



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

**Special Publication 800-40**  
**Revision 3 (Draft)**

---

# **Guide to Enterprise Patch Management Technologies (Draft)**

---

**Recommendations of the National Institute  
of Standards and Technology**

---

Murugiah Souppaya  
Karen Scarfone

**NIST Special Publication 800-40**  
**Revision 3 (Draft)**

# Guide to Enterprise Patch Management Technologies (Draft)

*Recommendations of the National  
Institute of Standards and Technology*

**Murugiah Souppaya**

*Computer Security Division*

*Information Technology Laboratory*

*National Institute of Standards and Technology*

*Gaithersburg, MD 20899-8930*

**Karen Scarfone**

*Scarfone Cybersecurity*

---

# C O M P U T E R   S E C U R I T Y

---

September 2012



**U.S. Department of Commerce**

Rebecca Blank, Acting Secretary

**National Institute of Standards and Technology**

Patrick D. Gallagher, Under Secretary of Commerce  
for Standards and Technology and Director

## **Reports on Computer Systems Technology**

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in Federal information systems. The Special Publication 800-series reports on ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

## Authority

This publication has been developed by NIST to further its statutory responsibilities under the Federal Information Security Management Act (FISMA), Public Law (P.L.) 107-347. NIST is responsible for developing information security standards and guidelines, including minimum requirements for Federal information systems, but such standards and guidelines shall not apply to national security systems without the express approval of appropriate Federal officials exercising policy authority over such systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), *Securing Agency Information Systems*, as analyzed in Circular A-130, Appendix IV: *Analysis of Key Sections*. Supplemental information is provided in Circular A-130, Appendix III, *Security of Federal Automated Information Resources*.

Nothing in this publication should be taken to contradict the standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other Federal official. This publication may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright in the United States. Attribution would, however, be appreciated by NIST.

**National Institute of Standards and Technology Special Publication 800-40 Revision 3 (Draft)**  
**Natl. Inst. Stand. Technol. Spec. Publ. 800-40 r3, 26 pages (Sep. 2012)**  
**CODEN: NSPUE2**

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by Federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, Federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. All NIST publications, other than the ones noted above, are available at <http://csrc.nist.gov/publications>.

**Public comment period: September 5 through October 5, 2012**

National Institute of Standards and Technology  
Attn: Computer Security Division, Information Technology Laboratory  
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930  
Electronic mail: 800-40comments@nist.gov

## **Acknowledgments**

The authors, Murugiah Souppaya of the National Institute of Standards and Technology (NIST) and Karen Scarfone of Scarfone Cybersecurity, wish to thank their colleagues who reviewed drafts of this document and contributed to its technical content.

## **Acknowledgments, Version 2**

The authors, Peter Mell of NIST, Tiffany Bergeron of The MITRE Corporation, and David Henning of Hughes Network Systems, LLC, wish to express their thanks to Rob Pate of the United States Computer Emergency Readiness Team (US-CERT) for providing support for this publication. In addition, the authors would like to thank Miles Tracy of the U.S. Federal Reserve System, who co-authored the original version of the publication and provided significant input for this version, and Tanyette Miller of Booz Allen Hamilton, who put together the patching resources found in the appendices. The authors would also like to express their thanks to Timothy Grance of NIST, Manuel Costa and Todd Wittbold of The MITRE Corporation, Matthew Baum of the Corporation for National and Community Service, and Karen Kent of Booz Allen Hamilton for their insightful reviews, and to representatives from Department of Health and Human Services, Department of State, Environmental Protection Agency, Federal Reserve Board, and PatchAdvisor for their particularly valuable comments and suggestions.

## **Abstract**

Patch management is the process for identifying, acquiring, installing, and verifying patches for products and systems. Patches correct security and functionality problems in software and firmware. There are several challenges that complicate patch management. If organizations do not overcome these challenges, they will be unable to patch systems effectively and efficiently, leading to easily preventable compromises. This publication is designed to assist organizations in understanding the basics of enterprise patch management technologies. It explains the importance of patch management and examines the challenges inherent in performing patch management. It provides an overview of enterprise patch management technologies and it also briefly discusses metrics for measuring the technologies' effectiveness and for comparing the relative importance of patches.

## **Keywords**

information security; patch management; remediation; software patches; vulnerability management

## **Trademark Information**

Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.

All other names are registered trademarks or trademarks of their respective companies.

## Table of Contents

Executive Summary .....	vi
<b>1. Introduction .....</b>	<b>1</b>
1.1 Document Purpose and Scope .....	1
1.2 Audience .....	1
1.3 Document Structure .....	1
<b>2. The Importance of Patch Management .....</b>	<b>2</b>
<b>3. The Challenges of Patch Management .....</b>	<b>3</b>
3.1 Timing, Prioritization, and Testing .....	3
3.2 Patch Management Configuration .....	4
3.3 Alternative Host Architectures .....	5
3.4 Other Challenges .....	6
3.4.1 Software Inventory Management .....	6
3.4.2 Resource Overload .....	6
3.4.3 Installation Side Effects .....	6
3.4.4 Patch Implementation Verification .....	6
3.4.5 Application Whitelisting .....	6
<b>4. Enterprise Patch Management Technologies .....</b>	<b>8</b>
4.1 Components and Architecture .....	8
4.1.1 Agent-Based .....	8
4.1.2 Agentless Scanning .....	8
4.1.3 Passive Network Monitoring .....	9
4.2 Security Capabilities .....	9
4.2.1 Inventory Management Capabilities .....	9
4.2.2 Patch Management Capabilities .....	9
4.2.3 Other Capabilities .....	9
4.3 Management Capabilities .....	9
4.3.1 Technology Security .....	10
4.3.2 Phased Deployment .....	10
4.3.3 Usability and Availability .....	10
<b>5. Metrics .....</b>	<b>12</b>

