NIST Special Publication
NIST SP 800-40r4

Guide to Enterprise Patch Management Planning:

Preventive Maintenance for Technology

Murugiah Souppaya
Karen Scarfone

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-40r4

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

NIST Special Publication
NIST SP 800-40r4

Guide to Enterprise Patch Management Planning:

Preventive Maintenance for Technology

Murugiah Souppaya
Computer Security Division
Information Technology Laboratory

Karen Scarfone
Scarfone Cybersecurity
Clifton, VA

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-40r4

April 2022

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
James K. Olthoff, Performing the Non-Exclusive Functions and Duties of the Under Secretary of Commerce for Standards and Technology & Director, National Institute of Standards and Technology

Authority

National Institute of Standards and Technology Special Publication 800-40 Rev. 4
Natl. Inst. Stand. Technol. Spec. Publ. 800-40r4, 28 pages (April 2022)
CODEN: NSPUE2

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-40r4

Submit comments on this publication to: cyberhygiene@nist.gov

National Institute of Standards and Technology
Attn: Applied Cybersecurity Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 2000) Gaithersburg, MD 20899-2000

Reports on Computer Systems Technology

Abstract

Enterprise patch management is the process of identifying, prioritizing, acquiring, installing, and verifying the installation of patches, updates, and upgrades throughout an organization. Patching is more important than ever because of the increasing reliance on technology, but there is often a divide between business/mission owners and security/technology management about the value of patching. This publication frames patching as a critical component of preventive maintenance for computing technologies – a cost of doing business, and a necessary part of what organizations need to do in order to achieve their missions. This publication also discusses common factors that affect enterprise patch management and recommends creating an enterprise strategy to simplify and operationalize patching while also improving reduction of risk. Preventive maintenance through enterprise patch management helps prevent compromises, data breaches, operational disruptions, and other adverse events.

Keywords

enterprise patch management; patch; risk management; update; upgrade; vulnerability management.

Acknowledgments

The authors wish to thank their colleagues who reviewed the document and contributed to its technical content, especially Mark Simos from Microsoft, and Jamie Brown, Vincent Gilcreest, and Glen Pendley from Tenable. The authors also appreciate the efforts of all the individuals and organizations that provided feedback on the public comment draft, especially BlackBerry, the Cybersecurity & Infrastructure Security Agency (CISA), the National Security Agency (NSA), The SECOND Advisories, Carla Brinker, and Ron Colvin.

Audience

The primary audience for this publication is chief information officers, chief information security officers, cybersecurity directors and managers, and others who are responsible for managing organizational risk related to the use of software. Others with an interest in the topic, including business and mission owners, security engineers and architects, system administrators, and security operations personnel, may find portions of the publication to be informative.

Trademark Information

Patent Disclosure Notice

Executive Summary

Software used for computing technologies must be maintained because there are many in the world who continuously search for and exploit flaws in software. Software maintenance includes *patching*, which is the act of applying a change to installed software – such as firmware, operating systems, or applications – that corrects security or functionality problems or adds new capabilities. *Enterprise patch management* is the process of identifying, prioritizing, acquiring, installing, and verifying the installation of patches, updates, and upgrades throughout an organization.

In past perimeter-based security architectures, most software was operated on internal networks protected by several layers of network security controls. While patching was generally considered important for reducing the likelihood of compromise and was a common compliance requirement, patching was not always considered a priority. In today's environments, patching has become more important, often rising to the level of mission criticality. As part of a zero trust approach to security, it is now recognized that the perimeter largely does not exist anymore, and most technologies are directly exposed to the internet, putting systems at significantly greater risk of compromise. This dynamic applies across all computing technologies, whether they are information technology (IT), operational technology (OT), Internet of Things (IoT), mobile, cloud, virtual machine, container, or other types of assets. Zero trust architectures emphasize business asset-specific security over just protecting a network with assets on it, so patching is vital for reducing risk to those individual assets and determining the assets' trust status.

There is often a divide between business/mission owners and security/technology management. Business/mission owners may believe that patching negatively affects productivity, since it requires scheduled downtime for maintenance and introduces the risk of additional downtime if something goes wrong and disrupts operations. Leadership and business/mission owners should reconsider the priority of enterprise patch management in light of today's risks. Patching should be considered a standard cost of doing business and should be rigorously followed and tracked. Just as preventive maintenance on corporate fleet vehicles can help avoid costly breakdowns, patching should be viewed as a normal and necessary part of reliably achieving the organization's missions. If an organization needs a particular technology to support its mission, it also needs to maintain that technology throughout its life cycle – and that includes patching.

Leadership at all levels of the organization, business/mission owners, and security/technology management teams should jointly create an enterprise patch management strategy that simplifies and operationalizes patching while also improving its reduction of risk. This will strengthen organizational resiliency to active threats and minimize business and mission impacts. This publication provides recommendations for enterprise patch management planning.

Table of Contents

1 Introduction

1.1 Purpose and Scope

The purpose of this publication is to help organizations improve their enterprise patch management planning so that they can strengthen their management of risk. This publication strives to illustrate that enterprise patch management is preventive maintenance for an organization's technology. Adopting this mindset and following the recommendations and suggestions in this document should help organizations in the following ways:

- Security and technology management and leadership at all levels of the organization will gain a new understanding of the role of patching in enterprise risk management.
- The security/technology and business/mission sides of the organization will be able to communicate with each other more effectively regarding patch management and reach consensus on planning.
- Personnel from the security/technology and business/mission sides of the organization will be prepared to revamp their enterprise patching strategy throughout the entire patch management life cycle.

The discussion of patch management technologies is minimal in this publication. However, NIST Special Publication (SP) 1800-31 [1] provides information on using technologies to implement information technology (IT) patch management policies and processes. Also, Appendix A of SP 1800-31 Volume B discusses recommended practices for securing patch management technologies and the data they use.

