
IQDESK ENTERPRISE · NURIQ TECHNOLOGIES

Patch Management

A complete guide to detecting, prioritizing, and safely remediating vulnerabilities

PREPARED FOR
IQDesk Customers

DATE
August 2026

DOCUMENT
PATCHIQ-UM-1.0



For Customer Use
© 2026 NurIQ Technologies LLP

Contents

1. What Patch Management Does For You	
2. How Detection Works	
3. The Dashboard	
4. Findings — By Product and All Findings	
5. The Product Detail Page	
6. The Finding Detail Page	
7. Policies & Approval	
8. Reboots — Only When Actually Required	
9. Deploying a Patch	
10. Deployment Safety: Backup & Rollback	
11. Automated Playbooks	
12. Staged Rollouts	
13. Maintenance Windows	
14. Exceptions	
15. Software Catalog, Supported Software Catalog & Baselines	
16. Data Sources (Administrators)	
17. Roles & Permissions	
18. FAQ & Troubleshooting	

1. What Patch Management Does For You

Patch Management (module name **PatchIQ**) is IQDesk's vulnerability detection and patch-remediation workspace. Every device running the IQDesk agent reports what software is installed on it; Patch Management correlates that inventory against real, publicly published vulnerability data and shows you exactly which endpoints need which fix — with a recommended remediation, an approval workflow, and a deployment engine that installs it, backs up first where that's possible, and rolls back automatically if the install fails.

This manual covers everything a Manager or Administrator sees and does inside Patch Management, in the order you'll actually use it: understanding what was found, deciding what to fix, and getting the fix installed safely.

Patch Management is visible to **Manager** and above. Viewing findings, deployments, and reports is a Manager-level action throughout this module — but as you'll see in [Roles & Permissions](#), most of the buttons that actually change something (approve a remediation, start a deployment, run a playbook, create a policy or exception) require **Admin**.