## List of Appendices

Appendix A— Security Content Automation Protocol (SCAP) Tutorial .....	14
Appendix B— Summary of Recommendations .....	16
Appendix C— Acronyms and Abbreviations .....	18

## 1 **Executive Summary**

2 *Patch management* is the process for identifying, acquiring, installing, and verifying patches for products  
3 and systems. Patches correct security and functionality problems in software and firmware. From a  
4 security perspective, patches are most often of interest because they are mitigating software flaw  
5 vulnerabilities; applying patches to eliminate these vulnerabilities significantly reduces the opportunities  
6 for exploitation. Patches serve other purposes than just fixing software flaws; they can also add new  
7 features to software and firmware, including security capabilities.

8 There are several challenges that complicate patch management. If organizations do not overcome these  
9 challenges, they will be unable to patch systems effectively and efficiently, leading to easily preventable  
10 compromises. Organizations that can minimize the time they spend dealing with patching can use those  
11 resources for addressing other security concerns. Already many organizations have largely  
12 operationalized their patch management, making it more of a core IT function than a part of security.  
13 However, it is still important for all organizations to carefully consider patch management in the context  
14 of security because patch management is so important to achieving and maintaining sound security.

15 This publication is designed to assist organizations in understanding the basics of enterprise patch  
16 management technologies. It explains the importance of patch management and examines the challenges  
17 inherent in performing patch management. It provides an overview of enterprise patch management  
18 technologies and it also briefly discusses metrics for measuring the technologies' effectiveness and for  
19 comparing the relative importance of patches.

20 Organizations should implement the following recommendations to improve the effectiveness and  
21 efficiency of their enterprise patch management technologies.

### 22 **Organizations should deploy enterprise patch management tools using a phased approach.**

23 This allows process and user communication issues to be addressed with a small group before deploying  
24 the patch application universally. Most organizations deploy patch management tools first to standardized  
25 desktop systems and single-platform server farms of similarly configured servers. Once this has been  
26 accomplished, organizations should address the more difficult issue of integrating multiplatform  
27 environments, nonstandard desktop systems, legacy computers, and computers with unusual  
28 configurations. Manual methods may need to be used for operating systems and applications not  
29 supported by automated patching tools, as well as some computers with unusual configurations.

### 30 **Organizations should reduce the risks associated with enterprise patch management tools through 31 the application of standard security techniques that should be used when deploying any enterprise- 32 wide application.**

33 Deploying enterprise patch management tools within an enterprise can create additional security risks for  
34 an organization; however, a much greater risk is faced by organizations that do not effectively patch their  
35 systems. Such tools usually increase security far more than they decrease security, especially when the  
36 tools contain built-in security measures to protect against security risks and threats. Risk associated with  
37 these tools include patches being altered, credentials being misused, vulnerabilities in the tools being  
38 exploited, and entities monitoring tool communications to identify vulnerabilities. Examples of possible  
39 countermeasures to these risks include keeping the patching solution components tightly secured and up-  
40 to-date, encrypting network communications, verifying the integrity of patches before installing them, and  
41 testing patches before deployment.

42 **Organizations should balance their security needs with their needs for usability and availability.**

43 For example, installing a patch may “break” other applications; this can best be addressed by testing  
44 patches before deployment. Another example is that forcing application restarts, OS reboots, and other  
45 host state changes is disruptive and could cause loss of data or services. Again, organizations need to  
46 balance the need to get patches applied with the need to support operations. A final example, particularly  
47 important for mobile devices, is the acquisition of updates over low-bandwidth or metered connections; it  
48 may be technically or financially infeasible to download large patches over such connections.  
49 Organizations should make provisions for ensuring that their enterprise patching solution works for  
50 mobile hosts and other hosts used on low-bandwidth or metered networks.

51



## 52 **1. Introduction**

### 53 **1.1 Document Purpose and Scope**

54 This publication is designed to assist organizations in understanding the basics of enterprise patch  
55 management technologies.

### 56 **1.2 Audience**

57 This document has been created for security managers, engineers, administrators, and others who are  
58 responsible for acquiring, testing, prioritizing, implementing, and verifying security patches. Auditors and  
59 others who need to assess the security of systems may also find this publication useful.

### 60 **1.3 Document Structure**

61 This document is organized into the following sections and appendices:

- 62 • Section 2 explains the importance of patch management.
- 63 • Section 3 examines the challenges inherent in performing patch management.
- 64 • Section 4 provides an overview of enterprise patch management technologies.
- 65 • Section 5 briefly discusses possible metrics for measuring the effectiveness of patch management  
66 technologies and for comparing the relative importance of patches.
- 67 • Appendix A provides a tutorial on the Security Content Automation Protocol (SCAP) and its role  
68 in enterprise patch management.
- 69 • Appendix B provides a summary of the main recommendations made throughout the publication.
- 70 • Appendix C defines selected acronyms and other abbreviations for the document.

