state inventory to prevent defects from going undetected. |

The defect check to assess whether this sub-capability is operating effectively is defined as follows:

| Defect Check ID | Defect Check Name | Assessment Criteria Notes | Selected |
|---|---|---|---|
| VUL-Q03 | Low completeness-metric | The completeness metric is not a device-level defect but is applied to any collection of devices such as those in an authorization boundary. The completeness metric is used in assessing the trustworthiness of the collection system.<br><br>1) The actual state is the number of specified defect checks provided by the collection system in a reporting window.<br>*Note*: A specific check-device combination may only be counted once in the required minimal reporting period. For example, if checks are to be done every three days, a check done twice in that timeframe would still count as one check. However, if there are 30 days in the reporting window, that check-device combination could be counted for each of the 10 three-day periods included.<br>2) The desired state is the number of specified defect checks that should have been provided in that same reporting window.<br>*Note*: Different devices may have different sets of specified checks based on device function/type. The desired state in this example includes 10 instances of each specified defect check combinations for each of the three-day reporting cycles in a 30-day reporting window.<br>3) The metric is *completeness*, defined as the actual state number divided by the desired state number. Completeness is the percentage of specified defect checks collected during the reporting window. Completeness measures long-term ability to collect all needed data.<br>4) A defect is when completeness is too low (based on the defined threshold). When completeness is low, the risk of defects being undetected increases. An acceptable level of completeness balances technical feasibility against the need for 100 % completeness. | Yes |

Example Responses:

| Defect Check ID | Potential Response Action | Primary Responsibility |
|---|---|---|
| VUL-Q03 | Restore completeness | ISCM-Ops |
| VUL-Q03 | Accept risk | RskEx |
| VUL-Q03 | Oversee and coordinate response | RskEx |

Supporting Control Items:

| Defect Check ID | Baseline | NIST SP 800-53 Control Item Code |
|---|---|---|
| VUL-Q03 | Low | RA-5(a) |
| VUL-Q03 | Low | RA-5(c) |
| VUL-Q03 | Low | SI-2(a) |
| VUL-Q03 | Low | SI-2(b) |
| VUL-Q03 | Moderate | SI-2(2) |
| VUL-Q03 | High | SI-2(1) |

#### 3.2.2.4 Ensure Overall Reporting Timeliness Sub-Capability and Defect Check VUL-Q04

The purpose of this sub-capability is defined as follows:

| Sub-Capability Name | Sub-Capability Purpose |
|---|---|
| Ensure overall reporting timeliness | Ensure that data for as many defect checks as possible are reported in a timely manner in the actual state to limit delays in defect detection. To be effective, defects need to be found and mitigated considerably faster than they can be exploited. |

The defect check to assess whether this sub-capability is operating effectively is defined as follows:

| Defect Check ID | Defect Check Name | Assessment Criteria Notes | Selected |
|---|---|---|---|
| VUL-Q04 | Poor timeliness metric | The timeliness metric is not a device-level defect but can be applied to *any* collection of devices such as those within an authorization boundary. It is used in assessing the accuracy of the collection system.<br><br>1) The actual state is the number of specified defect checks provided by the collection system in one collection cycle—the period in which each defect should be checked once.<br>*Note*: A specific check-device combination is only counted once per collection cycle.<br>2) The desired state is the number of specified defect checks that *should have been* provided by the collection system in one collection cycle.<br>*Note*: Different devices may have different sets of specified checks based on device function/type.<br>3) The metric is *timeliness*, defined as the actual state number divided by the desired state number. Timeliness is the percentage of specified defect checks actually collected in the reporting cycle. Timeliness measures the percentage of data that is collected as recently as required.<br>4) A defect is when timeliness is too poor (based on the defined threshold). When timeliness is poor the risk of undetected defects increases. | Yes |

Example Responses:

| Defect Check ID | Potential Response Action | Primary Responsibility |
|---|---|---|
| VUL-Q04 | Restore frequency | ISCM-Ops |
| VUL-Q04 | Accept risk | RskEx |
| VUL-Q04 | Oversee and coordinate response | RskEx |

Supporting Control Items:

| Defect Check ID | Baseline | NIST SP 800-53 Control Item Code |
|---|---|---|
| VUL-Q04 | Low | RA-5(a) |
| VUL-Q04 | Low | RA-5(b) |
| VUL-Q04 | Low | RA-5(c) |
| VUL-Q04 | Low | SI-2(a) |
| VUL-Q04 | Low | SI-2(b) |
| VUL-Q04 | Low | SI-2(c) |
| VUL-Q04 | Moderate | SI-2(2) |
| VUL-Q04 | High | SI-2(1) |

### 3.2.3 Local Sub-Capabilities and Corresponding Defect Checks

This section includes one local defect check, VUL-L01, as an example of what organizations may add to the foundational check to support more complete automated assessment of NIST SP 800-53 controls that support VUL.

Organizations exercise authority to manage risk by choosing whether to select specific defect checks for implementation. In general, selecting more defect checks may lower risk (if there is capacity to address the defects found) and provide greater assurance but may also increase the cost of detection and mitigation. The organization selects defect checks for implementation (or not) to balance benefits and costs and prioritize risk response actions by focusing first on the problems that pose the greatest risk (i.e., manage risk).

Note that a local defect check may also include options to make the defect check more or less rigorous as the risk tolerance of the organization and impact level of the system indicates.

The "Selected" column is present to indicate which of the local defect checks the organization chooses to implement as documented or as modified by the organization.

#### 3.2.3.1 Reduce Poor Coding Practices Sub-Capability and Defect Check VUL-L01

The purpose of this sub-capability is defined as follows:

| Sub-Capability Name | Sub-Capability Purpose |
|---|---|
| Reduce poor coding practices | Reduce the presence of poor software coding practices (CWEs) listed in the reference https://cwe.mitre.org. |

The defect check to assess whether this sub-capability is operating effectively is defined as follows:

| Defect Check ID | Defect Check Name | Assessment Criteria Notes | Selected |
|---|---|---|---|
| VUL-L01 | Poor coding practices | The assessment for poor coding practices applies to any software for which the organization is responsible for finding—and developing patches to correct—poor coding practices. The assessment for poor coding practices may also be applied to COTS and/or GOTS software to verify results obtained from the software provider.<br><br>1) The actual state is the list (inventory) of software products and associated version, release and patch levels present on the device to which CWE code analysis is applied.<br>*Note*: The inventory list of software files originates with the SWAM capability. The inventory list of hardware devices originates with the HWAM capability.<br>2) The desired state specification is to have minimal (i.e., acceptable) risk present from instances of CWEs in the software files on the device.<br>3) A defect is the presence of an unacceptable coding practice (CWE) on a device in the actual state.<br>*Note*: Because code analyzers may produce a non-negligible number of false positives, it is important that false positives be identified by an independent risk assessment function (e.g., independent verification and validation team; assessment team; system security officer; organizational risk executives) and removed from the poor coding practice instance list. | To be determined (TBD) by organization |

Example Responses:

| Defect Check ID | Potential Response Action | Primary Responsibility |
|---|---|---|
| VUL-L01 | Assess as false positive | RskEx |
| VUL-L01 | Remove the software | PatMan |
| VUL-L01 | Obtain patch | SWFM |
| VUL-L01 | Patch the software | PatMan |
| VUL-L01 | Apply workaround mitigation | PatMan |
| VUL-L01 | Accept risk | RskEx |
| VUL-L01 | Oversee and coordinate response | DSM |

Supporting Control Items:

| Defect Check ID | Baseline | NIST SP 800-53 Control Item Code |
|---|---|---|
| VUL-L01 | Low | RA-5(a) |
| VUL-L01 | Low | RA-5(c) |
| VUL-L01 | Low | RA-5(d) |
| VUL-L01 | Low | RA-5(e) |
| VUL-L01 | Low | SI-2(a) |
| VUL-L01 | Low | SI-2(c) |
| VUL-L01 | Low | SI-2(d) |
| VUL-L01 | Moderate | SA-11(d) |
| VUL-L01 | High | SI-2(1) |

### 3.2.4 Security Impact of Each Sub-Capability on an Attack Step Model

Table 6 shows the primary ways the defect checks derived from the NIST SP 800-53 security controls contribute to blocking attacks/events as described in Figure 1. *Note*: certain cells in Table 6 may contain repeated information from other cells. This is by design, and due to the automated nature of the NISTIR 8011 development.