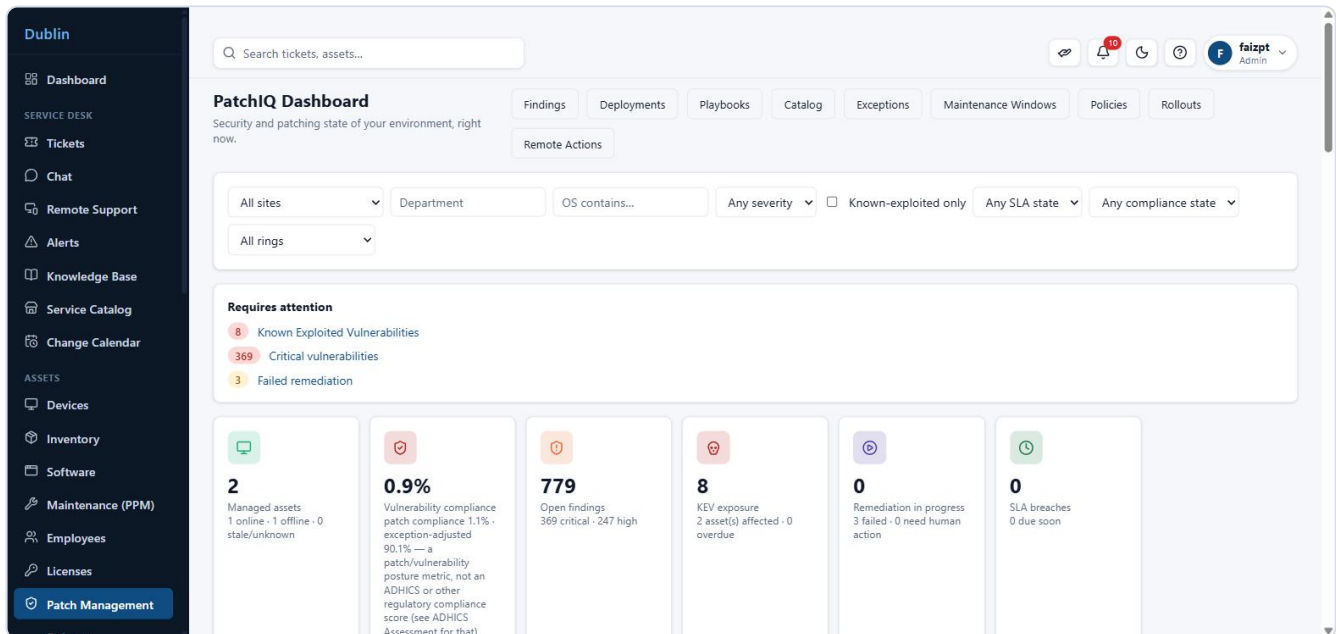
2. How Detection Works

Detection is a pipeline, not a single scan, and every step is real, running continuously in the background:

1. **Inventory collection** — the IQDesk agent on each endpoint reports its installed software (and, on Windows, which security updates are missing) at check-in. This is the same inventory you see under Devices → Software.
2. **Vulnerability feed sync** — IQDesk pulls known-vulnerability data from the same public sources security teams already use: the National Vulnerability Database (NVD) for CVE records and affected-version ranges, CISA's Known Exploited Vulnerabilities (KEV) catalog, FIRST.org's EPSS exploit-probability scores, Microsoft's MSRC advisories, the Open Source Vulnerabilities (OSV) database, and native package-security feeds for Ubuntu, Debian, and RHEL/Rocky. Nothing here is a private or invented dataset — it's the same public record any vendor's own security bulletin would point to.
3. **Correlation** — a background pass runs continuously (roughly every 30 minutes for correlation, matching how often endpoints check in), matching your installed software's exact version, or Windows update ID, against known affected-version ranges. A match becomes an open **Finding**.
4. **Prioritization** — every finding carries the vulnerability's official severity (Critical/ High/Medium/Low with its CVSS score), whether it's on CISA's Known Exploited list, and where available its EPSS score (a real-world probability that this specific vulnerability will actually be exploited in the next 30 days) — so you can tell "technically severe" apart from "actively being used against real targets right now."

This is honest, evidence-based detection, not a promise of omniscience. It surfaces what's already published in these public advisory databases — it cannot warn you about a vulnerability before it's publicly disclosed (a "zero-day"), and it can only see software your operating system's own inventory mechanism exposes (a small number of installers hide their real product behind a generic "launcher" entry — this is a limitation of the installer itself, not something any patch tool can see around).

3. The Dashboard



The Patch Management Dashboard — every number here is computed live from your actual inventory and findings.

The Dashboard is your daily landing page. A **Requires attention** panel leads with the three things that matter most: how many Known Exploited Vulnerabilities you're carrying, how many Critical vulnerabilities, and how many remediations have actually failed. Below it, stat tiles cover managed assets (online/offline/stale), a vulnerability compliance percentage, open findings by severity, KEV exposure, remediation-in-progress counts, and SLA breaches.

Filter the whole dashboard by site, department, OS, severity, KEV status, SLA state, compliance state, or deployment ring — every tile and list narrows to match.

4. Findings — By Product and All Findings

Vulnerabilities
Findings correlated against your asset inventory, with the recommended fix for each.

By product | All findings | Open | Any severity | ☐ Known-exploited only

PRODUCT	VULNERABILITIES	AFFECTED ASSETS	CRITICAL	HIGH	MEDIUM / LOW	LAST SEEN
Adobe Acrobat XI Pro KEY	683	1	369	213	111 / 8	Aug 26, 2026, 12:13 AM
Notepad++ (64-bit x64) KEY	13	1	0	5	8 / 0	Aug 26, 2026, 12:14 AM
7-Zip 19.00 (x64) KEY	17	1	0	8	6 / 3	Aug 26, 2026, 12:14 AM
Wireshark 3.4.0 64-bit	37	1	0	19	12 / 6	Aug 26, 2026, 12:14 AM
PuTTY release 0.74 (64-bit)	6	1	0	2	2 / 2	Aug 26, 2026, 12:14 AM
Java(TM) 6 Update 45	1	1	0	0	1 / 0	Aug 26, 2026, 12:13 AM
PuTTY release 0.83 (64-bit)	4	1	0	0	0 / 4	Aug 26, 2026, 12:13 AM

© 2026 NuriQ Technologies

Findings, By Product view — one row per product, not one row per CVE.