This publication is intended to apply to all types of computing technology assets – including IT[1], operational technology (OT)[2], Internet of Things (IoT), mobile devices, and cloud computing – and to all types of patchable software – including applications, operating systems, and firmware – on those assets.

1.2 Changes from Previous Versions

This is the fourth version of NIST SP 800-40. The original SP 800-40, *Procedures for Handling Security Patches* (2002), provided basic information on patching procedures and sources of patch and vulnerability information. SP 800-40 Version 2.0, *Creating a Patch and Vulnerability Management Program* (2005), built on the original by adding content on processes, metrics, and common issues. Although SP 800-40 and SP 800-40 Version 2.0 are primarily of interest from a historical perspective, they address many of the same topics that organizations are still struggling with today.

[1] IT includes desktop and laptop computers, on-premises servers, networking equipment, printers, and other types of assets typically found in an organization's own offices or data centers.

[2] OT refers to "programmable systems or devices that interact with the physical environment (or manage devices that interact with the physical environment). These systems/devices detect or cause a direct change through the monitoring and/or control of devices, processes, and events. Examples include industrial control systems, building management systems, fire control systems, and physical access control mechanisms." [from NIST SP 800-37 Rev. 2]

The third version, SP 800-40, Revision 3, *Guide to Enterprise Patch Management Technologies* (2013), was written under the assumption that readers already understood the basics of patch management and that what they most needed help with was implementing, configuring, securing, and using enterprise patch management technologies. The latest SP 800-40 version is based on the assumption that, in the overall scope of enterprise patch management, organizations would benefit more from rethinking their patch management planning than their patch management technology. Readers who are particularly interested in enterprise patch management technologies may still benefit from content in Revision 3, although some of it is outdated and there are gaps in its coverage.

1.3 Publication Structure

The rest of this publication is organized into the following sections and appendices:

- Section 2 outlines possible risk response approaches for software vulnerabilities and provides a brief overview of the enterprise patch management life cycle.
- Section 3 presents a set of principles and actionable recommendations that support those principles for enterprise patch management planning.
- The References section defines the references cited throughout the publication.
- Appendix A lists the NIST Cybersecurity Framework Subcategories and the SP 800-53 controls that are most important for enterprise patch management policies and processes.
- Appendix B contains an acronym list.

2 Risk Response Approaches for Software Vulnerabilities

This section outlines possible risk response approaches for software vulnerabilities, provides an overview of the software vulnerability management life cycle, and takes a closer look at parts of that life cycle with respect to patching.

2.1 Risk Responses

Patching is one of several ways to respond to risks from software vulnerabilities. This publication references four types of *risk responses* [2]:

1. ***Accept***: Accept the risk from vulnerable software as is, such as by relying on existing security controls to prevent vulnerability exploitation or by determining that the potential impact is low enough that no additional action is needed.
2. ***Mitigate***: Reduce the risk by eliminating the vulnerabilities (e.g., patching the vulnerable software, disabling a vulnerable feature, or upgrading to a newer software version without the vulnerabilities) and/or deploying additional security controls to reduce vulnerability exploitation (e.g., using firewalls and network segmentation to isolate vulnerable assets, thus reducing the attack surface).
3. ***Transfer***: Reduce the risk by sharing some of the consequences with another party, such as by purchasing cybersecurity insurance or by replacing conventional software installations with software-as-a-service (SaaS) usage where the SaaS vendor/managed service provider takes care of patching.
4. ***Avoid***: Ensure that the risk does not occur by eliminating the attack surface, such as by uninstalling the vulnerable software, decommissioning assets with the vulnerabilities, or disabling computing capabilities in assets that can function without them.

By default, an organization accepts the risk posed by using its software. Software could have vulnerabilities in it at any time that the organization does not know about, and sometimes previously unknown vulnerabilities are exploited – a zero-day attack. Once a new vulnerability becomes publicly known, risk usually increases because attackers are more likely to develop exploits that target the vulnerable software.

Installing a patch or update or upgrading software to a newer version without the vulnerabilities are the only forms of risk response that can completely eliminate the vulnerabilities without removing functionality. However, immediately patching, updating, or upgrading vulnerable software is sometimes not viable. Examples of why include the following:

- A patch may not be available yet. For example, a vulnerability may be announced before a patch is ready, and it could be days, weeks, or months before the patch is released.
- The vendor may no longer support the vulnerable software, meaning that a patch for it will never be released because the software is at end-of-life.
- The organization may need to wait for a scheduled outage window, perform testing first, update other software that interacts with the software to be patched, or train employees on new features or interfaces.

- Some patches may be considered a higher priority, so other patches are delayed due to limited resources.
- The manufacturer may require customers to update the software on a delayed schedule, such as for assets with human safety implications in a highly regulated sector, because of the extensive testing and certification that must be performed first. In these cases, organizations that choose to implement updates on their own may be voiding the product warranty and preventing future support from the manufacturer.
- The organization may need to comply with specific legal, regulatory, or business requirements. For example, an organization may need to use Federal Information Processing Standards (FIPS)-validated cryptographic modules for protecting data, but the cryptographic modules in the upgraded software are not yet FIPS-validated.

Even when patching, updating, or upgrading vulnerable software is viable, organizations can choose to respond to the risk from the vulnerabilities in a different way, such as any of the other risk response examples at the beginning of this section.

2.2 Software Vulnerability Management Life Cycle

The following describes a basic software vulnerability management life cycle. This life cycle applies to all risk response approaches.