71

## 72 2. The Importance of Patch Management

73 *Patch management* is the process for identifying, acquiring, installing, and verifying patches for products  
74 and systems. Patches correct security and functionality problems in software and firmware. From a  
75 security perspective, patches are most often of interest because they are mitigating software flaw  
76 vulnerabilities; applying patches to eliminate these vulnerabilities significantly reduces the opportunities  
77 for exploitation. Also, patches are usually the most effective way to mitigate software flaw vulnerabilities,  
78 and are often the only fully effective solution. Sometimes there are alternatives to patches, such as  
79 temporary workarounds involving software or security control reconfiguration, but these workarounds  
80 often negatively impact functionality.

81 Patches serve other purposes than just fixing software flaws; they can also add new features to software  
82 and firmware, including security capabilities. New features can also be added through upgrades, which  
83 bring software or firmware to a newer version in a much broader change than just applying a patch.  
84 Upgrades may also fix security and functionality problems in previous versions of software and firmware.  
85 Also, vendors often stop supporting older versions of their products, which includes no longer releasing  
86 patches to address new vulnerabilities, thus making older versions less secure over time. Upgrades are  
87 necessary to get such products to a supported version that is patched.

88 As Section 3 explains, there are several challenges that complicate patch management. If organizations do  
89 not overcome these challenges, they will be unable to patch systems effectively and efficiently, leading to  
90 easily preventable compromises. Organizations that can minimize the time they spend dealing with  
91 patching can use those resources for addressing other security concerns. Already many organizations have  
92 largely operationalized their patch management, making it more of a core IT function than a part of  
93 security. However, it is still important for all organizations to carefully consider patch management in the  
94 context of security because patch management is so important to achieving and maintaining sound  
95 security.

96 Patch management is required by various security compliance frameworks, mandates, and other policies.  
97 For example, NIST Special Publication (SP) 800-53<sup>1</sup> requires the SI-2, Flaw Remediation security  
98 control, which includes installing security-relevant software and firmware patches, testing patches before  
99 installing them, and incorporating patches into the organization's configuration management processes.  
100 Another example is the Payment Card Industry (PCI) Data Security Standard (DSS)<sup>2</sup>, which requires that  
101 the latest patches be installed and sets a maximum timeframe for installing the most critical patches.

102

103

---

<sup>1</sup> <http://csrc.nist.gov/publications/PubsSPs.html#800-53-rev4>

<sup>2</sup> [https://www.pcisecuritystandards.org/security\\_standards/](https://www.pcisecuritystandards.org/security_standards/)

## 104 **3. The Challenges of Patch Management**

105 This section briefly examines the challenges inherent in performing patch management. These are the  
106 challenges that the patch management technologies discussed in Section 4 are trying to solve.

### 107 **3.1 Timing, Prioritization, and Testing**

108 Timing, prioritization, and testing are intertwined issues for enterprise patch management. Ideally, an  
109 organization would deploy every new patch immediately to minimize the time that systems are  
110 vulnerable. However, in reality this is simply not possible because organizations have limited resources,  
111 which makes it necessary to prioritize which patches should be installed before other patches. Further  
112 complicating this is the significant risk of installing patches without first testing them, which could cause  
113 serious operational disruptions, potentially even more damaging than the corresponding security impact  
114 of not pushing the patches out. Unfortunately, testing patches consumes even more of the limited  
115 resources and makes prioritization even more important. For patch management, timing, prioritization,  
116 and testing are often in conflict.

117 Product vendors have responded to this conflict by bundling patches for their products. Instead of  
118 releasing dozens of patches one at a time over a period of three months, necessitating testing and patch  
119 deployment every few days, a vendor might release their patches in a single bundle once a quarter. This  
120 allows an organization to perform testing once and roll out patches once, which is far more efficient than  
121 testing and rolling out all the patches separately. It also reduces the need to prioritize patches—the  
122 organization just needs to prioritize the bundle instead of separately prioritizing each patch it contains.  
123 Vendors who bundle patches tend to release them monthly or quarterly, except for cases when an  
124 unpatched vulnerability is actively being exploited, in which case they usually issue the appropriate patch  
125 immediately instead of delaying it for the next bundle.

126 There is a downside to patch bundling; it lengthens the time from when a vulnerability is discovered to  
127 the time a patch for it becomes publicly available. If an attacker discovers the same vulnerability before  
128 the patch is released, the attacker may have a longer window of opportunity to exploit the vulnerability  
129 because of the intentional delay in releasing the patch. However, there are two mitigating factors here.  
130 One is that if exploitation is known to be occurring, the vendor is likely to release the patch immediately.  
131 The other factor is that patches may be installed more quickly if they are bundled than if they are all  
132 released separately. So that effectively helps to shrink the window of opportunity for vulnerabilities  
133 associated with bundled patches.

134 There are even more issues to consider with timing. The release of a patch may provide attackers with the  
135 information that they need to exploit the corresponding vulnerability (e.g., reverse engineer the  
136 vulnerability from the patch), meaning that a newly released patch might need to be applied immediately  
137 to avoid compromises. However, if a vulnerability is not being exploited yet, organizations should  
138 carefully weigh the security risks of not patching with the operational risks of patching without  
139 performing thorough testing first. In some operational environments, such as virtual hosts with snapshot  
140 capabilities enabled, it may be preferable to patch without testing as long as the organization is fully  
141 prepared to roll back the patches if there are usability or functionality problems caused by them.

142 Another fundamental issue with timing is forcing the implementation of changes, to make a patch take  
143 effect; this can require restarting a patched application or service, rebooting the operating system<sup>3</sup>, or

---

<sup>3</sup> This can be problematic when the host requires authentication before booting, such as the use of full disk encryption (FDE) software. Organizations using FDE software or other technologies that require authentication before booting should carefully consider the impact that these technologies may have on patch installation.