Click **Findings** from the Dashboard to see every open vulnerability. A single old, long-unpatched application (a legacy PDF reader, for example) can genuinely carry **hundreds** of individually-published CVEs — that's real published data, not an error — but reading hundreds of near-identical rows for one product on one machine tells you nothing useful. That's exactly what the **By Product** view (the default) is for:

- One row per product, showing how many distinct vulnerabilities it carries, how many of your endpoints are affected, a severity breakdown (Critical/High/Medium-Low), whether any of them are Known Exploited, and when it was last seen.
- Click a product to open its [detail page](#) — the full CVE list and the full affected-endpoint list for that one product.

The **All Findings** toggle switches back to the original flat list — one row per individual (endpoint, vulnerability) pair — useful when you want to work findings one at a time rather than by product.

Both views share the same filters: **Status** (Open / Resolved / All), **minimum severity**, and **Known-exploited only**. A finding that's no longer detected on a re-scan (because the software was upgraded, removed, or an exception now covers it) automatically moves to Resolved — nobody has to close it by hand.

Just fixed something and don't want to wait for the endpoint's normal check-in? Open the endpoint's own Findings row (in All Findings view) and click **Refresh** — it asks that one device to report its software list again on its next check-in, usually within about a minute.

5. The Product Detail Page

Dublin

Search tickets, assets...

← Vulnerabilities

7-Zip 19.00 (x64)

17 vulnerabilities across 1 affected asset

Vulnerabilities

CVE	SEVERITY	AFFECTED ASSETS
CVE-2025-0411 KEV 7-Zip Mark-of-the-Web Bypass Vulnerability. This vulnerability allows remote attackers to bypass the Mark-of-the-Web protection mechanism on affected installations of 7-Zip. User interaction is requir	High 7.0	1 endpoint
CVE-2026-14266 7-Zip XZ Decompression Heap-based Buffer Overflow Remote Code Execution Vulnerability. This vulnerability allows remote attackers to execute arbitrary code on affected installations of 7-Zip. User int	High 7.8	1 endpoint
CVE-2024-11477 7-Zip Zstandard Decompression Integer Underflow Remote Code Execution Vulnerability. This vulnerability allows remote attackers to execute arbitrary code on affected installations of 7-Zip. Interactio	High 7.8	1 endpoint
CVE-2023-31102 Ppmd7.c in 7-Zip before 23.00 allows an integer underflow and invalid read operation via a crafted 7Z archive.	High 7.8	1 endpoint
CVE-2025-53817 7-Zip is a file archiver with a high compression ratio. 7-Zip supports extracting from Compound Documents. Prior to version 25.0.0, a null pointer dereference in the CompoundDocument class lead to denia	High 7.5	1 endpoint

7-Zip's product detail page — every CVE, its own affected-endpoint count, and the recommended fix.

Opening a product from the By Product view shows two things side by side:

- **Vulnerabilities** — every CVE affecting this product across your whole environment, with its own severity, CVSS score, KEV flag, how many of your endpoints it specifically affects, and the recommended fix (usually "upgrade to version X"). Click the endpoint count to jump straight to [deploying that specific fix](#) to every affected endpoint at once.
- **Affected assets** — every endpoint carrying this product, its installed version, how many open findings it has for this product, its worst severity, and when it was last seen.

This is the page to use when you've decided "we need to deal with this product" rather than one finding at a time.

6. The Finding Detail Page

Opening an individual finding (from All Findings, or by clicking through from a device) shows the full story for that one (endpoint, vulnerability) pair: the installed version versus the affected version range, the CVE's severity/CVSS/KEV status, why it was flagged (the exact detection reason — a specific missing Windows update ID, or a version falling inside a published affected range), when it was first and last seen, and its current status.