1. **Know when new software vulnerabilities affect your organization's assets, including applications, operating systems, and firmware.** This involves knowing what assets your organization uses and which software and software versions those assets run down to the level of packages and libraries, as well as keeping track of new vulnerabilities in that software. For example, your organization might subscribe to vulnerability feeds from software vendors, security researchers, and the National Vulnerability Database (NVD).
2. **Plan the risk response.** This involves assessing the risk the vulnerability poses to your organization, choosing which form of risk response (or combination of forms) to use, and deciding how to implement the risk response. For example, you might determine that risk is elevated because the vulnerability is present in many organization assets and is being exploited in the wild, then choose mitigation as the risk response and mitigate the vulnerability by upgrading the vulnerable software and altering the software's configuration settings.
3. **Execute the risk response.** This will vary depending on the nature of the selected risk response, but common phases include the following:
    a. **Prepare the risk response.** This encompasses any preparatory activities, such as acquiring, validating, and testing patches for the vulnerable software; deploying additional security controls to safeguard the vulnerable software; or acquiring a replacement for a legacy asset that cannot be patched. It might also include scheduling the risk response and coordinating deployment plans with enterprise change management, business units, and others.
    b. **Implement the risk response.** Examples of this include distributing and installing a patch, purchasing cybersecurity insurance, deploying additional security controls, and changing asset configurations and state (e.g., software reset,

platform reboot). Any issues that occur during implementation should be resolved.

c. **Verify the risk response.** This step involves ensuring that the implementation has been completed successfully. For patching, this means confirming that the patch is installed and has taken effect. For deploying additional security controls, ensure they are functioning as intended. For risk avoidance, verify that vulnerable assets were decommissioned or replaced.

d. **Continuously monitor the risk response.** Make sure that the risk response continues to be in place: no one uninstalls the patch, deactivates the additional security controls, lets the cybersecurity insurance lapse, or restarts the decommissioned asset.

In addition, there are administrative activities occurring throughout the software vulnerability management life cycle, such as updating documentation, audit logging, and generating actionable insights and reports as part of enterprise change management. Having robust change management policies and processes in place is a fundamental part of software vulnerability management.

2.3 Risk Response Execution

This section takes a closer look at the common phases of executing a risk response, as described in the previous subsection, specifically within the context of patching.

2.3.1 Prepare to Deploy the Patch

Examples of common steps for preparing to deploy a patch include the following (not necessarily in this order):

- **Prioritize the patch.** A patch may be a higher priority to deploy than others because its deployment would reduce cybersecurity risk more than other patches would. Another patch may be a lower priority because it addresses a low-risk vulnerability on a small number of low-importance assets.
- **Schedule patch deployment.** Many organizations schedule patch deployments as part of their enterprise change management activities.
- **Acquire the patch.** Patches may be downloaded from the internet, built internally by developers or system administrators, or provided through removable media.
- **Validate the patch.** A patch's authenticity and integrity should be confirmed, preferably by automated means, before the patch is tested or installed. The patch could have been acquired from a rogue source or tampered with in transit or after acquisition.
- **Test the patch.** A patch may be tested before deployment. This is intended to reduce operational risk by identifying problems with a patch before placing it into production. Testing may be performed manually or through automated methods.

2.3.2 Deploy the Patch

Patch deployment varies widely based on several factors, including:

- The type of software being updated (e.g., firmware, operating system [OS], application)
- The asset platform type (e.g., IT, OT, IoT, mobile, cloud, virtual machine [VM], containers)
- Platform traits, such as managed/unmanaged asset, on-premises or not, virtualized or not, and containerized or not
- Environmental limitations, such as network connectivity and bandwidth

Many aspects of patch deployment are dependent on patch management technologies, which are out of the scope of this publication. At a high level, examples of common steps for deploying a patch include the following:

- **Distribute the patch.** Distributing the patch to the assets that need to have it installed can be organization-controlled (and occur automatically, manually, or as scheduled) or vendor-controlled, such as delivered from the cloud.
- **Validate the patch.** As discussed in Section 2.3.1, a patch's authenticity and integrity should be confirmed before installation, preferably through automated means.
- **Install the patch.** Installation can occur in numerous ways, including automatically; manually when directed to do so by a user, administrator, vendor, or tool; as a result of other software being installed or updated; and through the replacement of removable media used by an asset. Some installations require administrator privileges, such as installing firmware patches for a system basic input/output system (BIOS).[3] Some patch installations require user participation or cooperation.
- **Change software configuration and state.** In some cases, making a patch take effect necessitates implementing changes. Examples include restarting patched software, rebooting the operating system or platform on which the patched software runs, redeploying the applications, or altering software configuration settings. In other cases, no such changes are needed.
- **Resolve any issues.** Installing a patch may cause side effects to occur, like inadvertently altering existing security configuration settings or adding new settings, and these side effects can inadvertently create a new security problem while fixing the original one. Patch installation can also cause operational issues that may necessitate uninstalling the patch, reverting to the previous version of the software, or restoring the software or asset from backups.

[3] See NIST SP 800-147, *BIOS Protection Guidelines* (2011), for additional information on BIOS updates.

2.3.3 Verify Deployment

A patch's deployment can be verified to ensure that it has been installed successfully and taken effect. The robustness of verification can vary a great deal and is largely dependent on an organization's needs, but automated means are generally needed to achieve verification at scale.

2.3.4 Monitor the Deployed Patches

In the last phase of the life cycle, the patch's deployment can be monitored using automation to confirm that the patch is still installed. For example, monitoring could confirm that the patch has not been uninstalled by a user or an attacker, an unpatched version of the software has not been restored from a backup, and the device has not been reset to a vulnerable factory-default state.

Another reason for monitoring the deployed patches is to see if the patched software's behavior changes after patching. As part of a layered security approach to mitigating supply chain risk, this might be helpful at detecting, responding to, and recovering from situations where the installed patch was itself compromised.

3 Recommendations for Enterprise Patch Management Planning

Enterprise patch management has been a contentious issue for decades, with personnel from the security and business/mission sides of organizations often having conflicting opinions. For example, many organizations have struggled with balancing the trade-offs between earlier deployment and more testing. Deploying patches more quickly reduces the window of opportunity for attackers but increases the risk of operational disruption because of the lack of testing. Conversely, testing patches before deployment decreases the risk of operational disruption but increases the window of opportunity for attackers. Testing can also consume considerable staff resources, and it still might miss problems.