144 making other changes to the state of the host. Ultimately what matters is not when the patch was installed,  
145 but when the patch actually takes effect. In some cases it may make more sense to mitigate a vulnerability  
146 through an alternative method, at least until patches are fully operational. An example is changing  
147 configuration settings for vulnerable software to temporarily block vulnerable application functionality.  
148 Each mitigation option has different implications for the security, functionality, and operations of the  
149 vulnerable host, so it is not a trivial matter to select one option over others. Also, if configuration settings  
150 are changed, this necessitates preserving the old setting values and restoring them at the appropriate time.  
151 Another problem with changing configuration settings is that they often require a state change to the host  
152 to take effect, such as restarting an application. Implementing configuration changes may be as disruptive  
153 to the operations of a host as installing a patch.

154 Prioritizing which patches to apply and when to apply them is closely related to timing, but there are other  
155 considerations as well. It can depend on the relative importance of the vulnerable systems (for example,  
156 servers versus clients) and the relative severity of each vulnerability (e.g., vulnerability severity metrics  
157 such as the Common Vulnerability Scoring System [CVSS]). Another consideration is dependencies that  
158 patches may have on each other; installing one patch may require installing other patches first, and in  
159 some cases restarting an application or rebooting a host multiple times to make the patches take effect  
160 sequentially.

161 In summary, organizations should carefully consider the relevant issues related to timing, prioritization,  
162 and testing when planning and executing their enterprise patch management processes.

### 163 **3.2 Patch Management Configuration**

164 Another major challenge in enterprise patch management is that there are usually multiple mechanisms  
165 for applying patches. For example:

- 166 • A piece of software may be able to automatically update itself.
- 167 • A centralized OS management tool may be able to initiate patching.
- 168 • Third-party patch management applications may be able to initiate patching.
- 169 • Network access control, health check technologies, and similar technologies may be able to  
170 initiate patching.
- 171 • A user may be able to manually direct software to update itself.
- 172 • A user may be able to manually install a patch or a new version of the software.

173 Having multiple ways of applying patches can cause conflicts. Multiple methods might each try to patch  
174 the same software, which is particularly problematic when the organization doesn't want certain patches  
175 applied because of issues with those patches, testing delays, etc. Multiple methods can also cause patches  
176 to be delayed or missed because each tool or administrator may assume another one is already taking care  
177 of a particular patch. Organizations should identify all the ways in which patches could be applied and act  
178 to resolve any conflicts among patch application methods.

179 A related problem with patch management configuration is that users may override or circumvent patch  
180 management processes. If users are able to make changes to their hosts' software, such as altering settings  
181 (e.g., enabling direct updates, disabling patch management software), installing old versions of software,  
182 and uninstalling patches, they can undermine patch management integrity. To address these problems,

183 organizations should ensure that users cannot disable or otherwise negatively affect enterprise patch  
184 management technologies, and organizations should perform continuous monitoring of enterprise patch  
185 management technologies to identify any issues that occur.

### 186 3.3 Alternative Host Architectures

187 Enterprise patch management is relatively straightforward when all of the hosts are fully managed and  
188 running typical applications and operating systems on a regular platform. When alternative host  
189 architectures are employed, patch management can be considerably more challenging. Examples of these  
190 architectures include the following:

- 191 • **Unmanaged hosts.** As discussed in Section 3.2, it can be much more difficult to control patching  
192 when hosts are not centrally managed (i.e., users manage their own hosts).
  
- 193 • **Out-of-office hosts (e.g., telework laptops).** Hosts on other networks are not protected by the  
194 enterprise’s network security controls (firewalls, network intrusion detection systems,  
195 vulnerability scanners, etc.)
  
- 196 • **Non-standard IT components (e.g., appliances).** On such hosts, it’s often not possible to patch  
197 individual applications independently. Rather, the organization must wait for the component  
198 vendor to release updated software.
  
- 199 • **Mobile devices.** Smartphones, tablets, and other mobile devices (excluding laptops) typically run  
200 mobile operating systems, and patching for these devices is fundamentally different. It is often  
201 necessary to connect the mobile device to a desktop or laptop and to acquire and download  
202 updates through that desktop or laptop. Some mobile devices can directly download updates, but  
203 this can be problematic because of bandwidth considerations (such as taking a long time to  
204 download large updates and paying data charges for the downloads). Another option for keeping  
205 mobile devices updated is the use of enterprise mobile device management software. Enterprise  
206 mobile device management software is used to manage mobile devices, even personally owned  
207 devices not controlled by the organization. It can install, update, and remove applications, and it  
208 can restrict enterprise access if the phone’s operating system and mobile device management  
209 software are not up to date. See Section 3 of SP 800-124 Revision 1, *Guidelines for Managing  
210 and Securing Mobile Devices in the Enterprise*, for more information.
  
- 211 • **Operating system virtualization.** Patches need to be maintained for every OS image and  
212 snapshot used for full virtualization. Patching capabilities are often built into virtualized  
213 environments, such as the ability to patch offline images and quarantine dormant virtual machine  
214 instances. See NIST SP 800-125, *Guide to Security for Full Virtualization Technologies*, for  
215 additional information—specifically, Section 3.3 discusses virtual machine image and snapshot  
216 management.
  
- 217 • **Firmware.** Firmware updates, such as updating the system BIOS, generally require special  
218 privileges and involve different procedures than other types of updates. See NIST SP 800-147,  
219 *BIOS Protection Guidelines*, for additional information on BIOS updates.

220 Organizations should carefully consider all alternative host architectures in use for the enterprise when  
221 designing enterprise patch management policies and solutions.