Below that, **Remediation options** lists the available fix (or fixes) for this finding, each with a live policy-evaluation badge (see [Policies & Approval](#), next) and, once eligible, the controls to actually deploy it.

7. Policies & Approval

Nothing deploys itself just because a fix exists. Every remediation is evaluated against your organization's **Policies** (Administration → Patch Management → Policies) before it can be approved or deployed. A policy matches on OS, product name, minimum severity, minimum CVSS score, KEV status, remediation type, and optionally a specific list of target or excluded assets — and, once matched, decides whether the fix needs a human sign-off (**requires approval**) or can be **auto-approved**, plus its reboot behavior, whether to back up first, and an SLA (how many days you have to remediate it).

Fail-closed by design. If no enabled policy matches a remediation at all, it does **not** silently become deployable — it falls back to **Approval Required** until either a policy is created to cover it, or someone approves it directly. Nothing in Patch Management can accidentally auto-deploy simply because no one configured a rule for it.

When you open a finding, each remediation option shows its live evaluation result:

- **Eligible** — ready to deploy or schedule right now.
- **Approval Required** — needs a sign-off first; an eligible approver sees an **Approve remediation** button.
- **Excluded** — an active [Exception](#) covers this exact vulnerability/ asset combination.
- **Blocked** — a prior approval was explicitly rejected.
- **Not Eligible** — no safe automated fix exists yet (for example, the vendor hasn't published a package IQDesk can install automatically).

Auto-approval is real and fully audited: when a policy is configured to auto-approve, IQDesk still creates a genuine approval record (marked "Auto-approved by policy") the moment it applies, rather than simply skipping the step — so the audit trail always shows exactly how and when something became eligible to deploy.

8. Reboots — Only When Actually Required

THE SHORT VERSION

IQDesk only reboots an endpoint after a patch when that specific install genuinely needed one — never as a blanket habit, and never because of an unrelated, leftover reboot flag from something else that happened on that machine days earlier.

An unnecessary reboot is one of the most disruptive things a patch tool can do to a user's working day, so IQDesk checks for a real need in two independent ways before it ever reboots anything:

1. **The installer's own signal.** A standard Windows Installer (MSI) package can explicitly report "exit code 3010" — the industry-standard way an installer says "I succeeded, but a reboot is needed to finish." IQDesk always honors this.
2. **A genuinely new pending-reboot condition.** IQDesk also checks Windows' own low-level reboot-pending signals immediately before and immediately after the install runs, and only counts it if that specific install is what caused the change from "no reboot pending" to "reboot now pending." A reboot flag that was already sitting there before this install started — left over from something unrelated — is never attributed to the current job, so it can never trigger a reboot the current patch didn't actually need.

On top of that signal, your own [Policy](#) settings control what actually happens next:

- **Reboot if required** (the default) — reboots only when the checks above say it's genuinely needed.
- **No reboot** — never reboots automatically; the endpoint stays in a pending state until someone reboots it by hand.
- **Force reboot** — always reboots after this policy's deployments, regardless of signal (useful for a maintenance-window batch where you'd rather reboot proactively than risk a half-applied update lingering).

You can also require a reboot to be deferrable: **Allow defer** plus a maximum number of deferrals lets a user postpone an unavoidable reboot a limited number of times before it's forced, and **notify before reboot** gives them an on-screen warning first.

A normal application update — 7-Zip, Notepad++, PuTTY, and the vast majority of everyday software — essentially never needs a reboot at all, and will not trigger one under this behavior. Reboots remain genuinely common only for the category of fix that has always needed them: an operating-system-level Windows security update.

9. Deploying a Patch

There are two ways to actually push a fix out, matched to two different situations:

One remediation, from a finding

From a finding's detail page, once a remediation shows **Eligible**, **Schedule deployment** lets you pick an exact date/time (or leave it blank to deploy as soon as it's eligible / at the next open maintenance window). **Run playbook** instead routes it through an [Automated Playbook](#) if one exists for a more careful, multi-step rollout.