What has made enterprise patch management tougher recently is how dynamic and dispersed computing assets are, as well as the sheer number of installed software components to patch. In addition, patch management processes and technology take different forms depending on the type of assets (e.g., OT, IoT, mobile, cloud, traditional IT, virtual machines, containers). The result is that many organizations are unable to keep up with patching. Patching often becomes primarily reactive (i.e., quickly deploy a patch when a severe vulnerability is being widely exploited) versus proactive (i.e., quickly deploy patches to correct many vulnerabilities before exploitation is likely to occur).

Being proactive means doing more work now to reduce the likelihood of incidents in the future. It also means that if a patch fails, that disruption can be managed and remediated on the organization's schedule. Being reactive means that if a compromise of an unpatched vulnerability occurs (e.g., a data breach, a ransomware infection), the organization will have to perform incident response, their reputation may be damaged, and/or they may potentially be fined or sued. As part of incident response efforts, the missing patch will probably need to be installed anyway in addition to installing all the preceding patches it is dependent upon, as well as performing other prerequisite recovery actions such as reverting to a good known state or rebuilding the environment from scratch.

What needs to change in many organizations is the perception that an operational disruption caused by patching is harm that the organization is doing to itself, while an operational disruption caused by a cybersecurity incident is harm caused by a third party. While those may be true statements in isolation, they are misleading and incomplete as part of an organization's risk responses. Disruptions from patching are largely controllable, while disruptions from incidents are largely uncontrollable. Disruptions from patching are also a necessary part of maintaining nearly all types of technology in order to avoid larger disruptions from incidents.

That being said, security and technology personnel can take steps to reduce the likelihood of patching causing disruptions, as well as direct patching efforts to prioritize the vulnerabilities that are causing the most risk to the organization. Planning these actions necessitates cooperation between the security/technology and business/mission sides of the organization. This section presents actionable recommendations that organizations should implement to improve their enterprise patch management planning, thereby minimizing the potential negatives of patching to operations.

The recommendations support the following principles, which organizations should strive to adopt in their enterprise patch management practices:

- **Problems are inevitable; be prepared for them.** Risk responses, including patching, will never be perfect. Some may inadvertently cause operational problems, for example, but most will not. To improve enterprise patch management, organizations need to change their culture so that instead of fearing problems and thus delaying risk responses, personnel are prepared to address problems when they occur. The organization needs to become more resilient, and everyone in the organization needs to understand that problems caused by patching are a necessary inconvenience that helps prevent major compromises.
- **Simplify decision making.** Conducting a risk assessment of each new vulnerability in order to plan the optimal risk response for it is simply not feasible. Organizations do not have the time, resources, expertise, or tools to do so. Planning needs to be done in advance so that when a new vulnerability becomes known, a decision can quickly be made about how to respond to it.
- **Rely on automation.** There is no way that an organization can keep up with patching without automation because of the sheer number of assets, software installations, vulnerabilities, and patches. Automation is also needed for emergency situations, like patching a severe vulnerability that attackers are actively exploiting. Having automation in place gives an organization agility and scalability when it comes to its risk responses.
- **Start improvements now.** Some of the changes that an organization may need to make might take years to put in place, but that does not mean that other practices cannot be improved in the meantime.

3.1 Reduce Patching-Related Disruptions

Organizations should strive to decrease the number of vulnerabilities introduced into their environments. This shrinks the attack surface and can lower the amount of patching that organizations need to do. Possible methods for decreasing the number of vulnerabilities include:

- Harden software, such as enforcing the principles of least privilege and least functionality (e.g., deactivating or uninstalling software services, features, and other components that are not needed). For additional information on hardening assets, especially those considered critical, see the NIST publication *Security Measures for "EO-Critical Software" Use Under Executive Order (EO) 14028* (July 9, 2021).
- Acquire software that is likely to have fewer vulnerabilities over time compared to other software.
- Work with software development partners that are likely to introduce fewer vulnerabilities into software over time, taking into consideration factors such as how rigorous their secure software development practices are, how quickly they address issues and release patches, how often problems are associated with their patches, and how transparent they are in their security-related communications.
- Use managed services instead of software when feasible.

- Select stacks or platforms that are likely to have fewer vulnerabilities over time compared to other stacks or platforms (e.g., running software within a small container instead of a larger operating system).

Organizations should consider deploying applications in ways that make patching less likely to disrupt operations. One example is to run applications on stacks or platforms where patching is a fundamental part of the deployed technology and is less likely to disrupt operations (e.g., modernizing and running software within cloud-based containers instead of on-premises server operating systems). Another example is to take advantage of existing toolchains that already build applications with updated components and test them before production release.

3.2 Inventory Your Software and Assets

Organizations should establish and constantly maintain up-to-date software inventories for their physical and virtual computing assets, including OT, IoT, and container assets. This information could be in a single enterprise asset inventory, or it could be split among multiple resources. While a comprehensive inventory of all assets is ideal, it may be impossible to achieve, given the highly dynamic nature of assets and software. A realistic goal is to maintain a close-to-comprehensive inventory by relying on automation to constantly discover new assets and collect up-to-date information on all assets. Some vendors might also provide machine-consumable data on their assets' software composition, such as a software bill of materials (SBOM), which could be used to augment organization inventories.

Without constant updates, inventories will quickly become outdated and provide increasingly inaccurate and incomplete information for patching efforts. At one time, when assets and software were mostly static and were located within static logical and physical perimeters, it was generally considered acceptable to update inventories on a monthly or quarterly basis by performing a vulnerability scan. That model should no longer be used.