**Table 6: Mapping of Attack Steps to Security Sub-Capability**

| Attack Step | Attack Step Description | Sub-Capability ID and Name | Sub-Capability Purpose |
|---|---|---|---|
| 2) Initiate Attack Internally | The attacker is inside the boundary and initiates an attack on some assessment object internally.<br><br>Examples include: user opens spear phishing email or clicks on attachment; user installs unauthorized software or hardware; unauthorized personnel gain physical access to restricted facility and perform a malicious act. | VUL-F01: Reduce software vulnerabilities | Reduce the presence of software vulnerabilities (CVEs) listed in the reference defect list (e.g., National Vulnerability Database [NVD]). |
| 2) Initiate Attack Internally | The attacker is inside the boundary and initiates an attack on some assessment object internally.<br><br>Examples include: user opens spear phishing email or clicks on attachment; user installs unauthorized software or hardware; unauthorized personnel gain physical access to a restricted facility and perform a malicious act. | VUL-L01: Reduce poor coding practices | Reduce the presence of poor software coding practices (CWEs) listed in the reference https://cwe.mitre.org. |

| Attack Step | Attack Step Description | Sub-Capability ID and Name | Sub-Capability Purpose |
|---|---|---|---|
| 5) Expand Control - Escalate or Propagate | The attacker has persistence on the object and seeks to expand control by escalation of privileges on the object or propagation to another object.<br><br>Examples include: administrator privileges hijacked or stolen; administrator's password used by unauthorized party; secure configuration is changed and/or audit function is disabled; authorized users access resources they do not need to perform job; process or program that runs as root compromised or hijacked; cascading failures take down entire communications infrastructure. | VUL-F01: Reduce software vulnerabilities | Reduce the presence of software vulnerabilities (CVEs) listed in the reference defect list (e.g., National Vulnerability Database [NVD]). |
| 5) Expand Control - Escalate or Propagate | The attacker has persistence on the object and seeks to expand control by escalation of privileges on the object or propagation to another object.<br><br>Examples include: administrator privileges hijacked or stolen; administrator's password used by unauthorized party; secure configuration is changed and/or audit function is disabled; authorized users access resources they do not need to perform job; process or program that runs as root compromised or hijacked; cascading failures take down entire communications infrastructure. | VUL-L01: Reduce poor coding practices | Reduce the presence of poor software coding practices (CWEs) listed in the reference https://cwe.mitre.org. |

## 3.3 VUL Control (Item) Security Assessment Plan Narrative Tables and Templates

The security assessment plan narratives in this section are designed to provide the core of an assessment plan for the automated assessment as described in Section 6 of Volume 1 of this NISTIR. The narratives are supplemented by the other material in this section, including defect check tables (defining the tests to be used), and are summarized in the Control Allocation Tables in Section 3.4.

The roles referenced in the narratives match the roles defined by NIST in relevant special publications (e.g., NIST SP 800-37) and/or the VUL-specific roles defined in Section 2.7. The roles can be adapted and/or customized to the organization as described in the introduction to Section 3.

The determination statements listed here have been derived from the relevant control item language, specifically modified by the following three adjustments:

1. The limiting or scoping phrase {…software…} (possibly along with additional information within the brackets as appropriate) is inserted in determination statements where necessary for control items that apply to more capability areas than just VUL. The limiting phrase tailors the control item to remain within VUL since the same control item could appear in other capabilities with the relevant scoping for that capability. For example, using the limiting phrase {…software…} is appropriate where the control could apply to vulnerabilities in both software and hardware.

2. Where a control item includes inherently different actions that are best assessed by different defect checks (typically because the assessment criteria are different), the control item may be divided into multiple VUL-applicable determination statements.

3. Part of a control item may not apply to VUL while another part does. For example, consider the control item RA-5(b): the control text lists actions that do not necessarily apply to VUL capability, such as ensuring that scanning tools use standards for enumerating platforms (applies to the HWAM and SWAM capabilities) and assessing improper configurations not related to vulnerabilities (applies to the CSM capability).

> RA-5 VULNERABILITY SCANNING: …Employs vulnerability scanning tools and techniques that facilitate interoperability among tools and automate parts of the vulnerability management process by using standards for: 1) **Enumerating platforms**, software flaws, and **improper configurations**; 2) Formatting checklists and test procedures; and 3) Measuring vulnerability impact… [Emphasis added.]

To address the issue of multi-capability control items, the determination statements in this volume include only the portion of the control item applicable to the VUL capability.

### 3.3.1 Outline Followed for Each Control Item

The literal text of the control item follows the heading *Control Item Text*.

There may be one or more determination statements for each control item. Each determination statement is documented in a table, noting the:

- Determination statement ID (Control Item ID concatenated with the determination statement number, where determination statement number is enclosed in curly brackets);
- Determination statement text;
- Implemented by (responsibility);
- Assessment boundary;
- Assessment responsibility;
- Assessment method;
- Selected column (TBD by the organization);
- Rationale for risk acceptance (thresholds) (TBD by the organization);
- Frequency of assessment;[31] and
- Impact of not implementing the defect check (TBD by the organization).

The determination statement details are followed by a table showing the defect checks (and related sub-capability) that might be caused to fail if the control being tested fails.

The resulting text provides a template for the organization to edit as described in Section 3.1.

### 3.3.2 Outline Organized by Baselines

This section includes security control items selected in the NIST SP 800-53 Low, Moderate, and High baselines and that support the VUL capability. For convenience, the control items are presented in three sections as follows:

1. **Low Baseline Control Items** (Section 3.3.3). Security control items in the low baseline, which are required for all systems.
2. **Moderate Baseline Control Items** (Section 3.3.4). Security control items in the moderate baseline, which are also required for the high baseline.
3. **High Baseline Control Items** (Section 3.3.5). Security control items that are required only for the high baseline.

Table 7 illustrates the applicability of the security control items to each baseline.

[31] While automated tools may be able to assess as frequently as every 3-4 days, organizations determine the appropriate assessment frequency in accordance with the ISCM strategy.

**Table 7: Applicability of Control Items**

| **FIPS-199[a] (NIST SP 800-60)[b] System Impact Level** | **1) Low Control Items (Section 3.3.3)** | **2) Moderate Control Items (Section 3.3.4)** | **3) High Control Items (Section 3.3.5)** |
|---|---|---|---|
| Low | Applicable | ▨ | ▨ |
| Moderate | Applicable | Applicable | ▨ |
| High | Applicable | Applicable | Applicable |

[a] FIPS-199 defines Low, Moderate, and High overall potential impact designations.
[b] See [SP800-60-v1], Section 3.2.

### 3.3.3 Low Baseline Security Control Item Narratives

#### *3.3.3.1 Control Item RA-5(a): VULNERABILITY SCANNING*

**Control Item Text**

Control: The organization:

a. Scans for vulnerabilities in the information system and hosted applications [Assignment: organization-defined frequency and/or randomly in accordance with organization-defined process] and when new vulnerabilities potentially affecting the system/applications are identified and reported.

**Determination Statement 1**

| Determination Statement ID | Determination Statement Text |
|---|---|
| RA-5(a){1} | Determine if the organization: scans for {software} vulnerabilities in the system and hosted applications [Assignment: organization-defined frequency and/or randomly in accordance with organization-defined process]. |

**Roles and Assessment Methods**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| RA-5(a){1} | ISCM-Ops | ISCM-TN | ISCM-Sys | Test | | | | |

**Defect Check Rationale Table**

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| **Determination Statement ID** | **Defect Check ID** | **Defect Check Name** | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **conducting scans for {software} vulnerabilities in the information system and hosted applications [Assignment: organization-defined frequency and/or randomly (with adequate frequency) in accordance with organization-defined process]** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| RA-5(a){1} | VUL-Q04 | Poor timeliness metric | …poor timeliness of overall ISCM reporting. |

#### *3.3.3.2 Control Item RA-5(a): VULNERABILITY SCANNING*

### Control Item Text

Control: The organization:

a. Scans for vulnerabilities in the information system and hosted applications [Assignment: organization-defined frequency and/or randomly in accordance with organization-defined process] and when new vulnerabilities potentially affecting the system/applications are identified and reported

### Determination Statement 1

| Determination Statement ID | Determination Statement Text |
|---|---|
| RA-5(a){2} | Determine if the organization: [ensures] that when new vulnerabilities potentially affecting the system/applications are identified, they are [added to the scanning process]. |

### Roles and Assessment Methods

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| RA-5(a){2} | DSM | ISCM-TN | ISCM-Sys | Test | | | | |

**Defect Check Rationale Table**

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| **Determination Statement ID** | **Defect Check ID** | **Defect Check Name** | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **ensuring that when new vulnerabilities potentially affecting the system/applications are identified, they are [added to the scanning process]** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| RA-5(a){2} | VUL-Q02 | Non-reporting applicable defect checks | …applicable defect checks failing to report. |

#### 3.3.3.3 *Control Item RA-5(b): VULNERABILITY SCANNING*

**Control Item Text**

Control: The organization:

b. Employs vulnerability scanning tools and techniques that facilitate interoperability among tools and automate parts of the vulnerability management process by using standards for:

1. Enumerating platforms, software flaws, and improper configurations;
2. Formatting checklists and test procedures; and
3. Measuring vulnerability impact.