## 222 **3.4 Other Challenges**

223 This section briefly discusses other challenges not covered earlier in this section.

### 224 **3.4.1 Software Inventory Management**

225 Enterprise patch management is dependent on having a current and complete inventory of the patchable  
226 software (applications and operating systems) installed on each host. This inventory should include not  
227 only which software is currently installed on each host, but also what version of each piece of software is  
228 installed. Without this information, the correct patches cannot be identified, acquired, and installed. This  
229 inventory information is also necessary for identifying older versions of installed software so that they  
230 can be brought up to date. A major benefit of updating older versions is that it reduces the number of  
231 software versions that need to be patched and have their patches tested.

### 232 **3.4.2 Resource Overload**

233 Enterprise patch management can cause resources to become overloaded. For example, many hosts might  
234 start downloading the same large patch (or bundle of patches) at the same time. This could consume  
235 excessive network bandwidth or, if the patches are coming from an organization patch server, overwhelm  
236 the resources of that server. Organizations should ensure that their enterprise patch management can  
237 avoid resource overload situations, such as by sizing the solution to meet expected volumes of requests,  
238 and staggering the delivery of patches so that the enterprise patch management system does not try to  
239 transfer patches to too many hosts at the same time.

### 240 **3.4.3 Installation Side Effects**

241 Installing a patch may cause side effects to occur. A common example is the installation inadvertently  
242 altering existing security configuration settings or adding new settings. This may create a new security  
243 problem in the process of fixing the original vulnerability via patching. Organizations should be capable  
244 of detecting side effects, such as changes to security configuration settings, caused by patch installation.

### 245 **3.4.4 Patch Implementation Verification**

246 As discussed in Section 3.1, an installed patch might not take effect until the affected software is restarted  
247 or other state changes are made. It can be surprisingly difficult to examine a host and determine whether  
248 or not a particular patch has taken effect. This is further complicated when there is no indication for a  
249 patch when it would take effect (reboot required/not required, etc.) One option is to attempt to exploit the  
250 vulnerability, but this is generally only feasible if an exploit already exists, and there are substantial risks  
251 with attempting exploitation, even under highly controlled conditions. Organizations should use other  
252 methods of confirming installation, such as a vulnerability scanner that is independent from the patch  
253 management system.

### 254 **3.4.5 Application Whitelisting**

255 Application whitelisting technologies can conflict with patch management technologies because the  
256 application whitelisting technologies function based on known characteristics of executables and other  
257 application components, which may be changed by patching. If the vendor is providing the whitelist  
258 information, the vendor will have to acquire the patch, record its files' characteristics, and send the  
259 corresponding information to customers. If the organization is building its own whitelist information, it  
260 will have to acquire each patch, record its files' characteristics, and update its whitelists with the new  
261 information. Either method may cause problematic delays for organizations that apply patches quickly,

262 especially automatically; patched software may be seen as unknown software and prohibited from  
263 running.

264 To avoid these problems with updates, most application whitelisting technologies offer maintenance  
265 options. For example, many technologies allow the administrator to select certain services (e.g., patch  
266 management software) to be trusted updaters. This means that any files that they add to or modify on a  
267 host are automatically added to the whitelist. Similar options are available for designating trusted  
268 publishers (i.e., software vendors), users (such as system administrators), sources (such as trusted network  
269 paths), and other trusted entities that may update whitelists. Organizations using application whitelisting  
270 technologies should ensure that they are configured to avoid problems with updates.

271

## 272 **4. Enterprise Patch Management Technologies**

273 This section provides an overview of enterprise patch management technologies. It discusses their  
274 composition, focuses on the security and management capabilities that they provide, and gives  
275 recommendations for their use.

### 276 **4.1 Components and Architecture**

277 Enterprise patch management technologies are similar architecturally to other enterprise security  
278 solutions: one or more centralized servers that provide management and reporting, and one or more  
279 consoles.<sup>4</sup> What distinguishes enterprise patch management technologies from each other architecturally  
280 are the techniques they use to identify missing patches. The three prevalent techniques are agent-based,  
281 agentless scanning, and passive network monitoring. Many products support only one of these techniques,  
282 while other products support more than one. All the techniques are explained in more detail below.  
283 Organizations should carefully consider the advantages and disadvantages of each technique when  
284 selecting enterprise patch management technologies.

#### 285 **4.1.1 Agent-Based**

286 An agent-based patch management technology requires an agent to be running on each host to be  
287 patched<sup>5</sup>, with one or more servers that manage the patching process and coordinate with the agents. Each  
288 agent is responsible for determining what vulnerable software is installed on the host, communicating  
289 with the patch management servers, determining what new patches are available for the host, installing  
290 those patches, and executing any state changes needed to make the patches take effect (e.g., application  
291 restart, OS reboot). Each agent runs with administrator privileges so it can perform these actions. The  
292 patch management server is responsible for providing the agents with information on vulnerable software  
293 and available patches, including where patches can be acquired from and what state changes are needed.

294 Compared to agentless scanning and passive network monitoring, agent-based patch management  
295 technologies are strongly preferred for hosts that are not on the local network all the time, such as  
296 telecommuter laptops and smartphones.

297 There are a few limitations to agent-based patch management technologies. Hosts that don't permit direct  
298 administrator access to the operating system, such as many appliances, generally cannot run agents. Also,  
299 agents may not be available for all of the organization's platforms.