Constantly updating inventories for all of the technologies and environments in use today requires a combination of automation techniques and tools. Organizations should leverage inventory capabilities built into platforms and assets whenever feasible. For example, APIs built into a cloud-based platform may enable continuous updates of inventory information for the software on that platform, as well as other platform characteristics helpful for patch management purposes. Vulnerability scans and passive network monitoring on local networks can still contribute to asset inventories, especially in terms of asset discovery. If vulnerability scans are to be used for software inventories, they will need sufficient access to the assets (i.e., authenticated scanning) in order to detect changes to their software and other technical characteristics.

Organizations should approach patching from a per-asset perspective. Software inventories should include information on each computing asset's technical characteristics and mission/business characteristics. Making decisions for risk responses and their prioritization should not be based solely on which software and software versions are in use. Each asset has technical and mission/business characteristics that should be taken into consideration because they provide context for the vulnerable software running on that asset.

The characteristics that an organization should inventory will vary, but the following are examples of possible characteristics to track:

- The asset's platform type (e.g., IT, OT, IoT, mobile, cloud, VM)
- The party who administrates the asset (e.g., IT department, third party, end user, vendor/manufacturer, shared responsibility model)
- The applications, services, or other mechanisms used to manage the asset (e.g., endpoint management software, virtual machine manager, container management software)
- The asset's network connectivity in terms of protocols, frequency/duration, and bandwidth
- The technical security controls already in place to safeguard the asset
- The asset's primary user(s) or interconnected services and their privileges

Examples of mission/business characteristics that an organization should track include:

- The asset's role and importance to the organization, which are contextual and may be hard to define or determine
- Laws, regulations, or policies that specify how soon a new vulnerability in the asset must be addressed
- Contractual restrictions on patching (e.g., a highly regulated asset can only be patched by its manufacturer after testing and certification)
- Mission/business restrictions on risk responses for that asset (e.g., an asset can only be rebooted during a monthly maintenance outage)

Tracking technical and mission/business characteristics for each computing asset provides the basis for better decision making regarding risk responses and priorities. The tracked characteristics are also valuable for other enterprise security and technology purposes, such as supporting efforts to shift to zero-trust architectures. If the history of patching is tracked for individual assets, that information may be particularly helpful to incident responders during an investigation.

3.3 Define Risk Response Scenarios

Organizations should define the software vulnerability risk response scenarios they need to be prepared to handle. Examples of such scenarios include:

- **Routine patching.** This is the standard procedure for patches that are on a regular release cycle and have not been elevated to emergency status. Most patching falls under this scenario. However, because routine patching does not have the urgency of emergency scenarios, and routine patch installation can interrupt operations (e.g., device reboots), it is often postponed and neglected. This provides many additional windows of opportunity for attackers. Delaying routine patching also makes emergency patching more difficult, time-consuming, and disruptive because of the need to first install previous patches that new patches depend upon.

- **Emergency patching.** This is the procedure to address patching emergencies in a crisis situation, such as a severe vulnerability or a vulnerability being actively exploited. If one or more of the organization's vulnerable assets have already been compromised, emergency patching may be part of incident response efforts. Emergency patching needs to be handled as efficiently as possible to prevent the imminent exploitation of vulnerable assets.
- **Emergency mitigation.** This is the emergency procedure in a crisis situation, like those described above for the emergency patching scenario, to temporarily mitigate vulnerabilities before a patch is available. The mitigation can vary and may or may not need to be rolled back afterward. Emergency mitigations are sometimes needed because of issues with a patch. For example, a patch might be flawed and not actually correct a vulnerability, or a patch might inadvertently disrupt the operation of other software or systems. A patch could even be compromised.
- **Unpatchable assets.** This is the implementation of isolation or other methods to mitigate the risk of systems that cannot be easily patched. This is typically required if routine patching is not able to accommodate these systems within a reasonable time frame. Examples of why an asset may be unpatchable include the vendor not providing patches (e.g., asset is at end-of-life, asset does not support updates) or an asset needing to run uninterrupted for an extended period of time because it provides mission-critical functions. Unpatchable assets need to be included in risk response planning because a new vulnerability in an asset might necessitate a change in the methods needed to mitigate its risk.

3.4 Assign Each Asset to a Maintenance Group

Organizations should use the software inventories, technical and business/mission characteristics, and risk response scenarios to assign each asset to a maintenance group. A *maintenance group* is a set of assets with similar characteristics that generally have the same software maintenance needs for each risk response scenario. Maintenance needs include not only patching (e.g., patch schedule, patch testing needs, outage restrictions, level of impact if vulnerable software is compromised) but also any other appropriate forms of mitigation and risk response, such as temporary mitigations used when patches are not yet available. Organizations should define their maintenance groups at whatever they decide the best level of granularity is, then periodically reassess their maintenance group definitions and adjust them as needed.

Instead of denoting certain assets or types of assets as "exceptions," there should be maintenance groups for them. If an asset cannot be patched or should not be patched, there is one less option for addressing its vulnerabilities. It still has software maintenance needs, so it should belong to a maintenance group.

Here are a few simplified examples of possible maintenance groups:

- Mobile workforce laptops for standard end users
    - Software to patch: Firmware, operating systems, and client applications for end user devices
    - Outage restrictions: Tolerant to downtime