**Determination Statement 1**

| Determination Statement ID | Determination Statement Text |
|---|---|
| RA-5(b){1} | Determine if the organization: employs vulnerability scanning tools and techniques that facilitate interoperability among tools and automate parts of the vulnerability management process by using standards for [identifying] software flaws. |

**Roles and Assessment Methods**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| RA-5(b){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | |

### Defect Check Rationale Table

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| Determination Statement ID | Defect Check ID | Defect Check Name | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **using standards for [identifying] software flaws** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| RA-5(b){1} | VUL-Q02 | Non-reporting applicable defect checks | …applicable defect checks failing to report. |

### Determination Statement 2

| Determination Statement ID | Determination Statement Text |
|---|---|
| RA-5(b){2} | Determine if the organization: employs vulnerability scanning tools and techniques that facilitate interoperability among tools and automate parts of the vulnerability management process by using standards for formatting checklists and test procedures to minimize false positives. |

### Roles and Assessment Methods

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| RA-5(b){2} | ISCM-Ops | ISCM-TN | ISCM-Sys | Test | | | | |

### Defect Check Rationale Table

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| Determination Statement ID | Defect Check ID | Defect Check Name | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **using standards for formatting checklists and test procedures to minimize false positives** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| RA-5(b){2} | VUL-F01 | Vulnerable Software | …the presence of software vulnerabilities (CVEs or equivalent). |

### Determination Statement 3

| Determination Statement ID | Determination Statement Text |
|---|---|
| RA-5(b){3} | Determine if the organization: employs vulnerability scanning tools and techniques that facilitate interoperability among tools and automate parts of the vulnerability management process by using standards for formatting checklists and test procedures to minimize false negatives. |

### Roles and Assessment Methods

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| RA-5(b){3} | MAN | ISCM-TN | MAN | TBD | | | | |

### Defect Check Rationale Table

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

Not applicable because tested manually.

#### *3.3.3.4 Control Item RA-5(c): VULNERABILITY SCANNING*

**Control Item Text**

Control: The organization:

c. Analyzes vulnerability scan reports and results from security control assessments.

**Determination Statement 1**

| Determination Statement ID | Determination Statement Text |
|---|---|
| RA-5(c){1} | Determine if the organization: analyzes vulnerability scan reports and results from security control assessments. |

**Roles and Assessment Methods**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| RA-5(c){1} | RskEx | ISCM-TN | ISCM-Sys | Test | | | | |

**Defect Check Rationale Table**

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| Determination Statement ID | Defect Check ID | Defect Check Name | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **analyzing vulnerability scan reports and results from security control assessments** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| RA-5(c){1} | VUL-F01 | Vulnerable Software | …the presence of software vulnerabilities (CVEs or equivalent). |
| RA-5(c){1} | VUL-L01 | Poor coding practices | …the presence of software with poor coding practices (CWEs or equivalent). |
| RA-5(c){1} | VUL-Q01 | Non-reporting devices | …a device failing to report software vulnerabilities within the specified time frame. |

| **Determination Statement ID** | **Defect Check ID** | **Defect Check Name** | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **analyzing vulnerability scan reports and results from security control assessments** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| RA-5(c){1} | VUL-Q02 | Non-reporting applicable defect checks | …applicable defect checks failing to report. |
| RA-5(c){1} | VUL-Q03 | Low completeness metric | …completeness of overall ISCM reporting not meeting the threshold. |
| RA-5(c){1} | VUL-Q04 | Poor timeliness metric | …poor timeliness of overall ISCM reporting. |

#### *3.3.3.5 Control Item RA-5(d): VULNERABILITY SCANNING*

### Control Item Text

Control: The organization:

d. Remediates legitimate vulnerabilities [Assignment: organization-defined response times] in accordance with an organizational assessment of risk

### Determination Statement 1

| Determination Statement ID | Determination Statement Text |
|---|---|
| RA-5(d){1} | Determine if the organization: remediates legitimate vulnerabilities [Assignment: organization-defined response times] in accordance with an organizational assessment of risk. |

### Roles and Assessment Methods

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| RA-5(d){1} | PatMan | ISCM-TN | ISCM-Sys | Test | | | | |

### Defect Check Rationale Table

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| Determination Statement ID | Defect Check ID | Defect Check Name | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **remediating legitimate vulnerabilities** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| RA-5(d){1} | VUL-F01 | Vulnerable Software | …the presence of software vulnerabilities (CVEs or equivalent). |
| RA-5(d){1} | VUL-L01 | Poor coding practices | …the presence of software with poor coding practices (CWEs or equivalent). |

#### *3.3.3.6 Control Item RA-5(e): VULNERABILITY SCANNING*

### Control Item Text

Control: The organization:

e. Shares information obtained from the vulnerability scanning process and security control assessments with [Assignment: organization-defined personnel or roles] to help eliminate similar vulnerabilities in other information systems (i.e., systemic weaknesses or deficiencies).

### Determination Statement 1

| Determination Statement ID | Determination Statement Text |
|---|---|
| RA-5(e){1} | Determine if the organization: shares information obtained from the vulnerability scanning process with [Assignment: organization-defined personnel or roles] to help eliminate similar vulnerabilities in other systems (i.e., systemic weaknesses or deficiencies). |

### Roles and Assessment Methods

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| RA-5(e){1} | RskEx | ISCM-TN | ISCM-Sys | Test | | | | |

**Defect Check Rationale Table**

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| **Determination Statement ID** | **Defect Check ID**[32] | **Defect Check Name** | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **sharing information obtained from the vulnerability scanning process with [Assignment: organization-defined personnel or roles] to help eliminate similar vulnerabilities in other information systems** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| RA-5(e){1} | VUL-F01 | Vulnerable Software | …the presence of software vulnerabilities (CVEs or equivalent). |
| RA-5(e){1} | VUL-L01 | Poor coding practices | …the presence of software with poor coding practices (CWEs or equivalent). |

[32] As written, defect checks VUL-F01 and VUL-L01 assume that there is an automated dashboard to which personnel or roles designated for sharing vulnerability scanning information already have access. To be more thorough, the organization could verify that: 1) the dashboard displays scan results, 2) the organization-defined personnel or roles have access, and/or 3) the organization-defined personnel or roles are using the access. Such verifications could be done either manually or through automation, in each case by comparing what is desired (sharing information on vulnerability scan results with the organization-defined personnel or roles) to what is observed (whether the information is actually shared and reviewed by defined personnel or roles).

