PatchMon PATCHMON BLOG

Linux Patch Compliance: What SOC 2 and ISO 27001 Actually Require

A practical breakdown of patching requirements for SOC 2, ISO 27001, PCI-DSS, and HIPAA - what auditors actually ask for, what evidence you need, and how to stop dreading audit season.

AUTHOR

PatchMon Team

PUBLISHED

20 March 2026

CATEGORIES

Security, Compliance

READ ONLINE

https://patchmon.net/blog/linux-patch-compliance-soc2-iso27001

Contents

There is a question that has derailed more audit prep meetings than any other. It usually arrives in an email from your auditor, phrased with the politeness of someone who knows exactly how much pain they are about to cause:

"Please provide evidence that all in-scope servers were patched within 30 days of CVE publication for the audit period."

If you have ever felt your stomach drop reading something like that, this article is for you. We are going to walk through what SOC 2, ISO 27001, PCI-DSS, and HIPAA actually require when it comes to Linux patch management: not what vendors tell you they require, not vague "best practices," but the specific controls, the specific evidence, and the specific gaps that trip up teams every audit cycle.

Why Patching Is the Compliance Control That Bites Hardest

Most compliance frameworks do not care which web framework you use or whether your CI/CD pipeline has a pretty dashboard. But every single one of them cares about patching. The reason is simple: unpatched vulnerabilities are the most common attack vector, and every framework author knows it.

The problem is not that patching is hard to understand. The problem is that proving you patched everything, on time, with proper approval, across every server in your fleet, for the entire audit period, is hard. Especially when "your fleet" includes three servers that someone spun up for a migration two years ago and forgot to decommission.

Let us break down what each framework actually says.

What Each Framework Actually Requires

SOC 2 Type II

SOC 2 does not hand you a checklist of specific technical controls. It is a principles-based framework built around the Trust Services Criteria (TSC). That means your auditor has discretion in how they evaluate you, which can be both a blessing and a curse. The relevant criteria for patching are:

CC6.1: Logical and Physical Access Controls

CC6.1 deals with how you identify, manage, and restrict access to information assets. For patching, the connection is your asset inventory. You cannot patch what you do not know about. Auditors use CC6.1 to verify that you have a complete, current inventory of all systems in scope.

What they will ask for:

- A list of all servers, VMs, and containers in the environment
- Evidence that this list is reviewed and updated regularly
- Proof that decommissioned systems are actually removed

CC7.1: Monitoring

CC7.1 requires that you detect and monitor for vulnerabilities and anomalies. In practice, this means vulnerability scanning. Auditors want to see that you are running regular scans and that you have a process for acting on the results.

What they will ask for:

- Vulnerability scan reports covering the audit period
- Evidence that scan results feed into a remediation workflow
- Scan frequency documentation (monthly for external-facing, quarterly minimum for internal)

CC7.2: Incident Response and Monitoring

CC7.2 covers how you evaluate and respond to identified events. When a critical CVE drops and you do not patch for 90 days, that is an event that should have triggered your response process.

CC8.1: Change Management

This is the big one for patching. In 2022, the AICPA added an explicit point of focus under CC8.1: *"Manages Patch Changes: A process is in place to identify, evaluate, test, approve and implement patches in a timely manner on infrastructure and software."*

What they will ask for:

- A documented patch management policy with defined cadences
- Evidence that patches go through change control (approval, testing, deployment)
- Logs showing patch deployment dates relative to CVE publication dates
- Exception documentation for anything not patched within your stated SLA

The SOC 2 summary: There is no mandated "30-day patching window" in SOC 2 itself. But you must define your own patching SLA in your policy, and then your auditor will hold you to it. If your policy says critical patches within 14 days and you have servers sitting unpatched for 60 days with no documented exception, you have a finding.

ISO 27001

ISO 27001 is a controls-based framework. If you are being audited against the 2022 revision (which you should be by now), the relevant control is:

Annex A 8.8: Management of Technical Vulnerabilities (formerly A.12.6.1 in the 2013 version)

The control text states: *"Information about technical vulnerabilities of information systems being used shall be obtained in a timely fashion, the organization's exposure to such vulnerabilities evaluated, and appropriate measures taken to address the associated risk."*

This breaks down into several concrete expectations:

Vulnerability identification: You must have a process for learning about new vulnerabilities relevant to your systems. Subscribing to vendor security mailing lists (e.g., RHEL errata, Ubuntu USNs, Debian DSAs), monitoring CVE databases, or using vulnerability scanning tools all count.