  - Existing mitigations: Endpoint security controls running on the laptops
  - Level of impact to the organization if compromised: Moderate
- On-premises datacenter (including servers, network equipment, storage, etc.)
  - Software to patch: Firmware, operating systems, and applications for server platforms
  - Outage restrictions: Must adhere to scheduled outage windows for all non-emergency situations
  - Existing mitigations: Network-based security controls restricting access to the assets and security controls running on the assets themselves
  - Level of impact to the organization if compromised: High
- Legacy OT assets
  - Software to patch: None; existing software is no longer supported and cannot be patched
  - Outage restrictions: Must adhere to scheduled outage windows for all non-emergency situations
  - Existing mitigations: Network isolation, physical security controls
  - Level of impact to the organization if compromised: High
- Smartphones for the mobile workforce
  - Software to patch: Operating systems and mobile apps
  - Outage restrictions: Tolerant to downtime
  - Existing mitigations: Mobile device security controls running on the smartphones
  - Level of impact to the organization if compromised: Moderate
- On-premises servers for automated software testing
  - Software to patch: Firmware, server operating systems, virtualization software, server and client guest operating systems, server and client applications
  - Outage restrictions: Usually tolerant to downtime
  - Existing mitigations: Network-based security controls restricting access to the assets, and security controls running on the assets themselves
  - Level of impact to the organization if compromised: Moderate
- Containers with customer-facing applications in the public cloud
  - Software to patch: Container operating systems, application modules
  - Outage restrictions: Highly tolerant to downtime
  - Existing mitigations: Security controls running on the container operating system
  - Level of impact to the organization if compromised: High

Maintenance groups can also be defined based on other characteristics, like personnel roles (e.g., software developer workstations, system administrator workstations) or asset importance (e.g., low-impact IoT consumer assets, OT and IoT assets with life-safety impact).

3.5 Define Maintenance Plans for Each Maintenance Group

Organizations should define a maintenance plan for each maintenance group for each applicable risk response scenario. A maintenance plan defines the actions to be taken when a scenario occurs for a maintenance group, including the timeframes for beginning and ending each action, along with any other pertinent information. Along with the maintenance plans, organizations should define a risk assessment process for determining which plan should be used at any given time and for deciding when to switch from one plan to another as the understanding of risk changes.

The following subsections discuss what the maintenance plan for each scenario might involve.

The Known Exploited Vulnerabilities Catalog [3] is a list of vulnerabilities that are considered particularly important for organizations to mitigate. Each meets a set of criteria, such as being exploited in the wild and having patches or other mitigations available. The Cybersecurity and Infrastructure Security Agency (CISA) created and maintains the catalog. CISA's Binding Operational Directive 22-01 [4] requires federal agencies to remediate new vulnerabilities added to the catalog within two weeks. Other organizations may voluntarily choose to use the catalog to help prioritize their patching efforts.

CISA has also released *Cybersecurity Incident & Vulnerability Response Playbooks* [5] for federal agencies. The Vulnerability Response Playbook "standardizes the high-level process that agencies should follow when responding to…urgent and high-priority vulnerabilities" being actively exploited in the wild, so it may be particularly helpful to federal agencies in developing their maintenance plans.

3.5.1 Maintenance Plans for Scenario 1, Routine Patching

Organizations should consider adopting phased deployments for routine patching in which a small subset of the assets to be patched receive the patch first. These assets act as canaries (i.e., bellwethers) for identifying issues and determining the likely operational impact of the patch. In effect, this is how the patching gets tested. If the canary assets indicate that the patch should have minimal impact, the deployment can expand to more or all of the vulnerable assets. Significant problems can be addressed before the rollout expands, or a different risk response – like a temporary mitigation – can be planned instead of the patch while the problems are resolved.

For larger routine patch rollouts, especially those that will directly impact the organization's users, multiple rounds of canary assets could be used. For example, there could be a small first round with technically knowledgeable users (e.g., system administrators, security engineers) that lasts a few days, followed by a larger second round with "early adopters" across the organization who are willing to try app updates for a few weeks and report any problems that occur. Ideally, the early adopters will be representative of the entire user community who will eventually be using the patch.

Organizations should offer flexibility with how soon routine patches are to be installed, while also forcing installation after a grace period has ended. A routine patch does not necessitate immediate installation, but at some point, patches must be installed to reduce the risk for the entire environment. Forcing installation can be direct, like triggering patch execution, or indirect, like preventing network access for unpatched assets until they are patched.

3.5.2 Maintenance Plans for Scenario 2, Emergency Patching

Organizations should consider using the same general approach for emergency patching as for routine patching, except with a highly accelerated schedule. Even under emergency circumstances, it may still be beneficial to first deploy a new patch to a small number of canary assets to confirm that the patch is not corrupted and does not break the software. This period could last a few minutes to a few hours, and the emergency patching itself could occur in the following hours or days, depending on how urgent the emergency is.

3.5.3 Maintenance Plans for Scenario 3, Emergency Mitigation

Organizations should plan for the quick implementation of multiple types of emergency mitigations to protect vulnerable assets. Mitigations may require deactivating system functionality or isolating an asset from other assets and having automated mechanisms to apply these changes. Without the processes, procedures, and tools in place to implement mitigations, too much time may be lost, and vulnerable assets may be compromised.

Organizations should plan to replace emergency mitigations with permanent fixes. Once a permanent fix, such as a patch, is available, the patch will need to be deployed and the mitigation removed. Schedules should be set and enforced for both patch deployment and mitigation removal.

3.5.4 Maintenance Plans for Scenario 4, Unpatchable Assets

Organizations should plan to implement multiple types of long-term risk mitigation methods besides patching to protect vulnerable assets. There should be an approved set of methods for each maintenance group, and these methods should have been reviewed and analyzed in advance by security architects/engineers to determine their adequacy in mitigating risk. For example, Section 3 of NIST SP 800-207 [6] describes the use of micro-segmentation, software-defined perimeters, and other risk mitigation methods for isolating assets based on a defined policy.

Organizations should plan to implement multiple types of mitigations to protect vulnerable unpatchable assets. In addition to using long-term risk mitigation methods for unpatchable assets, organizations should also implement mitigations as needed to prevent exploitation of specific vulnerabilities that the long-term risk mitigation methods don't adequately address.

Organizations should plan on periodically reevaluating their alternatives to patching. There are two main aspects to this. One is conducting a risk assessment to see if the alternatives to patching are still sufficiently effective at mitigating risk. The other is conducting a cost-benefit analysis to see if the assets provide sufficient value to the organization compared with the additional costs of mitigating, transferring, or accepting the risk of unpatchable assets.