#### 300 **4.1.2 Agentless Scanning**

301 An agentless scanning patch management technology has one or more servers that perform network  
302 scanning of each host to be patched and determine what patches each host needs. Generally agentless  
303 scanning requires the servers to have administrative privileges on each host, so that they can return more  
304 accurate scanning results and they have the ability to install patches and implement state changes on the  
305 hosts (application restarts, OS reboots, etc.)

306 The main advantage of agentless scanning is that it doesn't require the installation and execution of an  
307 agent on each host.

308 One of the primary limitations of agentless scanning is that it omits hosts not on the local network, such

---

<sup>4</sup> Enterprise patch management technologies can also be offered as a managed service.

<sup>5</sup> Agent-based patch management technology is built into some operating systems.



309 as telecommuter laptops and mobile devices. Also, network security controls (e.g., host-based firewalls)  
310 and network technologies (e.g., network address translation) may inadvertently block scanning or  
311 otherwise negatively affect scanning results. Agentless scanning may also negatively impact operations  
312 by consuming excessive amounts of bandwidth. Finally, agentless scanning may not support all of the  
313 organization's platforms.

### 314 **4.1.3 Passive Network Monitoring**

315 Passive network monitoring technologies for patch management monitor local network traffic to identify  
316 applications (and in some cases, operating systems) that are in need of patching.

317 These technologies can be effective at identifying hosts that are not being maintained by other patch  
318 management solutions (agent-based, agentless scanning). They do not require any privileges on the hosts  
319 to be monitored, so they can be used to monitor the patch status of hosts that the organization does not  
320 control (unmanaged systems, visitor systems, contractor systems, etc.)

321 The primary disadvantage of passive network monitoring is that it only works with software where you  
322 can identify the version based on its network traffic (assumed to be unencrypted). Also, of course, it only  
323 works with hosts on the local network.

## 324 **4.2 Security Capabilities**

325 This section describes common security capabilities provided by patch management technologies, divided  
326 into three categories: inventory management, patch management, and other.

### 327 **4.2.1 Inventory Management Capabilities**

328 Patch management technologies typically have capabilities for identifying which software and versions of  
329 software are installed on each host, or alternately, just identifying vulnerable versions of software that are  
330 installed. In addition, some products have features for installing new versions of software, installing or  
331 uninstalling software features, and uninstalling software.

### 332 **4.2.2 Patch Management Capabilities**

333 Patch management technologies obviously provide a range of patch management capabilities. Common  
334 features include identifying which patches are needed, bundling and sequencing patches for distribution,  
335 allowing administrators to select which patches may or may not be deployed, and installing patches and  
336 verifying installation. Many patch management technologies also allow patches to be stored centrally  
337 (within the organization) or downloaded as needed from external sources.

### 338 **4.2.3 Other Capabilities**

339 Many host-based products that have patch management capabilities also provide a variety of other  
340 security capabilities, such as antivirus software, configuration management, and vulnerability scanning.  
341 Further discussion of these capabilities is outside the scope of this document.

## 342 **4.3 Management Capabilities**

343 Once a patch management technology has been selected, its administrators should design a solution  
344 architecture, perform testing, deploy and secure the solution, and maintain its operations and security.  
345 This section highlights issues of particular interest with the management—the implementation, operation,

346 and maintenance—of patch management technologies, and provides recommendations for performing  
347 them effectively and efficiently.

### 348 **4.3.1 Technology Security**

349 Deploying enterprise patch management tools within an enterprise can create additional security risks for  
350 an organization; however, a much greater risk is faced by organizations that do not effectively patch their  
351 systems. Such tools usually increase security far more than they decrease security, especially when the  
352 tools contain built-in security measures to protect against security risks and threats. The following are  
353 some risks with using these tools:

- 354 ■ A patch may have been altered (inadvertently or intentionally).
- 355 ■ Credentials may be misused.
- 356 ■ Vulnerabilities in the solution components (including agents) may be exploited.
- 357 ■ An entity could monitor tool communications to identify vulnerabilities (particularly when the  
358 host is on an external network).

359 Organizations should reduce these risks through the application of standard security techniques that  
360 should be used when deploying any enterprise-wide application. Examples of countermeasures include  
361 the following:

- 362 ■ Keeping the patching solution components tightly secured (including patching them)
- 363 ■ Encrypting network communications
- 364 ■ Verifying integrity of patches before installing them
- 365 ■ Testing patches before deployment (to identify corruption)

### 366 **4.3.2 Phased Deployment**

367 Organizations should deploy enterprise patch management tools using a phased approach. This allows  
368 process and user communication issues to be addressed with a small group before deploying the patch  
369 application universally. Most organizations deploy patch management tools first to standardized desktop  
370 systems and single-platform server farms of similarly configured servers. Once this has been  
371 accomplished, organizations should address the more difficult issue of integrating multiplatform  
372 environments, nonstandard desktop systems, legacy computers, and computers with unusual  
373 configurations. Manual methods may need to be used for operating systems and applications not  
374 supported by automated patching tools, as well as some computers with unusual configurations; examples  
375 include embedded systems, industrial control systems, medical devices, and experimental systems. For  
376 such computers, there should be a written and implemented procedure for the manual patching process.

### 377 **4.3.3 Usability and Availability**

378 Organizations should balance their security needs with their needs for usability and availability. For  
379 example, installing a patch may “break” other applications; this can best be addressed by testing patches  
380 before deployment. Another example is that forcing application restarts, OS reboots, and other host state  
381 changes is disruptive and could cause loss of data or services. Again, organizations need to balance the  
382 need to get patches applied with the need to support operations. A final example, particularly important  
383 for mobile devices, is the acquisition of updates over low-bandwidth or metered connections; it may be

384 technically or financially infeasible to download large patches over such connections. Organizations  
385 should make provisions for ensuring that their enterprise patching solution works for mobile hosts and  
386 other hosts used on low-bandwidth or metered networks.