#### 3.3.3.7 *Control Item SI-2(a): FLAW REMEDIATION*

**Control Item Text**

Control: The organization:

a. Identifies, reports, and corrects information system flaws

**Determination Statement 1**

| Determination Statement ID | Determination Statement Text |
|---|---|
| SI-2(a){1} | Determine if the organization: identifies and reports system flaws. |

**Roles and Assessment Methods**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| SI-2(a){1} | SWFM | ISCM-TN | ISCM-Ops | Test | | | | |

**Defect Check Rationale Table**

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| Determination Statement ID | Defect Check ID | Defect Check Name | Rationale<br>If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **identifying and reporting information system flaws** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| SI-2(a){1} | VUL-Q01 | Non-reporting devices | …a device failing to report software vulnerabilities within the specified time frame |
| SI-2(a){1} | VUL-Q02 | Non-reporting applicable defect checks | …applicable defect checks failing to report |
| SI-2(a){1} | VUL-Q03 | Low completeness metric | …completeness of overall ISCM reporting not meeting the threshold |
| SI-2(a){1} | VUL-Q04 | Poor timeliness metric | …poor timeliness of overall ISCM reporting |

## Determination Statement 2

| Determination Statement ID | Determination Statement Text |
|---|---|
| SI-2(a){2} | Determine if the organization: corrects system flaws. |

## Roles and Assessment Methods

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| SI-2(a){2} | PatMan | ISCM-TN | ISCM-Sys | Test | | | | |

## Defect Check Rationale Table

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| Determination Statement ID | Defect Check ID | Defect Check Name | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **correcting information system flaws** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| SI-2(a){2} | VUL-F01 | Vulnerable Software | …the presence of software vulnerabilities (CVEs or equivalent). |
| SI-2(a){2} | VUL-L01 | Poor coding practices | …the presence of software with poor coding practices (CWEs or equivalent). |

#### *3.3.3.8 Control Item SI-2(b): FLAW REMEDIATION*

**Control Item Text**

Control: The organization:

b. Tests software and firmware updates related to flaw remediation for effectiveness and potential side effects before installation

**Determination Statement 1**

| Determination Statement ID | Determination Statement Text |
|---|---|
| SI-2(b){1} | Determine if the organization: tests software and firmware updates related to flaw remediation for effectiveness and potential side effects before installation. |

**Roles and Assessment Methods**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| SI-2(b){1} | MAN | ISCM-TN | MAN | TBD | | | | |

**Defect Check Rationale Table**

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

Not applicable because tested manually.

#### *3.3.3.9 Control Item SI-2(c): FLAW REMEDIATION*

### Control Item Text

Control: The organization:

c. Installs security-relevant software and firmware updates within [Assignment: organization-defined time period] of the release of the updates

### Determination Statement 1

| Determination Statement ID | Determination Statement Text |
|---|---|
| SI-2(c){1} | Determine if the organization: installs security-relevant software and firmware updates within [Assignment: organization-defined time period] of the release of the updates. |

### Roles and Assessment Methods

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| SI-2(c){1} | PatMan | ISCM-TN | ISCM-Sys | Test | | | | |

**Defect Check Rationale Table**

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| **Determination Statement ID** | **Defect Check ID** | **Defect Check Name** | **Rationale**<br>If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **installing security-relevant software and firmware updates within [Assignment: organization-defined time period] of the release of the updates** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| SI-2(c){1} | VUL-F01 | Vulnerable Software | …the presence of software vulnerabilities (CVEs or equivalent). |
| SI-2(c){1} | VUL-L01 | Poor coding practices | …the presence of software with poor coding practices (CWEs or equivalent). |
| SI-2(c){1} | VUL-Q04 | Poor timeliness metric | …poor timeliness of overall ISCM reporting. |

#### *3.3.3.10 Control Item SI-2(d): FLAW REMEDIATION*

### Control Item Text

Control: The organization:

d. Incorporates flaw remediation into the organizational configuration management process

### Determination Statement 1

| Determination Statement ID | Determination Statement Text |
|---|---|
| SI-2(d){1} | Determine if the organization: incorporates flaw remediation into the organizational configuration management process. |

### Roles and Assessment Methods

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| SI-2(d){1} | SWFM | ISCM-TN | ISCM-Sys | Test | | | | |

### Defect Check Rationale Table

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| Determination Statement ID | Defect Check ID | Defect Check Name | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **incorporating flaw remediation into the organizational configuration management process** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| SI-2(d){1} | VUL-F01 | Vulnerable software | …presence of software vulnerabilities (CVEs or equivalent) |
| SI-2(d){1} | VUL-L01 | Poor coding practices | …presence of software with poor coding practices (CWEs or equivalent) |

### 3.3.4 Moderate Baseline Security Control Item Narratives

#### *3.3.4.1 Control Item RA-5(1): VULNERABILITY SCANNING | UPDATE TOOL CAPABILITY*

**Control Item Text**

The organization employs vulnerability scanning tools that include the capability to readily update the information system vulnerabilities to be scanned.

**Determination Statement 1**

| Determination Statement ID | Determination Statement Text |
|---|---|
| RA-5(1){1} | Determine if the organization: employs vulnerability scanning tools to actually update the system vulnerabilities to be scanned. |

**Roles and Assessment Methods**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| RA-5(1){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | |

**Defect Check Rationale Table**

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| Determination Statement ID | Defect Check ID | Defect Check Name | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **updating the information system vulnerabilities to be scanned** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| RA-5(1){1} | VUL-F01 | Vulnerable Software | …the presence of software vulnerabilities (CVEs or equivalent). |
| RA-5(1){1} | VUL-L01 | Poor coding practices | …the presence of software with poor coding practices (CWEs or equivalent). |
| RA-5(1){1} | VUL-Q02 | Non-reporting applicable defect checks | …applicable defect checks failing to report. |

#### *3.3.4.2 Control Item RA-5(2): VULNERABILITY SCANNING | UPDATE BY FREQUENCY / PRIOR TO NEW SCAN / WHEN IDENTIFIED*

**Control Item Text**

The organization updates the information system vulnerabilities scanned [Selection (one or more): [Assignment: organization-defined frequency]; prior to a new scan; when new vulnerabilities are identified and reported].

**Determination Statement 1**

| Determination Statement ID | Determination Statement Text |
|---|---|
| RA-5(2){1} | Determine if the organization: updates the system vulnerabilities scanned [Selection (one or more): [Assignment: organization-defined frequency]; prior to a new scan; when new vulnerabilities are identified and reported]. |

**Roles and Assessment Methods**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| RA-5(2){1} | ISCM-Ops | ISCM-TN | ISCM-Sys | Test | | | | |

**Defect Check Rationale Table**

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| Determination Statement ID | Defect Check ID | Defect Check Name | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **updating the information system vulnerabilities scanned when new vulnerabilities are identified and reported** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| RA-5(2){1} | VUL-F01 | Vulnerable Software | …the presence of software vulnerabilities (CVEs or equivalent). |
| RA-5(2){1} | VUL-L01 | Poor coding practices | …the presence of software with poor coding practices (CWEs or equivalent). |

| **Determination Statement ID** | **Defect Check ID** | **Defect Check Name** | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **updating the information system vulnerabilities scanned when new vulnerabilities are identified and reported** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| RA-5(2){1} | VUL-Q02 | Non-reporting applicable defect checks | …applicable defect checks failing to report. |

### 3.3.4.3 Control Item SA-11(d): DEVELOPER SECURITY TESTING AND EVALUATION

#### Control Item Text

Control: The organization requires the developer of the information system, system component, or information system service to:

d. Implement a verifiable flaw remediation process

#### Determination Statement 1

| Determination Statement ID | Determination Statement Text |
|---|---|
| SA-11(d){1} | Determine if the organization: requires the developer of the system, system component, or system service to implement a verifiable flaw remediation process. |

#### Roles and Assessment Methods

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| SA-11(d){1} | SWFM | ISCM-TN | ISCM-Sys | Test | | | | |

#### Defect Check Rationale Table

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:[33]

| Determination Statement ID | Defect Check ID | Defect Check Name | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **requiring the developer of the information system, system component, or information system service to implement a verifiable flaw remediation process** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| SA-11(d){1} | VUL-F01 | Vulnerable Software | …the presence of software vulnerabilities (CVEs or equivalent). |

[33] Because control item SA-11(d) is focused on the flaw remediation *process* of the system developer, organizations requiring additional assurance may wish to supplement the automated assessment method *test* with manual assessment methods *examine* and *interview* at an organization-defined frequency.

| **Determination Statement ID** | **Defect Check ID** | **Defect Check Name** | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **requiring the developer of the information system, system component, or information system service to implement a verifiable flaw remediation process** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| SA-11(d){1} | VUL-L01 | Poor coding practices | …the presence of software with poor coding practices (CWEs or equivalent). |

### 3.3.4.4 *Control Item SI-2(2): FLAW REMEDIATION | AUTOMATED FLAW REMEDIATION STATUS*

**Control Item Text**

The organization employs automated mechanisms [Assignment: organization-defined frequency] to determine the state of information system components with regard to flaw remediation.