3.5.5 Exceptions to Maintenance Plans

Organizations should closely track and monitor all exceptions to maintenance plans. As explained in Section 3.4, maintenance groups should be defined to minimize assets considered "exceptions." However, having some exceptions is inevitable. All exceptions to maintenance plans should be reviewed regularly to determine if the maintenance plan can be implemented now. Assets with similar long-term exceptions might need to be moved to a separate maintenance group with its own maintenance plan.

3.6 Choose Actionable Enterprise-Level Patching Metrics

Metrics play several roles in patch management and vulnerability management. A common example is estimating the relative importance of a new vulnerability so that its remediation can be prioritized appropriately, such as something to be addressed by routine patching versus emergency patching or mitigations. There are many free and commercial sources of this information for organizations to leverage.

What organizations often find more challenging is identifying meaningful, actionable enterprise-level metrics that they can adopt to monitor and track their progress with patch management to support their continuous improvement and effectiveness program. Moreover, there are numerous audiences for these metrics, potentially including the organization's chief executive officer (CEO)/Board of Directors, chief information officer (CIO), chief information security officer (CISO), mission/business unit leadership, application developers, security and system administrators, and other cybersecurity and IT personnel. Typically, each of these audiences needs a somewhat different set of metrics that corresponds to their role and responsibilities. For example, a Linux system administrator might need metrics on the performance of Linux asset patching, while a business unit's leadership might want to compare the overall effectiveness of their assets' patching with that of the organization's other business units.

Organizations should take advantage of low-level metrics that they already collect when developing enterprise-level metrics to capture patching performance. There is often a wealth of information already available from the inventories of software and assets, especially the assets' technical and mission/business characteristics. Similarly, organizations often have detailed information about the vulnerabilities themselves, such as Common Vulnerability Scoring System (CVSS) scores, threat intelligence about the vulnerabilities being exploited, and other metrics provided by vulnerability management tools that help indicate how important each vulnerability was to mitigate.

Organizations should utilize their existing low-level metrics to develop enterprise-level metrics that reflect the relative importance of each vulnerability and patch. Overly simplistic metrics, such as counting the number of vulnerabilities that the entire organization had and what percentage of them were patched, are not actionable. If you were told that 10 % of your assets were not being patched, what does that actually mean in terms of your organization's risk? What is the relative importance of each of those assets? If they are the most important assets, then not patching them might be a major problem. If they are the least important assets, then not patching them might indicate reasonable prioritization of limited resources. To look at the assets another way, what is the relative severity of the unpatched vulnerabilities versus the patched vulnerabilities?

Metrics that are too simple are generally not actionable because they do not provide enough information. They do not offer the insights into the performance of vulnerability management that are needed to identify the nature of problems and the improvements necessary to address those problems and improve the organization's vulnerability management performance. Table 1 shows a notional example of actionable performance metrics. Each cell provides mitigation metrics based on the relative importance of the assets (low, moderate, or high) and the vulnerabilities (low, medium, high, or critical), with the categories defined by the organization. The metrics in each cell reflect the percentage of assets that were patched by the corresponding maintenance plans' deadlines, as well as the average (mean) time and median time for patching.

Table 1: Vulnerability Mitigation Time Summary Matrix

Vulnerability Importance	Asset Importance		
	Low	Moderate	High
Low	By deadline: 64.7 % Average time: 80.4 days Median time: 75.2 days	By deadline: 72.4 % Average time: 34.7 days Median time: 33.7 days	By deadline: 85.0 % Average time: 14.6 days Median time: 8.1 days
Medium	By deadline: 66.5 % Average time: 75.1 days Median time: 70.7 days	By deadline: 68.7 % Average time: 33.2 days Median time: 31.6 days	By deadline: 71.4 % Average time: 12.9 days Median time: 10.5 days
High	By deadline: 68.6 % Average time: 62.1 days Median time: 58.0 days	By deadline: 78.8 % Average time: 26.8 days Median time: 22.1 days	By deadline: 85.5 % Average time: 8.8 days Median time: 8.1 days
Critical	By deadline: 81.4 % Average time: 44.4 days Median time: 41.3 days	By deadline: 92.3 % Average time: 21.2 days Median time: 23.9 days	By deadline: 95.2 % Average time: 5.2 days Median time: 5.1 days

With the additional characteristics that an organization has on their assets and vulnerabilities, it can analyze its mitigation time data by platform, business unit, maintenance group, and other characteristics to find the aspects of vulnerability mitigation that are in greatest need of improvement, as well as to set target values for improving those metrics. Analyzing the data by different characteristics can also provide metrics that are more relevant to particular audiences, such as organizational executives, technology leadership, IT operations staff, and compliance professionals.

Organizations should frequently update their low-level metrics and strive for them to be as accurate as possible in order to improve the enterprise-level metrics based on them. If low-level metrics are incorrect, they will negatively impact the enterprise-level metrics calculated from them. For example, if an organization only scans for vulnerabilities monthly, their low-level metrics for the number of vulnerabilities present in their assets would be much smaller than they should be. This would provide a misleading picture of the organization's vulnerability management program. Similarly, collecting low-level metrics through less accurate methods, such as passive (unauthenticated) instead of active (authenticated) vulnerability scans, will generally underreport vulnerabilities and thus skew the higher-level metrics.

