Heimdal®

2023

Patch Management Wireframing Guide

Your Journey to Business-Friendly Patch Management

Table of contents:

Introduction

Introduction

Patch management, or the process of distributing and applying update packages to a web of endpoints, is a quintessential step in addressing vulnerabilities associated with software design or unexpected system interactions. As to the importance of (timely) patching/updating in the context of active security vulnerability remediation, we should point out that more than 60% of cyber-aggressions that resulted in covert data exfiltration (i.e., data breaches) were associated or directly related to missing, misconfigured or partially deployed patches and/or updates.
To that end, it was discovered that

More than 50% of organizations are unable to patch critical vulnerabilities within the recommended time of 72 hours of their release, and around 15% of them remain unpatched even after 30 days.

In a Ponemon Institute-driven paper regarding the consequences and cost of tardive patch deployment, it has been brought to attention that the patching process can be, intentionally or unintentionally, delayed to up to 12 days due to characteristic technical issues, such as 'siloing' or turfing. The same study also raises the issue of the post-patching detection gap – on average, it takes about 45 days to 'pick up' residual, post-attack activity, even if the security patches are in place and correctly configured and deployed.

This eBook represents a patch management primer that will debate outstanding and emergent patching practices, the value of early patching in vulnerability mitigation, patch management frameworks, deployment, and post-deployment issues, and how early patch policy enactment can dramatically reduce the costs associated with a cybersecurity attack.

Patch Management Concept

Patch Management Concepts

Automated patch deployment – an automatized and multi-step process designed to distribute update packages to multiple endpoints across a business environment with minimal human intervention.

Generic patch testing – a standardized set of tests performed prior to or post-patch deployment.

Post-deployment verification – test-based investigation aimed at determining correct patch deployment, software integrity, and stability.

Patch information retrieval – data-gathering process to determine the utility of a patch within a given patching context or software state.

Patch process governance – process to designate or delegate person(s) responsible for various path-related roles (e.g., vulnerability scanning, vulnerability database curation, generic patch testing, post-deployment verification, etc.)

Principle of Least Privilege (P.O.L.P) – IAM/PAM concept stating that all users should access only the resources or assets they need in order to perform their tasks.

Security patch –software designed to correct known or recently identified security flaws of a computer program.

Vulnerability – in the context of an information system, a vulnerability is a weakness, flaw – either by design or spawned arbitrarily – that can be leveraged by threat actors.

Vulnerability management – a systematic approach to vulnerability identification, enumeration, analysis, and remediation based on internal protocol, procedures, controls, tools, and policies.

Vulnerability scanner – computer program capable of scanning devices and networks for vulnerabilities.

Vulnerability database – a local- or cloud-hosted collection of known and newly discovered software and/or hardware vulnerabilities.

An Introductory View of Patch Management

An Introductory View of Patch Management

In the context of complementary cybersecurity, patching has become vital in threat-centric prophylaxis. Statistics indicate that over 60% of software and OS-related vulnerabilities can be 'fixed' through periodical patching and updating. However, taking into consideration the various challenges associated with the patching process (i.e., number of patches per pre-determined timeframe, versioning operating system, group policy limitations, bandwidth allocation, cross-compatibility issues, build stability, post-deployment problems, etc.), a systematic methodology is warranted. Patch management is the palpable solution, having brought the tools, flow mapping, and knowledgebase necessary to retrieve, deploy, test, and benchmark patches.

Critical to any patch management approach is the ability to identify and categorize the various types of patches available, based on function, a vulnerability severity rating, and deployment difficulty. This section of the ebook concerns the terminology employed in patch management to describe and differentiate between the various improvement-carrying packages.

Terminology of Security and Non-Security Improvements Managed Within A Generic Business Ecosystem

- **Critical update**

Definition: an improvement-carrying computer program poised to solve a single, critical issue. Critical updates are regarded as non-security.

- **Definition update**

Definition: an improvement-carrying computer program released periodically designed to affix definitions to the definition database.

- **Driver**

Definition: a computer program designed to control the output and the input of a specific device or device class.

- **Feature pack**

Definition: a collection of product improvements that are usually distributed after the product's release.

- **Security update**

Definition: an improvement-carrying computer program designed to address a specific (security) vulnerability.

- **Service pack**

Definition: a collection of tested security updates, updates, critical updates, and hotfixes
.
- **Tool**

Definition: computer program (i.e., feature, utility, tool collection) that allows the user to finish one or more tasks.

- **Update**

Definition: an improvement-carrying computer program deployed on a large scale to address bugs that aren't marked as critical or security-related.

- **Update rollup**

Definition: a collection of improvement-carrying computer programs consisting of updates, hotfixes, security updates, and critical updates. Update rollups are usually bundled for quick deployment.

- **Security-only update (SSU)**

Definition: a collection of security-related updates packaged for easy deployment via Microsoft Update Catalog, SCCM, or WSUS.

- **Monthly rollup**

Definition: a collection of security and non-security updates packaged for quick deployment via Windows Update, WSUS, SCCM, or Microsoft Update Catalog.

- **Preview of monthly rollup**

Definition: a collection of new updates packaged for easy deployment via WSUS, Windows Update, SCCM, or Microsoft Update Catalog.

- **Servicing Stack Updates (SSU)**

Definition: improvement-carrying packages for the servicing stack, the code that controls OS updating and other Windows deployment features such as repairing, changing Win features and/or roles, SFC (System File Checker), and DISM (Deployment Image Servicing and Management Tool).