Risk assessment: Not every vulnerability needs an emergency patch. ISO 27001 expects you to evaluate each vulnerability against your risk appetite. A critical RCE in OpenSSH on an internet-facing server is very different from a low-severity information disclosure bug in a library used only in a sandboxed dev environment.

Timely remediation: ISO 27001 does not specify a number of days. Instead, your remediation timeline must align with your documented risk framework and SLAs. Your auditor will check whether you followed your own policy.

Verification: After patching, you need evidence that the patch was actually applied successfully. "We ran apt upgrade" is not enough. Show the auditor the before-and-after state.

Also relevant: Annex A 8.32 (formerly A.14.2.2), Change Management

Patches are changes. Your change management process must cover them. This means patches need approval records, rollback plans, and post-implementation verification, just like any other change to production systems.

The ISO 27001 summary: Define your patching SLAs based on risk severity, document your vulnerability management process, follow it consistently, and keep evidence of every step. The auditor will look at your ISMS documentation and then verify that reality matches the documentation.

PCI-DSS 4.0.1

PCI-DSS is the most prescriptive of the bunch when it comes to patching timelines. If you process, store, or transmit cardholder data, here is what you need to know:

Requirement 6.3.3: Patch Installation

Under PCI-DSS 4.0.1 (which replaced 4.0 effective December 31, 2024):

> *Critical security patches must be installed within one month of release.*

Note the change from PCI-DSS 4.0, which required both critical *and* high-severity patches within 30 days. Version 4.0.1 scaled this back to critical only. For non-critical patches, you must still install them, but the timeline is determined by your risk-based assessment.

What auditors will ask for:

- A list of all critical CVEs affecting in-scope systems during the audit period
- Patch deployment dates for each, with evidence
- For anything over 30 days: documented risk acceptance, compensating controls, and a remediation plan
- Evidence of a defined process for identifying applicable patches (you cannot claim ignorance)

Requirement 6.3.1: Vulnerability Identification

You must have a process for identifying security vulnerabilities using reputable sources (NVD, vendor advisories, security research groups) and assigning risk rankings.

The PCI-DSS summary: 30 days for critical patches. No wiggle room unless you have documented compensating controls. This is the framework where "we meant to get to it" will earn you a finding the fastest.

HIPAA

HIPAA is deliberately vague about technical implementation, which ironically makes it harder to comply with because you cannot point to a specific technical requirement and say "done."

The relevant sections are:

45 CFR 164.308(a)(1): Security Management Process

Requires covered entities to "implement policies and procedures to prevent, detect, contain, and correct security violations." Patch management falls squarely under "prevent and correct."

45 CFR 164.308(a)(1)(ii)(A): Risk Analysis

You must conduct an accurate and thorough assessment of risks and vulnerabilities to ePHI. Unpatched systems are vulnerabilities. If your risk analysis does not account for patch status, it is incomplete.

45 CFR 164.308(a)(1)(ii)(B): Risk Management

You must implement security measures sufficient to reduce risks to a reasonable and appropriate level. If your risk analysis identifies unpatched servers and you do nothing about them, you have failed risk management.

45 CFR 164.308(a)(5)(ii)(B): Protection from Malicious Software

This addressable specification covers procedures for protecting against, detecting, and reporting malicious software. Patching is a primary defense against malware that exploits known vulnerabilities.

The HIPAA summary: There is no specific patch timeline. But the OCR (Office for Civil Rights) has issued guidance making it clear that a lack of patch management is considered a HIPAA violation. In enforcement actions, failure to patch known vulnerabilities has been cited repeatedly as evidence of inadequate risk management.

Quick Reference: Framework Comparison

Framework	Patching Timeline	Specificity	Key Control	Evidence Type
SOC 2 Type II	Your defined SLA	Principles-based	CC8.1	Policy + logs + exceptions
ISO 27001:2022	Risk-based per your SLAs	Controls-based	A.8.8 (was A.12.6.1)	ISMS docs + scan reports + change records
PCI-DSS 4.0.1	30 days (critical)	Prescriptive	6.3.3	Patch dates vs. CVE dates
HIPAA	"Reasonable"	Vague	164.308(a)(1)	Risk analysis + remediation evidence

What Auditors Actually Look For

Forget what the framework text says for a moment. Here is what happens in a real audit, based on patterns that repeat across SOC 2, ISO 27001, PCI-DSS, and HIPAA assessments.