3.7 Consider Software Maintenance in Procurement

Organizations should take software maintenance into consideration when procuring software. Software maintenance is one factor of many that organizations should consider. It is beyond the scope of this publication to provide methodologies for estimating software maintenance costs or factoring software maintenance into procurement decisions. However, the following is a sample questionnaire that an organization could use to help it understand the software maintenance needs of new software that it may procure:

1. Will you be releasing updates for this software to address vulnerabilities?
2. Approximately how many patches, updates, and upgrades do you expect to release each year for this software? On average, how quickly do you expect to address each vulnerability and release a patch? Do you bundle patches together or is each patch separate? How often do you anticipate releasing emergency updates?
3. For how many years are you committed to correcting vulnerabilities in the software?
4. Will you release updates on a regular schedule, as needed, or both? If a schedule will be followed, what is that schedule (weekly, monthly, quarterly, etc.)?
5. Do you have a vulnerability disclosure and incident response program for your software? How transparent are you in your security-related communications to your customers?
6. When a vulnerability in your software becomes public but a patch, update, or upgrade is not available, how do you recommend that customers protect their computing assets running your software? Will you provide an emergency mitigation to prevent vulnerability exploitation while maintaining most or all software functionality?
7. When your software is patched or updated, how disruptive will that be to the operating software? For instance, will it require restarting the software, rebooting the asset on which the software is running, etc.?
8. How do you test your patches before release? How often do customers experience significant issues with your patches? Do you provide rollback capabilities for uninstalling patches?

References

[1] Diamond T, Kerman A, Souppaya MP, Stine KM, Johnson B, Peloquin C, Ruffin V, Simos M, Sweeney S, Scarfone KA (2022) Improving Enterprise Patching for General IT Systems: Utilizing Existing Tools and Performing Processes in Better Ways. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 1800-31. Available at https://www.nccoe.nist.gov/projects/critical-cybersecurity-hygiene-patching-enterprise

[2] Stine KM, Quinn SD, Witte GA, Gardner RK (2020) Integrating Cybersecurity and Enterprise Risk Management (ERM). (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 8286. https://doi.org/10.6028/NIST.IR.8286

[3] Cybersecurity & Infrastructure Security Agency (2022) *Known Exploited Vulnerabilities Catalog*. Available at https://www.cisa.gov/known-exploited-vulnerabilities-catalog

[4] Cybersecurity & Infrastructure Security Agency (2021) Reducing the Significant Risk of Known Exploited Vulnerabilities. (Cybersecurity and Infrastructure Security Agency, Washington, DC), Binding Operational Directive 22-01. Available at https://www.cisa.gov/binding-operational-directive-22-01

[5] Cybersecurity & Infrastructure Security Agency (2021) Cybersecurity Incident & Vulnerability Response Playbooks: Operational Procedures for Planning and Conducting Cybersecurity Incident and Vulnerability Response Activities in FCEB Information Systems. (Cybersecurity and Infrastructure Security Agency, Washington, DC). Available at https://www.cisa.gov/sites/default/files/publications/Federal_Government_Cybersecurity_Incident_and_Vulnerability_Response_Playbooks_508C.pdf

[6] Rose SW, Borchert O, Mitchell S, Connelly S (2020) Zero Trust Architecture. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-207. https://doi.org/10.6028/NIST.SP.800-207

[7] Joint Task Force (2020) Security and Privacy Controls for Information Systems and Organizations. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-53, Rev. 5, Includes updates as of December 10, 2020. https://doi.org/10.6028/NIST.SP.800-53r5

[8] National Institute of Standards and Technology (2018) Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1. (National Institute of Standards and Technology, Gaithersburg, MD). https://doi.org/10.6028/NIST.CSWP.04162018

Appendix A—Mappings to NIST Guidance and Frameworks

The controls in the NIST SP 800-53 Revision 5, *Security and Privacy Controls for Information Systems and Organizations* [7], control catalog that are most important for enterprise patch management planning are:

- CM-2, Baseline Configuration
- CM-3, Configuration Change Control
- CM-8, System Component Inventory
- RA-7, Risk Response
- SI-2, Flaw Remediation
- SR-2, Supply Chain Risk Management Plan
- SR-3, Supply Chain Controls and Processes
- SR-5, Acquisition Strategies, Tools, and Methods

The Subcategories from the Cybersecurity Framework [8] that are most important for enterprise patch management planning are:

- ID.AM-1: Physical devices and systems within the organization are inventoried
- ID.AM-2: Software platforms and applications within the organization are inventoried
- ID.AM-4: External information systems are catalogued
- ID.AM-5: Resources (e.g., hardware, devices, data, time, personnel, and software) are prioritized based on their classification, criticality, and business value
- ID.BE-3: Priorities for organizational mission, objectives, and activities are established and communicated
- ID.BE-5: Resilience requirements to support delivery of critical services are established for all operating states (e.g., under duress/attack, during recovery, normal operations)
- ID.GV-3: Legal and regulatory requirements regarding cybersecurity, including privacy and civil liberties obligations, are understood and managed
- ID.RA-5: Threats, vulnerabilities, likelihoods, and impacts are used to determine risk
- ID.RA-6: Risk responses are identified and prioritized
- ID.SC-1: Cyber supply chain risk management processes are identified, established, assessed, managed, and agreed to by organizational stakeholders
- PR.IP-1: A baseline configuration of information technology/industrial control systems is created and maintained incorporating security principles (e.g., concept of least functionality)
- PR.IP-3: Configuration change control processes are in place
- PR.IP-12: A vulnerability management plan is developed and implemented

Appendix B—Acronyms

Selected acronyms and abbreviations used in this paper are defined below.

BIOS	Basic Input/Output System
BYOD	Bring Your Own Device
CEO	Chief Executive Officer
CIO	Chief Information Officer
CISA	Cybersecurity & Infrastructure Security Agency
CISO	Chief Information Security Officer
CVSS	Common Vulnerability Scoring System
ERM	Enterprise Risk Management
FIPS	Federal Information Processing Standards
FISMA	Federal Information Security Modernization Act
FOIA	Freedom of Information Act
IoT	Internet of Things
IR	Interagency or Internal Report
IT	Information Technology
ITL	Information Technology Laboratory
NIST	National Institute of Standards and Technology
OMB	Office of Management and Budget
OS	Operating System
OT	Operational Technology
SaaS	Software as a Service
SBOM	Software Bill of Materials
SP	Special Publication