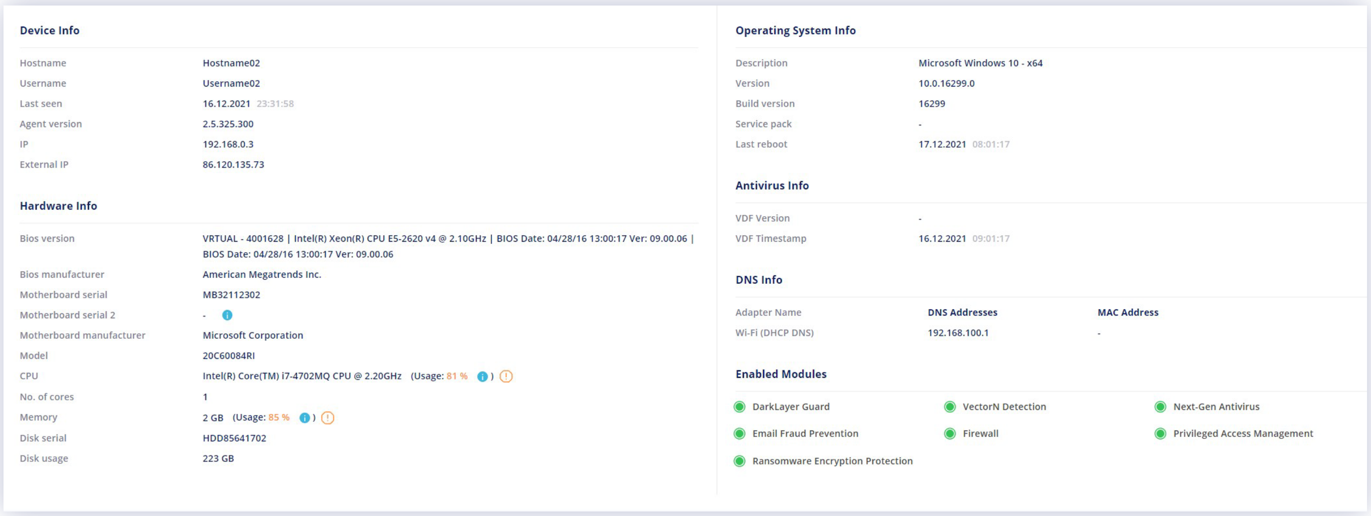
Patch Management Flow (Abridged)

Patch Management Flow (Abridged)

Clearly, any patch management flow, whether a WSUS/SCCM or third-party-mediated one (i.e., employing a Microsoft-independent patch management architecture or framework) must conform to a set of rules that govern the patching process as a whole. Those steps are summarized below:

1) Asset & Software Inventory

Asset and software inventory are the first steps towards 'healthy' vulnerability management. Using specialized tools (i.e., automated device discovery software), the person assigned to the task (i.e., usually an IT) gathers info on the number, type, running/stable configurations of all devices owned and operated by the company's employees. Asset inventory can be as 'basic' as enumerating device numbers and types (i.e., laptop, desktop computer, tablet, smartphone, physical firewall, WAPs, servers, router, switch, hub, etc.) or as intricate as advanced hardware probing tools often are. To the latter statement, we have reproduced a screenshot from Heimdal Patch & Asset Management's hardware view section to highlight the probing feature.



Device Info

Hostname	Hostname02
Username	Username02
Last seen	16.12.2021 23:31:58
Agent version	2.5.325.300
IP	192.168.0.3
External IP	86.120.135.73

Hardware Info

Bios version	VRTUAL - 4001628 | Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz | BIOS Date: 04/28/16 13:00:17 Ver: 09.00.06 | BIOS Date: 04/28/16 13:00:17 Ver: 09.00.06
Bios manufacturer	American Megatrends Inc.
Motherboard serial	MB32112302
Motherboard serial 2	-
Motherboard manufacturer	Microsoft Corporation
Model	20C60084RI
CPU	Intel(R) Core(TM) i7-4702MQ CPU @ 2.20GHz (Usage: 81 %)
No. of cores	1
Memory	2 GB (Usage: 85 %)
Disk serial	HDD85641702
Disk usage	223 GB

Operating System Info

Description	Microsoft Windows 10 - x64
Version	10.0.16299.0
Build version	16299
Service pack	-
Last reboot	17.12.2021 08:01:17

Antivirus Info

VDF Version	-
VDF Timestamp	16.12.2021 09:01:17

DNS Info

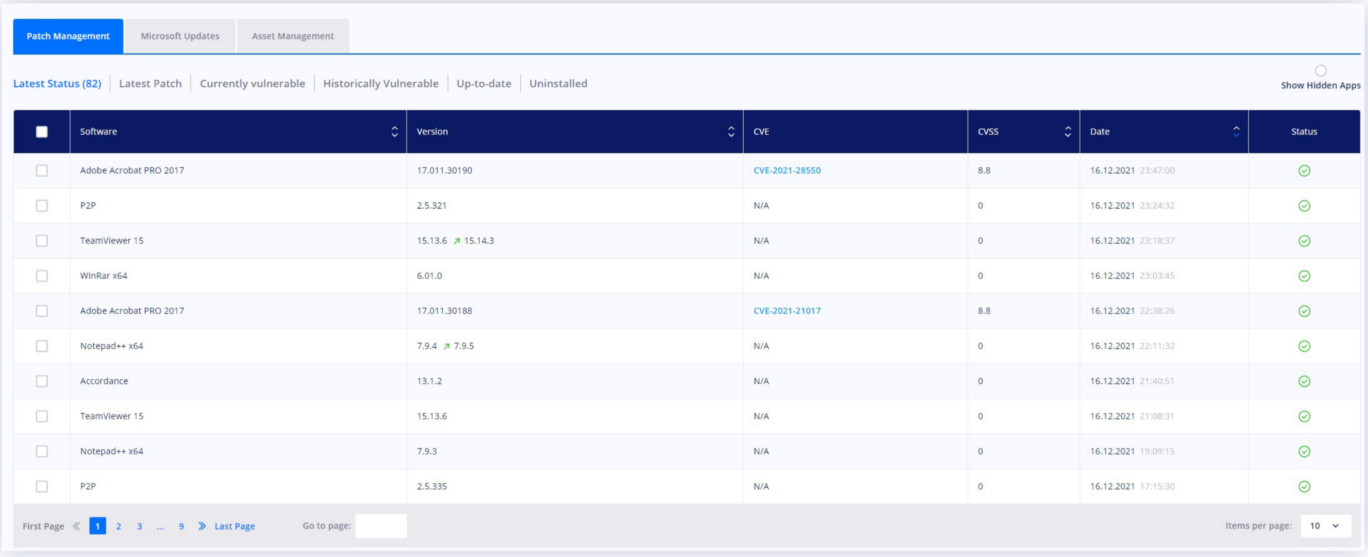
Adapter Name	DNS Addresses	MAC Address
Wi-Fi (DHCP DNS)	192.168.100.1	-

Enabled Modules

● DarkLayer Guard	● VectorN Detection	● Next-Gen Antivirus
● Email Fraud Prevention	● Firewall	● Privileged Access Management
● Ransomware Encryption Protection		

Fig 1. Unified Threat Platform hardware view

Evidently, an exhaustive query into the asset(s) are can yield actionable results – what requires patching, stable vs. non-stable software versions, potential compatibility issues, items requiring immediate attention, etc. In line with the hardware view screenshot, we have reproduced an additional one from the Software Inventory view (see image below).