1. Complete Asset Inventory

The auditor's first move is to ask for your server inventory. Then they will compare it against what they can see: your cloud provider's asset list, your monitoring tool, your DNS records, your load balancer configs. If there is a server in AWS that is not on your inventory, you have a problem before the patching discussion even starts.

What "good" looks like:

- Automated asset discovery (not a manually maintained spreadsheet)
- Every server has an owner, a purpose, and a classification
- Decommissioned servers are flagged with decommission dates
- The inventory includes OS version and patch level

2. Patch Status Per System With Timestamps

For every server in your inventory, the auditor wants to see:

- What patches are installed
- When each patch was installed
- What patches are available but not installed
- How long available patches have been pending

This is where most teams struggle. If your patching process is "SSH in, run apt upgrade, move on," you probably do not have timestamps for when individual patches were applied. Package manager logs help, but they are local to each server and nobody is aggregating them.

3. Policy Documentation

Your patch management policy needs to cover:

- **Patching cadence**: How often you check for and apply patches (e.g., weekly review, monthly application cycle, emergency patches within 72 hours)
- **Severity classification**: How you categorize patches (critical, high, medium, low) and what your SLA is for each
- **Approval workflow**: Who approves patches, especially for production systems
- **Testing requirements**: Whether patches are tested before production deployment, and how
- **Exception process**: What happens when a patch cannot be applied (risk acceptance, compensating controls, timeline for resolution)
- **Roles and responsibilities**: Who is responsible for identifying, testing, approving, and deploying patches

4. Evidence of Patch Application

Acceptable evidence varies by auditor, but generally includes:

- Configuration management tool logs (Ansible playbook runs, Puppet/Chef reports)
- Package manager logs with timestamps (`/var/log/dpkg.log`, `/var/log/yum.log`)
- Vulnerability scan results showing before-and-after states
- Change management tickets showing the patch was approved and deployed
- Screenshots or reports from patch management tools

The key is that evidence must be timestamped and attributable. "We patched it" is not evidence. "Server web-prod-03 had package openssl 3.0.2-0ubuntu1.12 installed on 2026-01-15 at 14:32 UTC via Ansible playbook run #4521, approved by J. Smith in ticket CHG-1847". That is evidence.

5. Exception Documentation

Every auditor understands that not every patch can be applied immediately. Legacy applications break. Kernel updates require downtime windows. Some patches introduce regressions. That is fine. What is not fine is having no documentation explaining why.

A proper exception record includes:

- The specific CVE or patch being deferred
- The affected systems
- The business reason for the deferral
- The risk assessment (what is the exposure?)
- Compensating controls in place (network segmentation, WAF rules, enhanced monitoring)
- The planned remediation date
- Approval from the risk owner (not just the sysadmin)

Building Your Evidence Trail

If you are reading this before your audit (good planning), here is how to build an evidence trail that will survive scrutiny.

Start With Your Policy

Write the policy first. It does not need to be 40 pages. It needs to be specific enough that someone could follow it without asking you questions. A strong patch management policy fits on 3-5 pages and covers:

- Scope (which systems are covered)
- Roles (who does what)
- Classification (how you score severity)
- SLAs (time-to-patch for each severity level)
- Process (identify, evaluate, test, approve, deploy, verify)
- Exceptions (how to request, who approves, what documentation is required)

Suggested SLA structure that works across most frameworks:

Severity	Definition	Patch SLA	Scan Verification
Critical	CVSS 9.0+, active exploitation, or RCE on internet-facing systems	72 hours	Next scheduled scan
High	CVSS 7.0-8.9, or privilege escalation	14 days	Next scheduled scan
Medium	CVSS 4.0-6.9	30 days	Next monthly scan
Low	CVSS below 4.0	90 days	Next quarterly scan

These timelines will satisfy PCI-DSS 4.0.1's 30-day critical patch requirement and give you defensible SLAs for SOC 2 and ISO 27001 auditors. Adjust based on your risk appetite, but be prepared to justify whatever timelines you choose.

Automate the Evidence Collection

Manual evidence collection does not scale and it does not hold up well under audit scrutiny. "I ran the command and copied the output into a document" raises questions about data integrity that automated logging does not.

Your evidence automation should cover:

Asset inventory: Pull from your cloud provider APIs, configuration management database, or monitoring system. Reconcile monthly at minimum. Flag any drift between expected and actual assets.

Patch status: Collect installed package lists and available updates from every server on a regular schedule. Store these centrally with timestamps. Diff them over time to show when patches were applied.

Change records: Link patch deployments to change management tickets. If you use Ansible, your playbook runs are your evidence; make sure you are logging them to a central location, not just stdout.