**Determination Statement 1**

| Determination Statement ID | Determination Statement Text |
|---|---|
| SI-2(2){1} | Determine if the organization: employs automated mechanisms [Assignment: organization-defined frequency] to determine the state of system components with regard to flaw remediation. |

**Roles and Assessment Methods**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| SI-2(2){1} | ISCM-Ops | ISCM-TN | ISCM-Sys | Test | | | | |

**Defect Check Rationale Table**

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| Determination Statement ID | Defect Check ID | Defect Check Name | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **employing automated mechanisms [Assignment: organization-defined frequency] to determine the state of information system components with regard to flaw remediation** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| SI-2(2){1} | VUL-F01 | Vulnerable Software | …the presence of software vulnerabilities (CVEs or equivalent) |
| SI-2(2){1} | VUL-L01 | Poor coding practices | …the presence of software with poor coding practices (CWEs or equivalent) |
| SI-2(2){1} | VUL-Q03 | Low completeness metric | …completeness of overall ISCM reporting not meeting the threshold |
| SI-2(2){1} | VUL-Q04 | Poor timeliness metric | …poor timeliness of overall ISCM reporting |

### 3.3.5 High Baseline Security Control Item Narratives

#### *3.3.5.1 Control Item SI-2(2): FLAW REMEDIATION | AUTOMATED FLAW REMEDIATION STATUS*

**Control Item Text**

The organization centrally manages the flaw remediation process.

**Determination Statement 1**

| Determination Statement ID | Determination Statement Text |
|---|---|
| SI-2(1){1} | Determine if the organization: centrally manages the flaw remediation process. |

**Roles and Assessment Methods**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| SI-2(1){1} | SWFM | ISCM-TN | ISCM-Sys | Test | | | | |

**Defect Check Rationale Table**

A failure in effectiveness of this control item results in a defect in one or more of the following defect checks:

| Determination Statement ID | Defect Check ID | Defect Check Name | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **centrally managing the flaw remediation process** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| SI-2(1){1} | VUL-F01 | Vulnerable Software | …the presence of software vulnerabilities (CVEs or equivalent). |
| SI-2(1){1} | VUL-L01 | Poor coding practices | …the presence of software with poor coding practices (CWEs or equivalent). |
| SI-2(1){1} | VUL-Q01 | Non-reporting devices | …a device failing to report software vulnerabilities within the specified time frame. |
| SI-2(1){1} | VUL-Q02 | Non-reporting applicable defect checks | …applicable defect checks failing to report. |

| **Determination Statement ID** | **Defect Check ID** | **Defect Check Name** | **Rationale** If an [organization-defined measure] for this defect check is above [the organization-defined threshold], then defects in **centrally managing the flaw remediation process** related to this control item might be the cause of the defect; i.e., ... |
|---|---|---|---|
| SI-2(1){1} | VUL-Q03 | Low completeness metric | …completeness of overall ISCM reporting not meeting the threshold. |
| SI-2(1){1} | VUL-Q04 | Poor timeliness metric | …poor timeliness of overall ISCM reporting. |

## 3.4 Control Allocation Tables (CATs)

Table 8: Low Baseline Control (Item) Allocation Table, Table 9: Moderate Baseline Control (Item) Allocation Table, and Table 10: High Baseline Control (Item) Allocation Table provide the low, moderate, and high baseline control allocation tables, respectively. The following is a summary of the material in the security plan assessment narrative for each determination statement in Section 3.3. It provides a concise summary of the assessment plan.

### 3.4.1 Low Baseline Control Allocation Table

**Table 8: Low Baseline Control (Item) Allocation Table**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| RA-5(a){1} | ISCM-Ops | ISCM-TN | ISCM-Sys | Test | | | | |
| RA-5(a){2} | DSM | ISCM-TN | ISCM-Sys | Test | | | | |
| RA-5(b){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | |
| RA-5(b){2} | ISCM-Ops | ISCM-TN | ISCM-Sys | Test | | | | |
| RA-5(b){3} | MAN | ISCM-TN | MAN | TBD | | | | |
| RA-5(c){1} | RskEx | ISCM-TN | ISCM-Sys | Test | | | | |
| RA-5(d){1} | PatMan | ISCM-TN | ISCM-Sys | Test | | | | |
| RA-5(e){1} | RskEx | ISCM-TN | ISCM-Sys | Test | | | | |
| SI-2(a){1} | SWFM | ISCM-TN | ISCM-Ops | Test | | | | |
| SI-2(a){2} | PatMan | ISCM-TN | ISCM-Sys | Test | | | | |
| SI-2(b){1} | MAN | ISCM-TN | MAN | TBD | | | | |
| SI-2(c){1} | PatMan | ISCM-TN | ISCM-Sys | Test | | | | |
| SI-2(d){1} | SWFM | ISCM-TN | ISCM-Sys | Test | | | | |

### 3.4.2 Moderate Baseline Control Allocation Table

**Table 9: Moderate Baseline Control (Item) Allocation Table**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| RA-5(1){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | |
| RA-5(2){1} | ISCM-Ops | ISCM-TN | ISCM-Sys | Test | | | | |
| SA-11(d){1} | SWFM | ISCM-TN | ISCM-Sys | Test | | | | |
| SI-2(2){1} | ISCM-Ops | ISCM-TN | ISCM-Sys | Test | | | | |

### 3.4.3 High Baseline Control Allocation Table

**Table 10: High Baseline Control (Item) Allocation Table**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing |
|---|---|---|---|---|---|---|---|---|
| SI-2(1){1} | SWFM | ISCM-TN | ISCM-Sys | Test | | | | |

## References

| | |
|---|---|
| [CNA] | The MITRE Corporation (2019) CVE Numbering Authorities. Available at: https://cve.mitre.org/cve/cna.html |
| [CNSSI 4009] | Committee on National Security Systems (2015) Committee on National Security Systems (CNSS) Glossary. (National Security Agency, Fort George G. Meade, MD), CNSS Instruction 4009. Available at https://www.cnss.gov/CNSS/issuances/Instructions.cfm |
| [CPE] | National Institute of Standards and Technology (2020) Common Platform Enumeration. Available at: https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/cpe/ |
| [CVE] | The MITRE Corporation (2019) Common Vulnerabilities and Exposures (CVE). Available at: https://cve.mitre.org |
| [CVENVD] | The MITRE Corporation (2019) CVE and NVD Relationship. Available at: https://cve.mitre.org/about/cve_and_nvd_relationship.html |
| [CVSS] | First.org, Inc (2020) Common Vulnerability Scoring System Special Interest Group (CVSS SIG). Available at: https://www.first.org/cvss/ |
| [CWE] | The MITRE Corporation (2019) Common Weakness Enumeration. Available at: https://cwe.mitre.org |
| [FIPS199] | National Institute of Standards and Technology (2004) Standards for Security Categorization of Federal Information and Information Systems. (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) 199. https://doi.org/10.6028/NIST.FIPS.199 |
| [IR7511] | Cook MR, Quinn SD, Waltermire DA, Prisaca D (2016) Security Content Automation Protocol (SCAP) Version 1.2 Validation Program Test Requirements. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 7511, Rev. 4. https://doi.org/10.6028/NIST.IR.7511r4 |
| [IR8011-1] | Dempsey KL, Eavy P, Moore G (2017) Automation Support for Security Control Assessments: Volume 1: Overview. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 8011, Vol. 1. https://doi.org/10.6028/NIST.IR.8011-1 |
| [IR8011-3] | Dempsey KL, Goren N, Eavy P, Moore G (2018) Automation Support for Security Control Assessments: Software Asset Management. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 8011, Vol. 3. https://doi.org/10.6028/NIST.IR.8011-3 |
| [IR8060] | Waltermire D., et al (2016), Guidelines for the Creation of Interoperable |

Software Identification (SWID) Tags (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency Report (NISTIR) 8060, https://csrc.nist.gov/publications/detail/nistir/8060/final

[NVD] National Institute of Standards and Technology (2019) National Vulnerability Database. Available at: https://nvd.nist.gov

[SEI] Householder, A.D., Wassermann, G., Manion, A., & King, C. (2017). The CERT Guide to Coordinated Vulnerability Disclosure. (Carnegie Mellon University Software Engineering Institute, Pittsburgh, PA), available at: https://resources.sei.cmu.edu/asset_files/SpecialReport/2017_003_001_503340.pdf