Patch Management | Microsoft Updates | Asset Management

Latest Status (82) | Latest Patch | Currently vulnerable | Historically Vulnerable | Up-to-date | Uninstalled

Show Hidden Apps

	Software	Version	CVE	CVSS	Date	Status
☐	Adobe Acrobat PRO 2017	17.011.30190	CVE-2021-28550	8.8	16.12.2021 23:47:00	✓
☐	P2P	2.5.321	N/A	0	16.12.2021 23:24:32	✓
☐	TeamViewer 15	15.13.6 ↗ 15.14.3	N/A	0	16.12.2021 23:18:37	✓
☐	WinRar x64	6.01.0	N/A	0	16.12.2021 23:03:45	✓
☐	Adobe Acrobat PRO 2017	17.011.30188	CVE-2021-21017	8.8	16.12.2021 22:38:26	✓
☐	Notepad++ x64	7.9.4 ↗ 7.9.5	N/A	0	16.12.2021 22:11:32	✓
☐	Accordance	13.1.2	N/A	0	16.12.2021 21:40:51	✓
☐	TeamViewer 15	15.13.6	N/A	0	16.12.2021 21:08:31	✓
☐	Notepad++ x64	7.9.3	N/A	0	16.12.2021 19:09:15	✓
☐	P2P	2.5.335	N/A	0	16.12.2021 17:15:30	✓

First Page « 1 2 3 ... 9 » Last Page   Go to page: Items per page: 10

Fig 2. Unified Threat Platform software (all) view.

2) Risk assessment and Patch Prioritization

RA (risk assessment) is a score-assignation process based on either industry-agreed criteria such as CVE, CVSS, CERT, CEW, OWASP, NVD, or DISA STIG or convention (i.e., companies may create industry-independent metrics and risk scores based on internal policies, protocols, applicability, availability or relevance). Based on (evaluated) severity, the delegated person can append a priority score – high, medium, low, requesting immediate attention.

3) Scenario Replication and Testing

Refers to the process of designing and 'building' a suitable environment for patch-testing purposes. Although virtualized environments are preferable, in some cases, the behavioral output may be different from 'real-world' outcomes. An empirical approach to patch testing shows that evaluations performed under non-critical systems (i.e., real-time) yield far better results. The patch testing process must also factor in various, field-encountered scenarios (e.g., for a patch addressing authentication bugs, a commonly used test case would be to verify the login mechanism in case of a correct or incorrect username and password. In an alternative scenario, the IT admin can assess the output of the login mechanism when the user inputs the correct username, but the wrong password). Patch testing and scenario replication have to conform to the requirements of the environment it is serving.

4) Stability Assessment

Test-stressing patching in 'real-world' scenarios includes the stability evaluation – a metric that is used to determine the viability of a soon-to-be-deployed improvement (i.e., application uptime vs. downtime pre- and post-patch-deployment, compatibility, resource consumption). Any type of stability-related test must be performed by a security team. Keep in mind that we're still in the pre-deployment phase – stability tests are performed on patches deployed within the chosen patch testing environment.

5) Monitorization

The security team continuously monitors the patches deployed inside the non-critical environment to further determine viability.

6) Backup

First of three steps to patch deployment. The critical areas are backed up and additional tests are performed on the backups and associated mechanisms, procedures, and tools to ensure data integrity. Backup applies to both apps and app configurations, regardless of state.

7) Configuration Management

Second of three steps to patch deployment. Using scriptural or CM (Config Management) tools, the person or persons delegated must document config state and subsequent changes. Should any unexpected output occur, the document can be consulted to isolate and fix the component.

8) Deployment

Last step of the patch deployment process. The procedure is as follows: review configuration management documentation, ensure backup storage consistency, and begin patch deployment. Remember that this is the 'point of no return'; if everything checks out, you can start patching critical software and hardware assets. The patch management 'rule of the thumb dictates that the patching process of the critical infrastructure should occur after business hours so it does not hinder normal ops.