Vulnerability scans: Run vulnerability scans on a regular cadence (weekly or monthly depending on framework requirements) and store the reports. These serve as independent verification that patches were actually applied.

Create a Patching Runbook

Auditors love runbooks because they demonstrate that your process is repeatable and not dependent on one person's knowledge. Your patching runbook should cover:

- How to check for available updates on each OS in your fleet

- How to evaluate whether a patch is relevant to your environment
- How to test a patch before production deployment
- How to deploy a patch (including rollback steps)
- How to verify the patch was applied successfully
- How to document the patch in your change management system
- How to handle a patch that breaks something

Common Gaps That Catch Teams Off Guard

These are the issues that come up in audit after audit. If you address them proactively, you will save yourself a lot of stress.

The Forgotten Server Problem

Every organization has them. The staging server that became semi-production. The monitoring box that nobody touches because "it works." The server that was supposed to be temporary three years ago. These machines are almost never in your patch cycle, and they are exactly what auditors look for.

Fix: Automated asset discovery that runs continuously. If a machine exists and is powered on, it should be in your inventory. Tag it, assign an owner, patch it or decommission it.

Manual Processes With No Logs

If your patching process is a human SSHing into servers and running commands, you probably have gaps in your evidence trail. Even if you patch everything on time, you cannot prove it without logs.

Fix: Use a configuration management tool (Ansible, Puppet, Chef, Salt) or a dedicated patch management solution. The tool itself becomes your audit trail. If you must patch manually, at minimum use `script` to capture terminal sessions and store them.

No Exception Documentation

This is possibly the most common finding. A server was not patched because the application team said it would break their app. Everyone verbally agreed this was acceptable. Nobody wrote it down. The auditor finds the unpatched server and there is no documentation explaining why.

Fix: Make exception requests a formal process. A simple template in your ticketing system works. The key is that someone with authority approved the risk, compensating controls are documented, and there is a remediation date.

Inconsistent Scan Coverage

You run vulnerability scans, but they do not cover every server. Maybe the scanner cannot reach certain network segments. Maybe a firewall rule blocks the scan. The auditor pulls your asset inventory, pulls your scan coverage, and finds 15% of servers have never been scanned.

Fix: Validate scan coverage against your asset inventory every month. If a server cannot be scanned remotely, use an agent-based scanner or audit it locally.

Kernel Patches and Reboot Avoidance

Linux kernel patches require a reboot to take effect (unless you are using kernel live patching). Many teams apply the package update but never reboot the server. The package manager shows the patch as installed, but the running kernel is still vulnerable.

Fix: Check running kernel version against installed kernel version. Monitor uptime relative to patch dates. If a server has been up for 200 days and you "patched" a kernel vulnerability 60 days ago, something does not add up.

Container and Ephemeral Workloads

Your compliance scope probably includes containers and ephemeral instances. "But they get replaced every deployment" is not a sufficient answer if the base images they are built from contain unpatched vulnerabilities.

Fix: Scan container images in your CI/CD pipeline and your registry. Track base image versions and their patch status. Include image rebuild cadence in your patch management policy.

Tools That Help

There is no single tool that solves compliance patching. Most teams use a combination:

For patch deployment:

- Ansible, Puppet, Chef, or Salt for automated patch deployment across Linux fleets. Ansible in particular works well because playbook runs generate logs that serve as audit evidence.
- OS-native tools: `unattended-upgrades` (Debian/Ubuntu), `dnf-automatic` (RHEL/Fedora) for automated security updates on less critical systems.

For vulnerability scanning:

- OpenVAS/Greenbone for network-based vulnerability scanning.
- OpenSCAP for compliance scanning against CIS benchmarks and STIG profiles.
- Nessus or Qualys for commercial scanning with compliance-specific reporting.

For inventory and visibility:

Your cloud provider's asset inventory tools (AWS Systems Manager, Azure Arc, GCP OS Inventory).

Configuration management databases that pull from multiple sources.

For compliance reporting and evidence:

Obligatory disclosure: this is the PatchMon blog, so you can guess what we are about to recommend. That said, PatchMon provides compliance scanning using OpenSCAP, CIS benchmarks, and Docker Bench on the Max tier, with automated reporting that maps patch status to framework requirements. If you are managing a fleet of Linux servers and need to produce audit-ready evidence without stitching together data from five different tools, it is worth evaluating. The platform maintains a complete history of patch status across your fleet with timestamps, which directly addresses the "show me evidence" requests that auditors lead with. The quickest way to evaluate it is a PatchMon Cloud trial (https://patchmon.net/pricing), which is billed per-host, runs for 14 days (card required, cancel before day 14 and you are not charged), and is managed for you. If you'd rather self-host, the AGPLv3 Community Edition is free and deployable in minutes on your own stack.