387

## 388 5. Metrics

389 As explained in Section 3.3 of NIST SP 800-55 Revision 1, *Performance Measurement Guide for*  
390 *Information Security* there are three types of measures:

- 391 • “Implementation measures are used to demonstrate progress in implementing security programs,  
392 specific security controls, and associated policies and procedures....
- 393 • Effectiveness/efficiency measures are used to monitor if program-level processes and system-  
394 level security controls are implemented correctly, operating as intended, and meeting the desired  
395 outcome....
- 396 • Impact measures are used to articulate the impact of information security on an organization’s  
397 mission....”

398 Regarding these types of measures, “less mature information security programs need to develop their  
399 goals and objectives before being able to implement effective measurement. More mature programs use  
400 implementation measures to evaluate performance, while the most mature programs use  
401 effectiveness/efficiency and business impact measures to determine the effect of their information  
402 security processes and procedures.” Accordingly, organizations should implement and use appropriate  
403 measures for their enterprise patch management technologies and processes.

404 Examples of possible implementation measures include:

- 405 • What percentage of the organization’s desktops and laptops are being covered by the enterprise  
406 patch management technologies?
- 407 • What percentage of the organization’s servers have their applications automatically inventoried  
408 by the enterprise patch management technologies?

409 Examples of possible effectiveness/efficiency measures include:

- 410 • How often are hosts checked for missing updates?
- 411 • How often are asset inventories for host applications updated?
- 412 • What is the minimum/average/maximum time to apply patches to X% of hosts?
- 413 • What percentage of the organization’s desktops and laptops are patched within X days of patch  
414 release? Y days? Z days? (where X, Y, and Z are different values, such as 10, 20, and 30)
- 415 • On average, what percentage of hosts are fully patched at any given time? Percentage of high  
416 impact hosts? Moderate impact? Low impact?
- 417 • What percentage of patches are applied fully automatically, versus partially automatically, versus  
418 manually?

419 Examples of possible impact measures include:

420 • What cost savings has the organization achieved through its patch management processes?

421 • What percentage of the agency's information system budget is devoted to patch management?

422

**Appendix A—Security Content Automation Protocol (SCAP) Tutorial**

423  
424  
425  
426

This appendix provides an overview of the Security Content Automation Protocol (SCAP) as it relates to enterprise patch management technologies. The appendix is based on material from NIST SP 800-117 Revision 1, *Guide to Adopting and Using the Security Content Automation Protocol (SCAP) Version 1.2*. Please see NIST SP 800-117 for additional information on SCAP.

427  
428  
429  
430  
431  
432  
433

SCAP (pronounced ess-cap), as expressed in NIST Special Publication (SP) 800-126, is “a suite of specifications that standardize the format and nomenclature by which software flaw and security configuration information is communicated, both to machines and humans.” SCAP is designed to organize, express, and measure security-related information in standardized ways, as well as related reference data, such as identifiers for software flaws and security configuration issues. SCAP can be used to maintain the security of enterprise systems, such as automatically verifying the installation of patches, checking system security configuration settings, and examining systems for signs of compromise.

434  
435

Table A-1 lists the component specifications for the SCAP version 1.2 protocol. The components are grouped by type:

436  
437

■ **Languages.** The SCAP languages provide standard vocabularies and conventions for expressing security policy, technical check mechanisms, and assessment results.

438  
439

■ **Reporting formats.** The SCAP reporting formats provide the necessary constructs to express collected information in standardized formats.

440  
441

■ **Enumerations.** Each SCAP enumeration defines a standard nomenclature (naming format) and an official dictionary or list of items expressed using that nomenclature.

442  
443  
444

■ **Measurement and scoring systems.** In SCAP this refers to evaluating specific characteristics of a security weakness (for example, software vulnerabilities and security configuration issues) and, based on those characteristics, generating a score that reflects their relative severity.

445  
446

■ **Integrity protection.** An SCAP integrity protection specification helps to preserve the integrity of SCAP content and results.

447

**Table A-1. SCAP Version 1.2 Component Specifications**

SCAP Component	Description
<b>Languages</b>	
Extensible Configuration Checklist Description Format (XCCDF) 1.2	A language for authoring security checklists/benchmarks and for reporting results of evaluating them
Open Vulnerability and Assessment Language (OVAL) 5.10	A language for representing system configuration information, assessing machine state, and reporting assessment results
Open Checklist Interactive Language (OCIL) 2.0	A language for representing assessment content that collects information from people or from existing data stores made by other data collection efforts
<b>Reporting Formats</b>	
Asset Reporting Format (ARF) 1,2	A format for expressing the exchange of information about assets and the relationships between assets and reports
Asset Identification	A format for uniquely identifying assets based on known identifiers and/or known information about the assets

SCAP Component	Description
<b>Enumerations</b>	
Common Platform Enumeration (CPE) 2.3	A nomenclature and dictionary of hardware, operating systems, and applications, plus an applicability language for constructing complex logical groupings of CPE names
Common Configuration Enumeration (CCE) 5	A nomenclature and dictionary of software security configurations
Common Vulnerabilities and Exposures (CVE)	A nomenclature and dictionary of security-related software flaws
<b>Measurement and Scoring Systems</b>	
Common Vulnerability Scoring System (CVSS) 2.0	A system for measuring the relative severity of software flaw vulnerabilities
Common Configuration Scoring System (CCSS) 1,0	A system for measuring the relative severity of system security configuration issues
<b>Integrity Protection</b>	
Trust Model for Security Automation Data (TMSAD) 1,0	A specification for using digital signatures in a common trust model applied to other security automation specifications