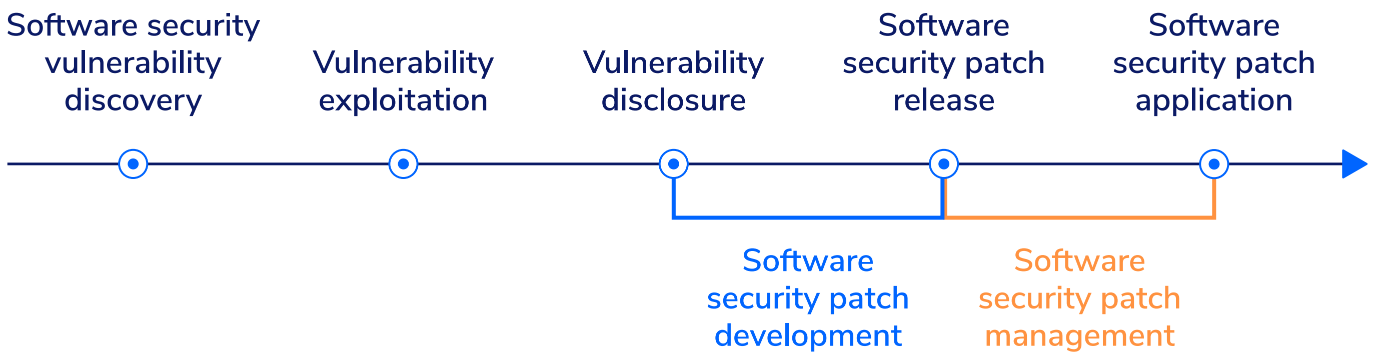
9) Maintenance

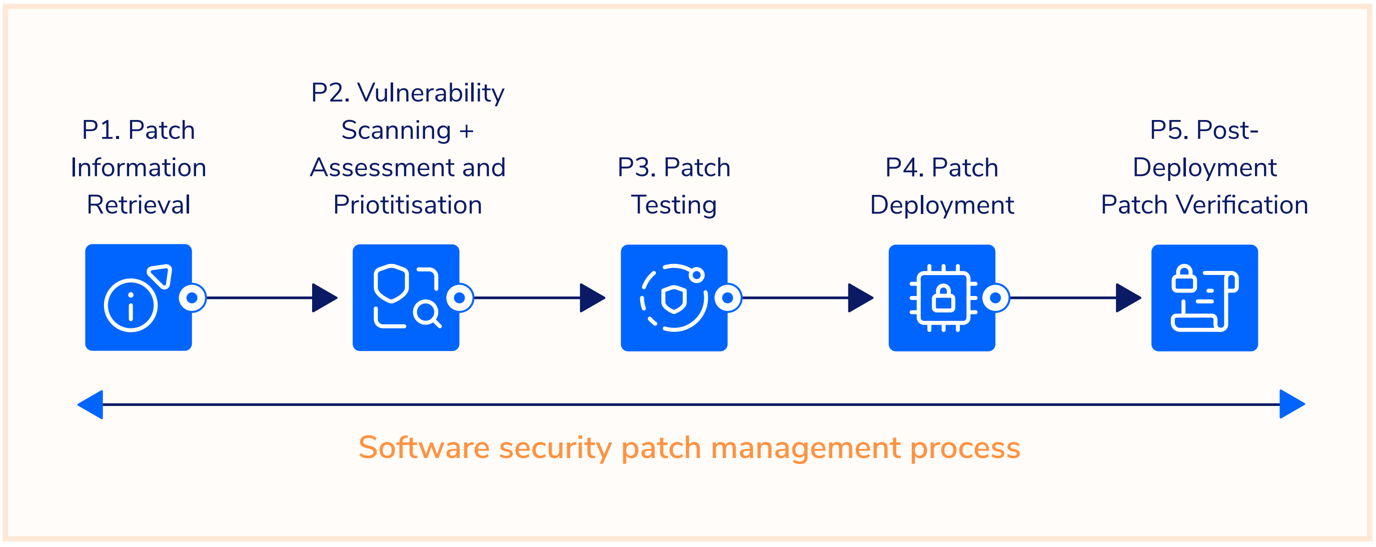
Post-deployment procedures should also factor in monitorization & maintenance. The former is the natural, applicable extension of step 5), while the latter focuses on patch and application health.

10) Documentation

End-to-end patch management process documentation – this aggregate should detail every step of the process, from software & hardware inventory to post-deployment.

The above-described patch management flow is summarized in the picture below:


Software security vulnerability discovery
Vulnerability exploitation
Vulnerability disclosure
Software security patch release
Software security patch application
Software security patch development
Software security patch management


P1. Patch Information Retrieval
P2. Vulnerability Scanning + Assessment and Priotitisation
P3. Patch Testing
P4. Patch Deployment
P5. Post-Deployment Patch Verification
Software security patch management process

Patch Management Checklist

Patch Management Checklist

In preparation for your patch management process, Heimdal provides an extensive checklist detailing the controls, areas, devices, and procedures that should be followed in pre-and post-patch deployment.

Patch Management Assessment Questionnaire

1) What is the coverage of your procedure? Does this procedure cover business-owned network infrastructure, applications, servers, computers, and systems?
☐ Yes ☐ No

2) Role assignation – have you identified and delegated a system owner or team to oversee the patching process?
☐ Yes ☐ No

3) Are there any security implications associated with the process?
☐ Yes ☐ No

4) Have you obtained/acquired the patches from a trusted source or vendor?
☐ Yes ☐ No

5) Have you performed any security tests on the obtained patches?
☐ Yes ☐ No

6) Have you commenced the pre-patch deployment backup process?
☐ Yes ☐ No

7) Have you begun the auditing process?

☐ Yes ☐ No

8) Does your auditing document include changes before and after patch deployment?

☐ Yes ☐ No

9) Have you established timelines, priority levels, and potential business impact for internet-facing devices?

☐ Yes ☐ No

10) Have you established timelines, priority levels, and potential business impact for non-internet-facing devices?

☐ Yes ☐ No

11) Have you established timelines, priority levels, and potential business impact for laptops and desktop computers?

☐ Yes ☐ No

12) Have you established timelines, priority levels, and potential business impact for network devices?

☐ Yes ☐ No

13) Have you appointed an error and exception handling team?

☐ Yes ☐ No

14) Have you established policies for the error and exception handling team?

☐ Yes ☐ No

15) Have you appointed a team to handle patch enforcement?

☐ Yes ☐ No

16) Is your patch enforcement team correctly and timely informed about policy violations and punitive actions?

☐ Yes ☐ No

17) Compliance. Check the patch management security regulatory compliances that apply to your business:

- ☐ FTI (Federal Tax Information);
- ☐ NIST (National Institute of Standards and Technology);
- ☐ FERPA (Family Educational Rights and Privacy Act);
- ☐ SOX (Sarbanes-Oxley);
- ☐ GLBA (Gramm-Leach-Blliey);
- ☐ FFIEC (Federal Financial Institutions Examination Council);
- ☐ GDPR (European Union General Data Protection Regulation);
- ☐ PCI-DSS (Payment Card Industry Data Security Standard);
- ☐ HIPAA (Health Insurance Portability and Accountability Act);
- ☐ MS-ISAC (Multi-State Information Sharing and Analysis Center);
- ☐ Department of Homeland's Security US-CERT.
- ☐ Other(s).........;

18) Risk assessment. Items:

- ☐ Organization data impact.
- ☐ The number of impacted systems.
- ☐ Types of systems.
- ☐ Vulnerability type(s).
- ☐ Attack vector specificity.

19) Establishing your CMP (Configuration Management Plan). Check the items that apply to your business:

- ☐ Information system details.
- ☐ Component inventory.
- ☐ Configuration items.
- ☐ Configuration modifications.
- ☐ Roles. Define roles.
- ☐ Beneficiaries. Define beneficiaries.
- ☐ Policies related to new configurations.
- ☐ Access Controls.
- ☐ Tracking Configuration Management artifacts.
- ☐ Identifying secure configurations.
- ☐ Establishing baseline configurations.
- ☐ Error and exceptions handling.