[SP800-30] Joint Task Force Transformation Initiative (2012) Guide for Conducting Risk Assessments. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-30, Rev. 1. https://doi.org/10.6028/NIST.SP.800-30r1

[SP800-37] Joint Task Force (2018) Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-37, Rev. 2. https://doi.org/10.6028/NIST.SP.800-37r2

[SP800-39] Joint Task Force Transformation Initiative (2011) Managing Information Security Risk: Organization, Mission, and Information System View. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-39. https://doi.org/10.6028/NIST.SP.800-39

[SP800-53] Joint Task Force Transformation Initiative (2013) Security and Privacy Controls for Federal Information Systems and Organizations. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-53, Rev. 4, Includes updates as of January 22, 2015. https://doi.org/10.6028/NIST.SP.800-53r4

[SP800-53A] Joint Task Force Transformation Initiative (2014) Assessing Security and Privacy Controls in Federal Information Systems and Organizations: Building Effective Assessment Plans. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-53A, Rev. 4, Includes updates as of December 18, 2014. https://doi.org/10.6028/NIST.SP.800-53Ar4

[SP800-60-v1] Stine KM, Kissel RL, Barker WC, Fahlsing J, Gulick J (2008) Guide for Mapping Types of Information and Information Systems to Security Categories. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-60, Vol. 1, Rev. 1. https://doi.org/10.6028/NIST.SP.800-60v1r1

[SP800-126] Waltermire DA, Quinn SD, Scarfone KA, Halbardier AM (2011) The

Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.2. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-126, Rev. 2, Includes updates as of March 19, 2012. https://doi.org/10.6028/NIST.SP.800-126r2

[SP800-163] Ogata MA, Franklin JM, Voas JM, Sritapan V, Quirolgico S (2019) Vetting the Security of Mobile Applications. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-163, Rev. 1. https://doi.org/10.6028/NIST.SP.800-163r1

## Appendix A Traceability of VUL Control Items to Example Attack Steps

*Note*: This Appendix includes only those control items that can be assessed (at least in part) via automation.

| Example Attack Step | NIST SP 800-53 Control Item Code |
|---|---|
| 2) Initiate Attack Internally | RA-5(b) |
| 2) Initiate Attack Internally | RA-5(c) |
| 2) Initiate Attack Internally | RA-5(d) |
| 2) Initiate Attack Internally | RA-5(e) |
| 2) Initiate Attack Internally | SA-11(d) |
| 2) Initiate Attack Internally | SI-2(a) |
| 2) Initiate Attack Internally | SI-2(c) |
| 2) Initiate Attack Internally | SI-2(d) |
| 2) Initiate Attack Internally | SI-2(1) |
| 5) Expand Control – Escalate or Propagate | RA-5(b) |
| 5) Expand Control – Escalate or Propagate | RA-5(c) |
| 5) Expand Control – Escalate or Propagate | RA-5(d) |
| 5) Expand Control – Escalate or Propagate | RA-5(e) |
| 5) Expand Control – Escalate or Propagate | SA-11(d) |
| 5) Expand Control – Escalate or Propagate | SI-2(a) |
| 5) Expand Control – Escalate or Propagate | SI-2(c) |
| 5) Expand Control – Escalate or Propagate | SI-2(d) |
| 5) Expand Control – Escalate or Propagate | SI-2(1) |

## Appendix B Keyword Rules Used to Identify Controls that Support VUL

Automated keyword searches were employed to identify candidate control items in NIST SP 800-53 that might support the VUL capability. After candidate controls were returned by the keyword searches, the language content of each control item was examined manually to separate those that support the VUL capability (true positives) from those that do not (false positives). The control items for the low, moderate, and high baselines are listed in Tables 8, 9, and 10, respectively. The specific keyword rules used to identify VUL controls appear in the table below.

| Keyword Rule | Rationale |
|---|---|
| *flaw remediation* | Ensuring that flaws (CWEs) are found and corrected prior to approval and periodically thereafter |
| *high-risk areas* | Ensuring that software moving to high risk areas is adequately patched for the new location or environment |
| *non-persisten* OR *persisten* | Ensuring that software is loaded from persistent and trusted sources which have already had flaws removed and have been patched |
| *vulnerabil* AND *scan* | Ensuring that software vulnerabilities are identified and corrected |

## Appendix C Control Items in the Low-High Baseline that were Selected by the Keyword Search for Controls that Support VUL, but were Manually Determined to be False Positives

| NIST SP 800-53 Control Item | Control Text | Level | Rationale for Calling a False Positive |
|---|---|---|---|
| AU-6 (5) | AUDIT REVIEW, ANALYSIS, AND REPORTING \| INTEGRATION / SCANNING AND MONITORING CAPABILITIES<br>The organization integrates analysis of audit records with analysis of [Selection (one or more): vulnerability scanning information; performance data; information system monitoring information; [Assignment: organization-defined data/information collected from other sources]] to further enhance the ability to identify inappropriate or unusual activity. | High | Relates to audit record analysis (not the VUL capability) |
| CA-2 (2) | SECURITY ASSESSMENTS \| SPECIALIZED ASSESSMENTS<br>The organization includes, as part of security control assessments, [Assignment: organization-defined frequency], [Selection: announced. unannounced], [Selection (one or more): in-depth monitoring; vulnerability scanning; malicious user testing; insider threat assessment; performance/load testing; [Assignment: organization-defined other forms of security assessment]]. | High | Relates to assessment capability |
| RA-5 (4) | VULNERABILITY SCANNING \| DISCOVERABLE INFORMATION<br>The organization determines what information about the information system is discoverable by adversaries and subsequently takes [Assignment: organization-defined corrective actions]. | High | Does not relate to removing software vulnerabilities |
| RA-5 (5) | VULNERABILITY SCANNING \| PRIVILEGED ACCESS<br>The information system implements privileged access authorization to [Assignment: organization-identified information system components] for selected [Assignment: organization-defined vulnerability scanning activities]. | Moderate | Relates to access/trust capability |

# Appendix D Control Items Not in the Low, Moderate, or High Baselines

The following security controls/control items are not included in an NIST SP 800-53 baseline and were therefore not analyzed further after the keyword search:

- Program Management (PM) Family because the PM controls do not apply to individual systems
- Controls/control items selected by the VUL keywords (as specified in Appendix B) that are not assigned to an NIST SP 800-53 baseline
- Privacy Controls.

The controls/control items matching the criteria in the bulleted list above are provided in this appendix in case an organization wants to develop its own automated tests.

| NIST SP 800-53 Control/Control Item | Control Text |
|---|---|
| RA-5(3) | VULNERABILITY SCANNING \| BREADTH / DEPTH OF COVERAGE<br>The organization employs vulnerability scanning procedures that can identify the breadth and depth of coverage (i.e., information system components scanned and vulnerabilities checked). |
| RA-5(6) | VULNERABILITY SCANNING \| AUTOMATED TREND ANALYSES<br>The organization employs automated mechanisms to compare the results of vulnerability scans over time to determine trends in information system vulnerabilities. |
| RA-5(8) | VULNERABILITY SCANNING \| REVIEW HISTORIC AUDIT LOGS<br>The organization reviews historic audit logs to determine if a vulnerability identified in the information system has been previously exploited. |
| RA-5(10) | VULNERABILITY SCANNING \| CORRELATE SCANNING INFORMATION<br>The organization correlates the output from vulnerability scanning tools to determine the presence of multi-vulnerability/multi-hop attack vectors. |
| SC-34(1) | NON-MODIFIABLE EXECUTABLE PROGRAMS \| NO WRITABLE STORAGE<br>The organization employs [Assignment: organization-defined information system components] with no writeable storage that is persistent across component restart or power on/off. |

| NIST SP 800-53 Control/Control Item | Control Text |
|---|---|
| SI-2(3)(a) | FLAW REMEDIATION \| TIME TO REMEDIATE FLAWS / BENCHMARKS FOR CORRECTIVE ACTIONS<br>The organization:<br>(a) Measures the time between flaw identification and flaw remediation. |
| SI-2(3)(b) | FLAW REMEDIATION \| TIME TO REMEDIATE FLAWS / BENCHMARKS FOR CORRECTIVE ACTIONS<br>The organization:<br>(b) Establishes [Assignment: organization-defined benchmarks] for taking corrective actions. |
| SI-2(5) | FLAW REMEDIATION \| AUTOMATIC SOFTWARE / FIRMWARE UPDATES<br>The organization installs [Assignment: organization-defined security-relevant software and firmware updates] automatically to [Assignment: organization-defined information system components]. |
| SI-2(6) | FLAW REMEDIATION \| REMOVAL OF PREVIOUS VERSIONS OF SOFTWARE / FIRMWARE<br>The organization removes [Assignment: organization-defined software and firmware components] after updated versions have been installed. |
| SI-3(10)(b) | MALICIOUS CODE PROTECTION \| MALICIOUS CODE ANALYSIS<br>The organization:<br>(b) Incorporates the results from malicious code analysis into organizational incident response and flaw remediation processes. |
| SI-14 | NON-PERSISTENCE<br>Control: The organization implements non-persistent [Assignment: organization-defined information system components and services] that are initiated in a known state and terminated [Selection (one or more): upon end of session of use; periodically at [Assignment: organization-defined frequency]]. |
| SI-14(1) | NON-PERSISTENCE \| REFRESH FROM TRUSTED SOURCES<br>The organization ensures that software and data employed during information system component and service refreshes are obtained from [Assignment: organization-defined trusted sources]. |