For change management:

Your existing ticketing system (Jira, ServiceNow, etc.) is usually sufficient. The key is linking patch deployments to tickets consistently.

The important thing is not which specific tools you use. It is that whatever you use generates timestamped, centralized, tamper-evident records. An auditor does not care whether you use Ansible or Puppet. They care that you can prove what happened, when, and who approved it.

Building Your Pre-Audit Checklist

Here is a practical checklist you can work through before your next audit. It covers the evidence requirements across SOC 2, ISO 27001, PCI-DSS, and HIPAA:

Documentation (prepare once, update annually):

- [ ] Patch management policy with defined SLAs per severity
- [ ] Roles and responsibilities matrix for vulnerability management
- [ ] Exception/risk acceptance process documentation
- [ ] Patching runbook for each OS in your environment
- [ ] Change management procedure that explicitly covers patches

Ongoing evidence (collect continuously):

- [ ] Complete asset inventory, reconciled monthly
- [ ] Vulnerability scan reports for every scan in the audit period
- [ ] Patch deployment logs with timestamps for every system

- [ ] Change management tickets for patch deployments
- [ ] Exception/risk acceptance records for deferred patches

Pre-audit verification:

- [ ] Confirm asset inventory matches reality (run a discovery scan)
- [ ] Verify scan coverage: every in-scope server has been scanned
- [ ] Spot-check patch evidence for 10-20 servers across the audit period
- [ ] Review all open exceptions: are compensating controls still in place?
- [ ] Confirm no servers have pending critical patches older than your SLA
- [ ] Check running kernel versions match installed kernel versions
- [ ] Verify container base images are current

A Note on "Continuous Compliance"

You will hear this phrase a lot from vendors. The underlying idea is sound: if you treat compliance as something you maintain continuously rather than something you scramble to prove once a year, audits become less painful.

In practice, this means:

Automate evidence collection so it happens without human intervention

Run vulnerability scans on a schedule, not just before audits

Review and patch on a regular cadence, not in a pre-audit panic

Address exceptions as they arise, not when the auditor finds them

Keep your policy current; if your actual process has changed, update the policy to match

This is what PatchMon was designed around. Continuous visibility into patch status across your fleet, compliance scanning with OpenSCAP, CIS, and Docker Bench (Max tier), automated reports with executive summaries, and a complete audit trail with timestamps for every patch run: who triggered it, who approved it, what changed, and what the output was. When the auditor asks for evidence, you pull up a report. You do not spend two days assembling it from scattered logs.

Final Thoughts

Compliance patching is not exciting work. Nobody got into systems administration because they dreamed of producing evidence packages for SOC 2 auditors. But it is work that matters, both for the security of your systems and for the continued operation of your organization.

The good news is that the frameworks, despite their differences, all want the same basic things: know what you have, keep it patched, document your process, follow your process, and have a plan for when you cannot follow your process. If you build a system that does those five things with timestamped evidence, you will pass your audit.

The teams that struggle are not the ones with complex environments or unusual architectures. They are the ones with no documented process, no centralized evidence, and no exception handling. Fix those three things and audit season becomes a minor inconvenience rather than a crisis.

Start with the checklist above. Fill in the gaps. And if you want a tool that handles the evidence collection, compliance scanning, and reporting automatically, PatchMon Cloud (https://patchmon.net/pricing) is the fastest path in. You get a 14-day trial on real hosts (card required, cancel before day 14 and you are not charged), billing tracks the hosts that actually check in rather than a fleet tier, and the OpenSCAP, CIS, and Docker Bench scans you need for SOC 2 and ISO 27001 evidence sit on the Max tier. If you'd rather self-host, the AGPLv3 Community Edition (https://patchmon.net/open-source) is free. Whichever route you take, deploy it before your next audit, not the night before.

PatchMon

ABOUT PATCHMON

The open source Linux patch management platform

PatchMon gives sysadmins one dashboard for patching across Linux, FreeBSD, and Windows fleets. Run it as a managed SaaS on PatchMon Cloud (per-host billing, 14-day trial, no fleet minimum) or self-host the AGPLv3 Community Edition on your own infrastructure.

Start a trial: patchmon.net/pricing