20) Component inventory. Check the items that apply to your business' patch management process.

- ☐ SN (serial number) or UI (unique identifier).
- ☐ Information system association.
- ☐ Nature of component.
- ☐ Model information.
- ☐ Operating System.
- ☐ Types of virtual machines or containers.
- ☐ Software versioning.
- ☐ Type of Licensing.
- ☐ Ownership.
- ☐ Status.
- ☐ Primary user.
- ☐ Administrator.
- ☐ Atomic indicators (e.g., MAC address, UP address).
- ☐ Location (on-site or remote).

21) Patching priority. Check the items that apply to your business' patch management process:
- ☐ Servers (DMZ).
- ☐ Internal Servers.
- ☐ Workstations.
- ☐ Perimeter defenses.
- ☐ Internal network defenses.

22) User-side notifications. Check the items that apply to your business' patch management process:
- ☐ Patch impact (i.e., the user should receive notification regarding the patch's impact on the system or application subjected to this process).
- ☐ Timeline indicators (i.e., the user should be briefed about the date and time the patching process will occur).
- ☐ Feedback (i.e., the user should be provisioned with means of expressing concerns or a way to notify the administrator about patch-related issues).
- ☐ Reason for patch process termination (i.e., the user should be informed in a timely manner regarding the reasons behind patch process termination).

23) Have you performed extensive testing on patches before deployment?

☐ Yes ☐ No

24) Are you actively monitoring the applied patches?

☐ Yes ☐ No

25) What metrics are you using to gauge application performance before and after patching? Please specify.

____

Patch Management Frameworks - ITIL & NIST.

Patch Management Frameworks - ITIL & NIST.

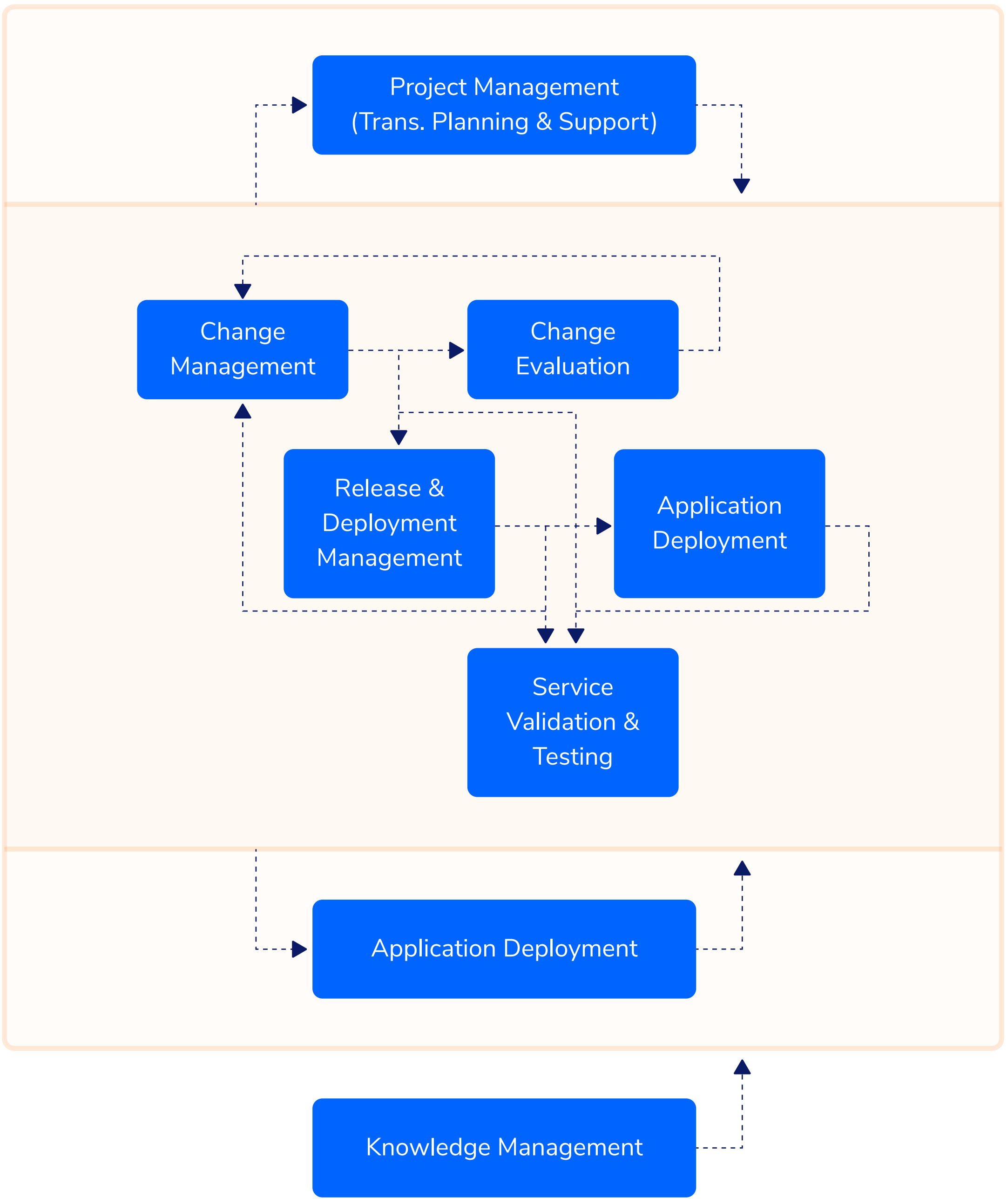
This chapter will examine some of the most prominent frameworks: ITIL (IT Infrastructure Library) and NIST (National Institute of Standards and Technology recommendations for handling security patches), both consistent with unmediated (experience-based) patch management.