448  
449 Each of the SCAP components offers unique functions and can be used independently, but greater  
450 benefits can be achieved by using the components together. For example, the ability to have XCCDF  
451 documents that use CCE, CPE, and CVE identifiers with OVAL definitions to express rules and  
452 relationships for technical checks and that use OCIL questionnaires to express management and  
453 operational checks comprises the building blocks for *SCAP-expressed* checklists.<sup>6</sup> In other words, SCAP-  
454 expressed checklists use a standardized language (XCCDF) to express what checks should be performed  
455 (OVAL, OCIL), which platforms are being discussed (CPE), and which security settings (CCE) and  
456 software flaw vulnerabilities (CVE) should be addressed.

457 Both comprehensive SCAP-expressed checklists, such as a checklist to secure an operating system, and  
458 more specialized SCAP-expressed checklists are valuable. A specialized checklist can be used to check  
459 particular characteristics of systems to identify potential security problems. A common example is using  
460 an SCAP checklist to confirm the installation of patches and identify which patches are missing. SCAP-  
461 formatted data for patch checking can be made publicly available by software vendors for their products;  
462 organizations can download this data and use it through their SCAP-capable tools.<sup>7</sup>

<sup>6</sup> SCAP-expressed checklists are further defined in Table 4-1 of NIST SP 800-70 Revision 1.

<sup>7</sup> Patch information can be downloaded from the MITRE OVAL Repository at <http://oval.mitre.org/repository/>.

## 463 **Appendix B—Summary of Recommendations**

464 This appendix provides a summary of the main recommendations made throughout the publication.

### 465 **Section 3**

466 Section 3.1: If a vulnerability is not being exploited yet, organizations should carefully weigh the security  
467 risks of not patching with the operational risks of patching without performing thorough testing first.

468 Section 3.1: Organizations should carefully consider the relevant issues related to timing, prioritization,  
469 and testing when planning and executing their enterprise patch management processes.

470 Section 3.2: Organizations should identify all the ways in which patches could be applied and act to  
471 resolve any conflicts among patch application methods.

472 Section 3.2: Organizations should ensure that users cannot disable or otherwise negatively affect  
473 enterprise patch management technologies, and organizations should perform continuous monitoring of  
474 enterprise patch management technologies to identify any issues that occur.

475 Section 3.3: Organizations should carefully consider all alternative host architectures in use for the  
476 enterprise when designing enterprise patch management policies and solutions.

477 Section 3.4.1: The inventory of the patchable software (applications and operating systems) installed on  
478 each host should include not only which software is currently installed on each host, but also what version  
479 of each piece of software is installed.

480 Section 3.4.2: Organizations should ensure that their enterprise patch management can avoid resource  
481 overload situations.

482 Section 3.4.3: Organizations should be capable of detecting side effects, such as changes to security  
483 configuration settings, caused by patch installation.

484 Section 3.4.4: Organizations should use other methods of confirming installation, such as a vulnerability  
485 scanner that is independent from the patch management system.

486 Section 3.4.5: Organizations using application whitelisting technologies should ensure that they are  
487 configured to avoid problems with updates.

### 488 **Section 4**

489 Section 4.1: Organizations should carefully consider the advantages and disadvantages of each technique  
490 for identifying missing patches (e.g., agent-based, agentless scanning, passive network monitoring) when  
491 selecting enterprise patch management technologies.

492 Section 4.3: A patch management technology's administrators should design a solution architecture,  
493 perform testing, deploy and secure the solution, and maintain its operations and security.

494 Section 4.3.1: Organizations should reduce the risks of using enterprise patch management tools through  
495 the application of standard security techniques that should be used when deploying any enterprise-wide  
496 application.



497 Section 4.3.2: Organizations should deploy enterprise patch management tools using a phased approach.

498 Section 4.3.3: Organizations should balance their security needs with their needs for usability and  
499 availability.

500 **Section 5**

501 Section 5: Organizations should implement and use appropriate measures for their enterprise patch  
502 management technologies and processes.

503

504 **Appendix C—Acronyms and Abbreviations**

505 Selected acronyms and abbreviations used in the guide are defined below.

506	<b>ARF</b>	Asset Reporting Format
507	<b>CCE</b>	Common Configuration Enumeration
508	<b>CCSS</b>	Common Configuration Scoring System
509	<b>CPE</b>	Common Platform Enumeration
510	<b>CVE</b>	Common Vulnerabilities and Exposures
511	<b>CVSS</b>	Common Vulnerability Scoring System
512	<b>FISMA</b>	Federal Information Security Management Act
513	<b>IT</b>	Information Technology
514	<b>ITL</b>	Information Technology Laboratory
515	<b>NIST</b>	National Institute of Standards and Technology
516	<b>OCIL</b>	Open Checklist Interactive Language
517	<b>OMB</b>	Office of Management and Budget
518	<b>OVAL</b>	Open Vulnerability and Assessment Language
519	<b>SCAP</b>	Security Content Automation Protocol
520	<b>SP</b>	Special Publication
521	<b>TMSAD</b>	Trust Model for Security Automation Data
522	<b>XCCDF</b>	Extensible Configuration Checklist Description Format
523		