## Appendix E VUL-Specific Acronyms and Abbreviations

| | |
|---|---|
| API | Application Programming Interface |
| CVE | Common Vulnerability and Exposure |
| CWE | Common Weakness Enumeration |
| SWID Tag | Software Identification Tag |

## Appendix F Glossary

| | |
|---|---|
| **common vulnerabilities and exposures (CVE)** [SP800-126] | A nomenclature and dictionary of security-related software flaws. |
| **common vulnerabilities and exposures (CVE)** [CVENVD] | A list of entries, each containing a unique identification number, a description, and at least one public reference—for publicly known cybersecurity vulnerabilities [CVENVD]. This list feeds the National Vulnerability Database (NVD).<br><br>See also: CVE equivalent. |
| **CVE equivalent** | A vulnerability—known by someone—that has been found in specific software—irrespective of whether that vulnerability is publicly known. CVEs are a subset of CVE equivalents. |
| **common weakness enumeration (CWE)** [CWE] | A list of known poor coding practices that may be present in software [CWE].<br><br>See also, weakness. |
| **common weakness enumeration (CWE)** [CNSSI 4009] | A taxonomy for identifying the common sources of software flaws (e.g., buffer overflows, failure to check input data). |
| **dynamic code analyzer** | A tool that analyzes computer software by executing programs built from the software being analyzed on a real or virtual processor and observing its behavior, probing the application and analyzing application responses. |
| **metacontrol** | A control of, or about, a control. For example, a control that specifies how the desired or actual state data for another control is to be managed. |
| **national vulnerability database (NVD)** [IR7511] | The U.S. government repository of standards-based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data informs automation of vulnerability management, security measurement, and compliance. NVD includes databases of security checklists, security related software flaws, misconfigurations, product names, and impact metrics. |
| **package management system** | An administrative tool or utility that facilitates the installation and maintenance of software on a given host, device or pool of centrally managed hosts, and the reporting of installed software attributes. May also be referred to as package manager, software manager, application manager, or app manager. |
| **package manifest** | A listing of the contents of a software package. |
| **patch level** | Denotes either a patch level or a patch set. More specifically, when patches must be applied in order, the patch level is the identifier of the most recently applied patch. |
| **patch set** | When patches do not need to be applied in any particular order, the patch set includes all (and only) the applied patches. |

| | |
|---|---|
| **software product and executable file version** | A patch level versioning of the software product or digital fingerprint version of a software file. |
| **software vulnerability** [SP800-163, Adapted] | A security flaw, glitch, or weakness found in software code that could be exploited by an attacker (threat source). |
| **static code analyzer** | A tool that analyzes source code without executing the code. Static code analyzers are designed to review bodies of source code (at the programming language level) or compiled code (at the machine language level) to identify poor coding practices. Static code analyzers provide feedback to developers during the code development phase on security flaws that might be introduced into code. |
| **vulnerability** [CNSSI 4009] | Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited by a threat source. |
| **vulnerability scanner** | (As used in this volume) A network tool (hardware and/or software) that scans network devices to identify generally known and organization specific CVEs. It may do this based on a wide range of signature strategies. |
| **vulnerability scanner** | A tool (hardware and/or software) used to identify hosts/host attributes and associated vulnerabilities (CVEs, CWEs, and others). |
| **weakness** | (As used in this volume) Poor coding practices, as exemplified by CWEs. |

## Appendix G Control Items Affecting Desired and/or Actual State from All Defect Checks in this Volume

This table supports:

- Identification of controls necessary to ensure that both the actual state and desired state data are maintained under effective configuration management in order to support complete, timely, and valid testing.

- Root cause analysis when a specific defect check fails. Such a failure might be caused not only by a failure of the specific control items mapped to that defect check in the defect check narratives but also by a failure in any of the listed control items.

As used here, the controls apply to potential defects in the desired state (DS) and/or actual state (AS). The rationale column explains how a defect in the control item might cause the defect check to fail.

For example, in the vulnerability management capability, suppose an organization has identified a set of vulnerabilities to be checked that is recorded in both the desired state metadata and the tool used to perform the check. The organization can then compare the desired state and the tool used to perform the check to make sure that the vulnerability "checking process" is complete. However, if the desired state data itself is not under effective configuration management, some of the vulnerability checks might be removed from the desired state checking process due to an insider threat, carelessness, or an external attack by someone who wants to exploit a particular vulnerability. If the desired state metadata is under effective configuration management, the disparity in the desired state can be found quickly. Otherwise, the removal of vulnerability checks might not be discovered until root cause analysis after a successful attack (assuming the attack is even discovered).

*Note*: These items are not explicitly included in the control item assessment narratives unless they also apply to the configuration management of items *other than the desired and actual states* for assessment.

| Determination Statement ID | Determination Statement Text | Impact Level | Affects DS and/or AS | Rationale |
|---|---|---|---|---|
| CM-2{1} | Determine if the organization: develops, documents, and maintains a current baseline configuration of the information system under configuration control. | Low | DS | Otherwise, there is no desired state for testing. |
| CM-2(1)(a){1} | Determine if the organization: reviews and updates the baseline configuration of the information system:<br>(a) [Assignment: organization-defined frequency]. | Moderate | DS | Otherwise, the desired state might not be updated as needed to maintain appropriate security. |
| CM-2(1)(b){1} | Determine if the organization: reviews and updates the baseline configuration of the information system:<br>(b) When required due to [Assignment organization-defined circumstances]. | Moderate | DS | Otherwise, desired state might not be updated based on the organization-defined circumstances. |
| CM-2(1)(c){1} | Determine if the organization: reviews and updates the baseline configuration of the information system:<br>(c) As an integral part of information system component installations and upgrades. | Moderate | DS | Otherwise, desired state might not be updated as appropriate when component installations and updates occur. |
| CM-2(2){1} | Determine if the organization: employs automated mechanisms to maintain an up-to-date, complete, accurate, and readily available baseline configuration of the information system. | High | DS | Otherwise, accurate testing information might not be provided. |
| CM-3(a){1} | Determine if the organization: employs automated mechanisms to determine the types of changes to the system {installed software} that are configuration-controlled. | Moderate | DS | Otherwise, the desired state might not specify all machine-readable data needed for implemented defect checks. |
| CM-3(b){1} | Determine if the organization: reviews proposed configuration-controlled changes to the {software of the} system and approves or disapproves such changes. | Moderate | DS | Otherwise, the decisions on desired state might not adequately reflect security impact of changes. |