ITIL Framework

ITIL version 4, the 'modernized' IT Service life-cycle, features 34 practices or procedures that cover the entire IT surface, from service strategy that dictates customer-facing strategies to CSI (i.e., Continual Service Improvement), the QAM (Quality Assurance Management) of IT. For the purpose of this E-Book, we will concentrate on Service Transition, ITIL's regulatory policy body governing the creation and deployment of new or modified services. The Service Transition procedure features 8 main processes and numerous sub-processes. They are as follows:

- Change(s) Management – focused on researching and implementing beneficial changes, by minimizing system impact and disruptions.
- Change(s) Evaluation – evaluation methodology that empowers IT admins to measure the impact and quality of a newly introduced or modified service as well as determining if a specific service or branch requires improvement.
- Application Development – a methodology (and praxis) focused on planning, creating, and benchmarking new applications or functionality extensions for one or more IT services.
- Release and Deployment Management - control live and virtual testing environments in addition to service planning, scheduling, and management.
- Service Validation and Testing – contains the methodology required to gauge the impact of deployed services.

- Service Asset and Configuration Management – documentation trail pertaining to the configuration states of deployed objects or services.
- Knowledge Management – the necessary framework to develop a functional and easily accessible knowledgebase that can be further used to develop new services, improve existing ones, benchmark or debug recently deployed services.


Project Management (Trans. Planning & Support)
Change Management
Change Evaluation
Release & Deployment Management
Application Deployment
Service Validation & Testing
Application Deployment
Knowledge Management

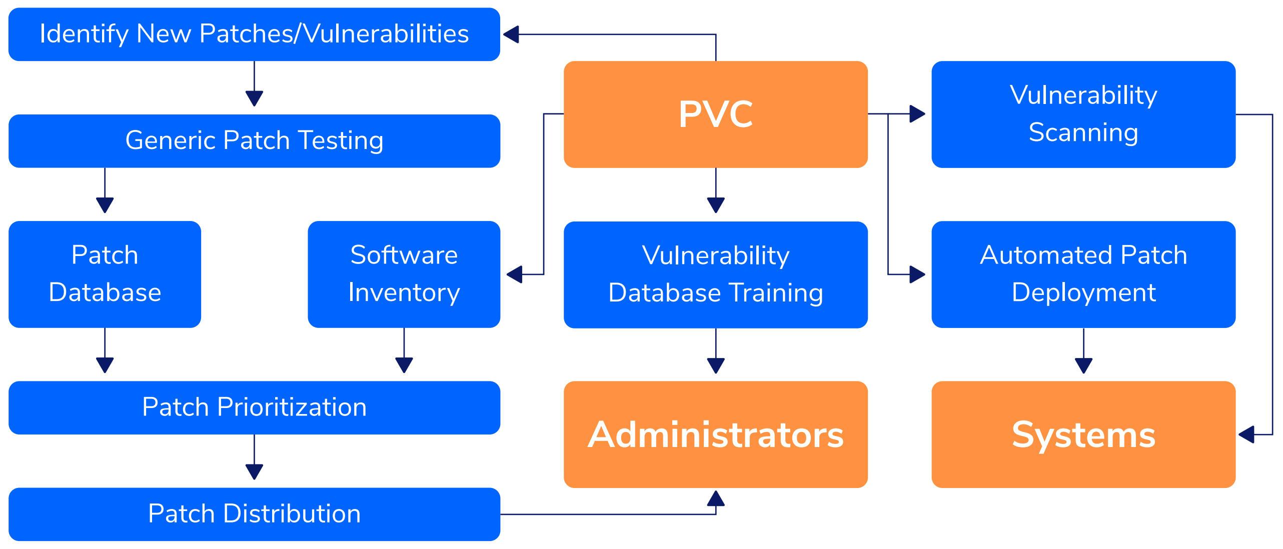
In relation to the generic framework outlined in this e-book's introductory chapter, ITIL further details the policies, controls, and processes vital to patch management. The Service Validation Testing process under ITIL's Service Transition flow further branches into four sub-processes or objectives, each related to how validation and testing are integrated into the patch management flow. For instance, testing occurs according to the model, an approach that details how the candidate should be probed and what criteria should be imposed for quality. After agreeing on the test model, specific components of a release are isolated and assessed. Only those who pass this evaluation can be subjected to further tests.

If results are within acceptable testing parameters, the QAed components, controls, and tools will become part of the deployment phase. Release candidates are also subjected to rigorous evaluation under the Release and Deployment Management process, which further branches into several other sub-processes, each governing over a specific release & deployment phase (e.g., release management support, planning, release build, release deployment, early life support, and release closure or termination). Finally, ITIL's Service Transition flow has numerous provisions in the area of Service Asset and Configuration Management. A brief glance at this sub-branch reveals a multitude of controls that govern everything from configuration identification and audit to configuration items, medial library, and CMDB (Configuration Management Database curation).

NIST Framework

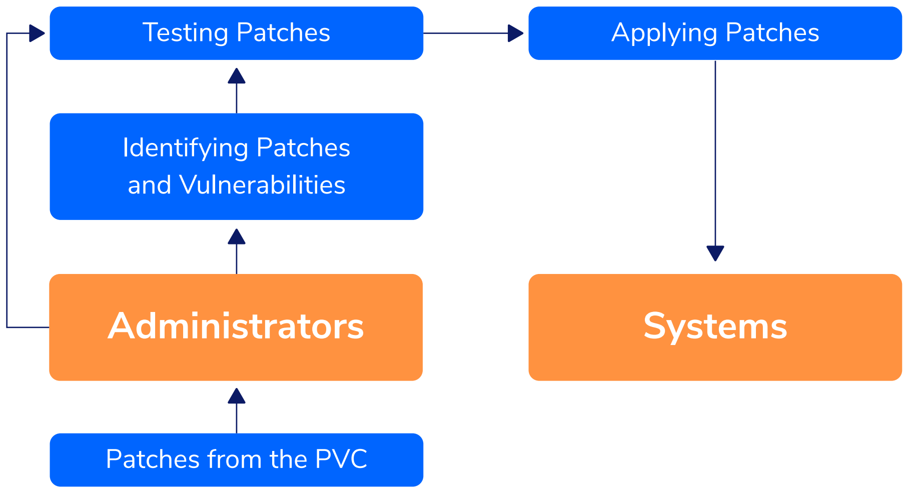
The fundamental difference between ITIL and NIST is that the former primarily focuses on patch management quality and flow mapping while the latter on the security side. NIST's 'go-to' methodology is PVG (i.e., Patch and Vulnerability Group), a wireframe that alloys patch deployment, assets & software inventory with vulnerability discovery and management. Another important fact is that NIST's approach also refers to the person or persons responsible for this type of audit. Another key difference between the two is that the newly created Patch and Vulnerability Group is not directly charged nor responsible for implementing changes. The group's mandate is to identify and label vulnerabilities in order to create a benchmarking baseline for the IT personnel responsible for carrying out these tasks. To further understand the dynamics of the two groups (i.e., PVG and IT persons), we have outlined below their responsibilities:

PVG should: generate and continuously curate a software and hardware inventory, seek out new vulnerabilities and, if possible, obtain associated security patches, assign priority levels to each security patch, generate, enrich, and maintain a security patch database, subject newly-discovered patches to security-relevant tests, make security patches and vulnerability-related information available to the IT personnel, supervise the deployment and installation processes, conduct host vulnerability scanning, train the IT personnel in vulnerability databases, supervise and/or perform automatic security patch deployment, and supervise and/or perform auto-app updates.



IT personnel (admins) should: review and enforce the security patches dispatched by PVG, conduct extensive testing on the relayed security patches, and conduct their own vulnerability scanning.

The relationship between administrators and members of the Patch & Vulnerability Group is summarized in the image below.



One cannot but remark the (obvious) symbiosis between the Patch and Vulnerability Group and the IT admins; this organic approach to patch management solves two major aspects:

- Ensuring that the security patches are safe and that all vulnerabilities are mapped out beforehand;
- Deploying the security alongside the non-security patches by leveraging a clearly defined system.

In some regards, NIST's foray into patch management can be considered ITIL's complement.

Designing a Patch Management Policy

Designing a Patch Management Policy

Based on the various patching and security aspects we've discussed so far, we can begin to draft out a program, a wireframe that serves a business-specific need. The absolute baseline for this is the Vulnerability and Patch Management Program, a compliance-oriented, framework-independent document, created and maintained by Compliance Forge. Below, you can find the abridged version of the VPMP.

VPMP (Vulnerability and Patch Management Program)

Outline:

- Scope.
- Common and uncommon vulnerabilities. Dependencies. Roles and responsibilities.
- Risk assessment and treatment. Considerations: black risk, gray risk, white risk, 0-day patches, and 0-day exploits.
- Vulnerability management governance.
- Vulnerability analysis.
- Sys and app patching.
- Scanning & Pen Testing.

Your company's version of the VPMP should reflect both security and patch-management flow-related features, pros, cons, limitations, and opportunities. The detailed VPMP-relevant version of this document is outlined below.

1) Scope

To include both network and geo-locations. This section should include company-owned assets & locations and third-party-owned assets & locations. E.g., corporate networks, e-shops, telecommunication infrastructure, physical infrastructure, 3rd party service providers, and BYODs.

2) Common and uncommon vulnerabilities. Dependencies. Roles and responsibilities.

Enumerate technical and non-technical-related vulnerabilities. E.g., tech vulnerabilities – misconfigured software, lack or insufficient encryption, blacklisted applications, access permissions, open ports, etc. Non-technical-related vulnerabilities – subpar business recovery planning, lack of data backup or recovery policies, lack of adequate software and hardware inventory, etc.

3) Risk assessment and treatment. Considerations: black risk, gray risk, white risk, 0-day patches, and 0-day exploits.

The RA framework should have a clear understanding of an exploitation's timeline to better formulate a security patch deployment plan. In practice, vulnerabilities that carry a black-type risk (i.e., the time between vulnerability discovery and becoming public) fall into the 0-day day exploits category and should be addressed with the utmost urgency. Unfortunately, black-type risks can be mitigated, not solved, since there's insufficient time to develop a patch or a fix for these specific issues. Another timeline-specific metric is the so-called gray risk – a temporal area covering the interval between public disclosure and the creation of a patch capable of solving that specific issue. Note that both black and gray areas carry a certain amount of exposure risk since there are insufficient resources or information available to address the issue.

The last timeline-specific metric is a white risk, defined as the temporal area between the fix's release and fix deployment. Most companies tend to disregard this detection gap, resulting in tardive patch deployment – an issue on its own with dire consequences both in terms of 'operativeness' and in increasing the risk of backdooring, malicious eavesdropping, and persistence.

Although these timelines and associated steps are not set in stone, one should always exercise caution when applying patches, regardless of their nature.

For instance, IT administrators, at the bequest of PVGs or the company's security team, will prioritize security patches for zero-day exploits and/or discovered vulnerabilities over operational and non-security-related patches. Although benign and endorsed, the process in itself can carry a certain amount of risk, meaning it can have a negative impact on the infrastructure. Patches should always be thoroughly tested by both PVGs and IT admins before being released for immediate deployment.

4) Vulnerability management governance.

Vulnerability analysis. Appoint security teams to identify, manage, and come up with resolutions for identified security vulnerabilities. For the best results, you should appoint a vulnerability management team for identity, scanning, and results gauging purposes, an IT operations team to implement, supervise, monitor, test, and maintain configurations (i.e., before and after patch management deployment). Additionally, the asset owners (i.e., users) should be fully briefed on the patch management process, regardless if it's routine or extraordinary.

5) System and application patching.

Scanning & Pen Testing. Rolling out both security and non-security related should be made in a controlled manner and according to your patch management flow. If your company operates under the NIST-PVG framework, both IT ops and the security team must provide test results for patches before and after deployment. Roll-backs are possible at this point, so it is crucial for IT ops to continuously monitor the patch deployment process. Post-deployment operations include periodical penetration testing (i.e., task delegated to the security team or a company-independent contractor) and scanning for vulnerabilities that may occur as a result of the patching process.

Patch Management Best Practices

Patch Management – Best Practices

As a whole, the patch management process goes beyond rules, regulations, policies, and procedures. Essentially, we should patch with the best practices in mind instead of taking the 'rulebook' for granted. To be more precise, the practice has shown countless times that wireframes and frameworks cannot be fully applied to real-life situations and scenarios. The purpose and intent of this e-book are to familiarize the reader with the various controls, actions, and functions of a 'healthy' patch management system; beyond this point, we shall allow the cumulated experience to speak for itself. Below, you shall find the best patch management practices that will further help you in defining your company's patching flow, discover new key points or raise new issues.