| Determination Statement ID | Determination Statement Text | Impact Level | Affects DS and/or AS | Rationale |
|---|---|---|---|---|
| CM-3(b){2} | Determine if the organization: explicitly considers security impact analysis when reviewing proposed configuration-controlled changes to the {software of the} system. | Moderate | DS | Otherwise, the decisions on desired state might not adequately reflect security impact of changes. |
| CM-3(c){1} | Determine if the organization: documents configuration change decisions associated with the system {installed software}. | Moderate | DS | Otherwise, changes to the desired state specification might not be documented and available as machine-readable data. |
| CM-3(d){1} | Determine if the organization: implements approved configuration-controlled changes to the system {installed software}. | Moderate | AS | Otherwise, defect checks might fail because changes were not implemented in the actual state. |
| CM-3(f){1} | Determine if the organization: audits activities associated with configuration-controlled changes to the {software of the} system. | Moderate | DS | Otherwise, errors in the desired state might not be detected. |
| CM-3(f){2} | Determine if the organization: reviews activities associated with configuration-controlled changes to the {software of the} system. | Moderate | DS | Otherwise, errors in the desired state might not be detected. |
| CM-3(g){1} | Determine if the organization: coordinates configuration change control activities {of software} through [Assignment: organization-defined configuration change control element (e.g., committee, board)] that convenes [Selection (one or more): [Assignment: organization-defined frequency]; [Assignment: organization-defined configuration change conditions]. | Moderate | DS | Otherwise, the persons authorized to make change approval decisions, and the scope of their authority might not be clearly defined to enable knowing what decisions are authorized. |
| CM-3(g){2} | Determine if the organization: provides oversight for configuration change control activities {of software} through [Assignment: organization-defined configuration change control element (e.g., committee, board)] that convenes [Selection (one or more): [Assignment: organization-defined frequency]; [Assignment: organization-defined configuration change conditions]. | Moderate | DS | Otherwise, the persons authorized to make change approval decisions and the scope of their authority might not be clearly defined to enable knowing what decisions are authorized. |

| Determination Statement ID | Determination Statement Text | Impact Level | Affects DS and/or AS | Rationale |
|---|---|---|---|---|
| CM-3(1)(a){1} | Determine if the organization: employs automated mechanisms to document proposed changes to the system {installed software}. | High | DS | Otherwise, changes to the desired state specification might not be documented and available for assessment. |
| CM-3(1)(b){1} | Determine if the organization: employs automated mechanisms to notify [Assignment: organized-defined approval authorities] of proposed changes to the system {installed software} and request change approval. | High | DS | Otherwise, needed changes might not be reviewed in a timely manner. |
| CM-3(1)(c){1} | Determine if the organization: employs automated mechanisms to highlight proposed changes to the system {installed software} that have not been approved or disapproved by [Assignment: organization-defined time period]. | High | DS | Otherwise, needed changes might not be reviewed in a timely manner. |
| CM-3(1)(d){1} | Determine if the organization: employs automated mechanisms to prohibit changes to the system {installed software} until designated approvals are received. | High | DS | Otherwise, unapproved changes might be implemented. |
| CM-3(1)(e){1} | Determine if the organization: employs automated mechanisms to document all changes to the system {installed software}. | High | AS | Otherwise, documented changes might not reflect the actual state of the system. |
| CM-3(1)(f){1} | Determine if the organization: employs automated mechanisms to notify [Assignment: organization-defined personnel] when approved changes to the system {installed software} are completed. | High | DS | Otherwise, required changes might be missed. |
| CM-3(2){1} | Determine if the organization: tests, validates, and documents changes to the {software of the} system before implementing the changes on the operational system.<br>Not applicable in the operational environment.<br>This should be assessed via manual reauthorization prior to placing policy in the desired state. Because it occurs as part of system engineering, it is outside of the scope of this operational capability. | Moderate | DS and AS | Otherwise, changes might increase risk by creating operational or security defects. |

| Determination Statement ID | Determination Statement Text | Impact Level | Affects DS and/or AS | Rationale |
|---|---|---|---|---|
| CM-8(a){1} | Determine if the organization: develops and documents an inventory of system components {for software} that (1) accurately reflects the current system and (2) includes all components within the authorization boundary of the system. | Low | DS and AS | Otherwise, the desired state and actual state inventories might have errors related to accuracy, completeness, and/or content. |
| CM-8(a){2} | Determine if the organization: develops and documents an inventory of system components {for software} that is at the level of granularity deemed necessary for tracking and reporting [by the organization]. | Low | DS and AS | Otherwise, the desired state and actual state inventories might have errors related to level of detail. |
| CM-8(b){1} | Determine if the organization: updates the system component inventory {for software} [Assignment: organization-defined frequency]. | Low | DS and AS | Otherwise, defects in the desired state and actual state inventories as well as related processes, might not be detected. |
| CM-8(b){2} | Determine if the organization: reviews the system component inventory {for software} [Assignment: organization-defined frequency]. | Low | DS and AS | Otherwise, defects in the desired state and actual state inventories as well as related processes might not be detected. |
| CM-8(1){1} | Determine if the organization: updates the inventory of system {installed software} components as an integral part of component installations, removals, and system updates. | Moderate | DS and AS | Otherwise, defects in desired state and actual state inventories as well as related processes might not be detected. |
| CM-8(2){1} | Determine if the organization: employs automated mechanisms to help maintain an up-to-date, complete, accurate, and readily available inventory of system {installed software} components. | High | DS and AS | Otherwise, an up-to-date and accurate desired state and actual state inventories might not be available for automated assessment. |
| CM-8(3)(a){1} | Determine if the organization: employs automated mechanisms [Assignment: organization-defined frequency] to detect the presence of unauthorized software and firmware components within the system. | Moderate | AS | Otherwise, inventory accuracy (e.g., completeness and timeliness) might be difficult or impossible to maintain. |

| Determination Statement ID | Determination Statement Text | Impact Level | Affects DS and/or AS | Rationale |
|---|---|---|---|---|
| CM-8(3)(b){1} | Determine if the organization: takes the following actions when unauthorized {installed software} components are detected: [Selection (one or more): disables network access by such components; isolates the components; notifies [Assignment: organization-defined personnel or roles]]. | Moderate | AS | Otherwise, detected security defects might not be mitigated. |
| CM-8(4){1} | Determine if the organization: includes in the {installed software} system component inventory information, a means for identifying by [Selection (one or more): name; position; role], individuals responsible/accountable for administering those components. | High | DS | Otherwise, when defects are detected, the automated systems cannot know what persons or groups to notify to take appropriate action. |

**Control Allocation Table for Appendix G**

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing | Level |
|---|---|---|---|---|---|---|---|---|---|
| CM-2{1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | Low |
| CM-2(1)(a){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |
| CM-2(1)(b){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |
| CM-2(1)(c){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |
| CM-2(2){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | High |
| CM-3(a){1} | DSM | ISCM-TN | MAN | TBD | | | | | Moderate |
| CM-3(b){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |
| CM-3(b){2} | DSM | ISCM-TN | MAN | TBD | | | | | Moderate |
| CM-3(c){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |
| CM-3(d){1} | PatMan | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |
| CM-3(f){1} | ISCM-Sys | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |
| CM-3(f){2} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |

| Determination Statement ID | Implemented By | Assessment Boundary | Assessment Responsibility | Assessment Methods | Selected | Rationale for Risk Acceptance | Frequency of Assessment | Impact of Not Implementing | Level |
|---|---|---|---|---|---|---|---|---|---|
| CM-3(g){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |
| CM-3(g){2} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |
| CM-3(1)(a){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | High |
| CM-3(1)(b){1} | ISCM-Sys | ISCM-TN | ISCM-Sys | Test | | | | | High |
| CM-3(1)(c){1} | ISCM-Sys | ISCM-TN | ISCM-Sys | Test | | | | | High |
| CM-3(1)(d){1} | ISCM-Sys | ISCM-TN | ISCM-Sys | Test | | | | | High |
| CM-3(1)(e){1} | ISCM-Sys | ISCM-TN | MAN | TBD | | | | | High |
| CM-3(1)(f){1} | ISCM-Sys | ISCM-TN | ISCM-Sys | Test | | | | | High |
| CM-3(2){1} | DSM | ISCM-TN | MAN | TBD | | | | | Moderate |
| CM-8(a){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | Low |
| CM-8(a){2} | ISCM-Sys | ISCM-TN | ISCM-Sys | Test | | | | | Low |
| CM-8(b){1} | ISCM-Sys | ISCM-TN | ISCM-Sys | Test | | | | | Low |
| CM-8(b){2} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | Low |
| CM-8(1){1} | ISCM-Sys | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |
| CM-8(2){1} | ISCM-Sys | ISCM-TN | ISCM-Sys | Test | | | | | High |
| CM-8(3)(a){1} | ISCM-Sys | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |
| CM-8(3)(b){1} | PatMan | ISCM-TN | ISCM-Sys | Test | | | | | Moderate |
| CM-8(4){1} | DSM | ISCM-TN | ISCM-Sys | Test | | | | | High |