1) Incident Management

- IM (Incident Management) should define the approaches to product remediation and/or restoration in case of damage (i.e., design flow, unexpected system interaction, stability issues).
- Early outlining of procedures and policies. Patches addressing zero-days or newly discovered vulnerabilities should have priority over non-security-related patches.
- A clear understanding of IM flow. Ensure that your incident management flow factors in the necessary tools and controls requires to identify, detect, respond, protect, and, if possible, recover a software component in case of a major incident (e.g., data breach).
- Database curation. IM-related information should be stored in a separate database. Ensure that info is properly cataloged, regularly backed up, and archived.
- In-house vs. 3rd party testing and patch creation. When formulating your IM plan, take into account the financial aspect – cost vs. benefits of testing and creating patches in-house or delegating this task to a third party.

2) Configuration Management

- CM (Configuration Management) should thoroughly document changes to the configuration before and after patch deployment.
- When drafting a CM ensure that the following items are included: timeline, contingency plan, justification, risk evaluation, dependencies, expected system(s) downtime, data sensitivity, impact on running systems, and roles.
- Stakeholder communication. Ensure a stable line of communication (e.g., electronic mail, instant messaging, telephony, SMS, etc.) between the individual or individuals enforcing changes and the stakeholders and/system owners.
- Contingency plan. Carefully formulate a contingency plan. This should contain procedures, controls, and policies for solving errors or unexpected outputs during the patch deployment process. Factor in rollbacks, multi-stage backups, and medial testing (i.e., testing the stability of a patch or service modification in the interval between post-sandboxing and deployment). Frequently review the changes on the systems and write them down in your Configuration Management tool.

3) Service-level Management (SLM)

- SLM (Service-Level Management) is an approach aimed at monitoring, reporting, and offering support to SLAs (Service-Level Agreement) beneficiaries.
- Properly configure SLA Properties. Review configuration for engine, logging module, the tools required to carry out SLA-related repairs.
- Keep your SLA records up to date. Review SLA conditions for reset, pause, stop and start actions, timelines, task recorder, and repair log.

4) Issue Management

- Commercial vs. in-house applications. Some 3rd party apps may have compatibility or stability issues. Intensive testing should help you route any of these issues before patch deployment.

- Rollbacks. Backups provide you with the means to roll back patched applications or changed configurations.
- Unexpected downtimes. There may be slight modifications to your predicted timeline, especially when addressing expected errors. Be sure to specify in your stakeholders/users' communications that these 'hiccups' may occur.

5) Service Continuity Management

- SC (Service Continuity) is a process that allows IT administrators to minimize disruptions and impact on operational services.
- BCPs vs. BIAs. Business Continuity plans focus on standardized flows business disruption flows. As part of your BCP, you should enact a Business Impact Analysis (BIA), a framework that helps you identify and respond to any disruption that could negatively impact your infrastructure.

6) Change Management

- Change management provides the procedures, policies, and controls necessary to identify and track all configuration changes before, after, and during patch deployment.
- All changes to configuration or machine states should be penned down in a change management document.
- All changes to configurations or machines should be conveyed to security teams and stakeholders.

7) Availability Management

- AM (availability management) is the process of identifying and acquiring security and non-security patches.
- Conduct research on system and business impact before making the patch acquisition case.

8) Release Management

- RM (Release Management) gauges system changes occurring after the deployment of one or more patches.
- Your RM should factor in testing (before and after patch release), rollback strategies in case of an unexpected error, and drafting accurate patch timelines.

9) Financial Management

- Patch-related financial considerations (i.e., cost of in-house vs. 3rd party patch development, the cost associated with path dismissal, etc.) should be thoroughly documented.

Who We Are and What Can We Do for Your Business?

★★★★★ ★★★★★ ★★★★☆ ★★★★☆ ★★★★☆ ★★★★☆

Who we are and what can we do for your business?

Founded in 2014 in Copenhagen, Denmark, Heimdal™ is a leading European provider of cloud-based cybersecurity solutions. The company offers a multi-layered security suite that combines threat prevention, patch and asset management, endpoint rights management, and antivirus and mail security which together secure customers against cyberattacks and keep critical information and intellectual property safe.

Heimdal has been recognized as a thought leader in the industry and has won multiple international awards both for its solutions and for its educational content creation.

Currently, Heimdal's cybersecurity solutions are deployed in more than 45 countries and supported regionally from offices in 15+ countries, by 175+ highly qualified specialists. Heimdal is ISAE 3000 certified and secures more than 2 million endpoints for over 10,000 companies.

Heimdal supports its partners without concessions on the basis of predictability and scalability. The common goal is to create a sustainable ecosystem and a strategic partnership.

Heimdal Patch & Asset Management product was engineered to go beyond the automatic patch management frameworks laid down by WSUS and SCCM. Its primary area of expertise is delivering hardware and software inventory without relying on third-party tools or APIs. Patch & Asset Management supports both Microsoft & 3rd party applications.

In addition, our solution makes available a benchmarking and testing environment for proprietary applications and software.

Advanced scripting features allow the IT administrator to globally enforce security and non-security patches, which are delivered through Heimdal Security's CDN via micro-HTTPS downloads.

All patches and updates stored in our repositories are tested before being made available. Heimdal Patch & Asset Management can easily help you set up new machines, commit changes on existing ones or curb the users' access to sensitive system areas. Ours is the promptest market-to-endpoint timeline – any patch, update, or hotfix received by Heimdal is delivered to the end-user in under four hours after being subjected to intensive security and non-security tests.

Become a Heimdal Partner

Conclusions

Conclusions

The patch management process is a multi-layered framework that incorporates security, business, and socio-professional-related aspects. All the methodologies, controls, and procedures do not exhaustively cover patch management, but they can become the baseline of a healthy and 'well-oiled' patch management flow. Always remember that automation is the key to dealing with vulnerabilities, whether they are human-centric or software design flaws.

YOUR JOURNEY TO IMMERSIVE PATCH MANAGEMENT STARTS HERE

Book a Demo

Heimdal®

Leading the fight against cybercrime.

www.heimdalsecurity.com