One vulnerability, many endpoints at once

From a product's [detail page](#), clicking a CVE's affected-endpoint count opens the multi-endpoint deploy picker: every endpoint affected by that specific vulnerability, pre-selected if it already has a ready recommended fix, with checkboxes to include or exclude individual machines. Confirming deploys immediately to everything selected — each endpoint is tracked completely independently, so one machine's failure never affects or hides the result on any other.

Both single- and multi-endpoint deployment actions require an **Administrator** — a Manager can view every finding and every deployment's progress, but starting the actual install is an Admin-level action, reflecting the real change being made to a production endpoint.

Every deployment then runs through the same real lifecycle regardless of which path started it: queued → downloading the package → (optionally) backing up → installing → verifying → completed (or, on failure, retried or automatically rolled back — see next section). A deployment's own page has a **Refresh status now** button so you don't have to wait for the endpoint's normal check-in schedule to see whether it worked.

10. Deployment Safety: Backup & Rollback

When a policy has **Backup before install** enabled, IQDesk takes a real backup of the existing installation on the endpoint itself before touching anything — the install is never even attempted if that backup fails to complete and verify. (This applies to installer-based packages; a Windows security update delivered as a native OS update doesn't go through this same file-level backup step, since Windows manages its own update rollback for those.)

If an install then fails — or fails its own post-install verification — and a good backup exists, IQDesk **automatically rolls it back** to the pre-patch state, with no one needing to notice and click anything. You'll also see a manual **Rollback** option on an already-completed deployment, for the rarer case where the software installed fine but you've since decided you need to revert it anyway.

A batch deployment (many endpoints, one action) reports its own honest aggregate status — Successful, Failed, Partially Successful, or Completed with Rollbacks — so a handful of failures on a 50-machine rollout can never quietly read as a plain, uncomplicated "success."

11. Automated Playbooks

A playbook is a multi-step remediation for cases where "just run the installer" isn't quite enough — for example: gracefully close the running application first, wait for it to actually exit, install the fix, then re-verify the new version is really in place. Each step is tracked individually. A step that genuinely needs a human (confirming an application actually closed, rather than assuming it did) pauses the whole playbook and waits — an Administrator confirms it from the playbook execution's own page to let it continue.

12. Staged Rollouts

Rather than deploying a fix to every endpoint at once, a **Rollout** stages it across ordered waves — for example, a small Test group, then Pilot, then Early Production, then full Production, and finally Critical Infrastructure. Each wave only starts once the previous one's own deployment jobs have finished, and a wave is only marked successful if it clears a real health check (a minimum sample size and a failure-rate threshold) — if a wave's failure rate is too high, the whole rollout pauses right there instead of ploughing ahead into the next, larger wave.

A wave targeting **Critical Infrastructure** always requires explicit approval before it can proceed, no matter how a rollout's other waves were configured — this is a hard rule, not a default that can be quietly turned off.

13. Maintenance Windows

A Maintenance Window defines when patching is allowed to actually happen for a given deployment — a start/end time, a time zone, which days it recurs on, and optional blackout/change-freeze date ranges. A deployment scheduled against a window is genuinely re-checked at the moment it's about to run, not just when it was first scheduled: if the window has since closed, the job is **deferred to the next open occurrence**, not silently pushed out anyway. It only ever runs inside a window you actually configured.

14. Exceptions

Sometimes a vulnerability genuinely can't be remediated yet — a business-critical application that can't be upgraded until its own vendor certifies compatibility, for example. An **Exception** records that decision formally: what it covers (a specific vulnerability, a specific asset, a specific product, or a combination), a mandatory written reason, and a mandatory expiry date — a permanent, open-ended exception is not allowed by design.

An active exception doesn't hide the underlying finding from your inventory — it stays visible — but it does mark that finding's remediation as **Excluded** from the normal approval/deploy workflow, and it's counted favorably in your exception-adjusted compliance percentage, so a documented, time-boxed waiver doesn't quietly drag your compliance score down the same way an ignored vulnerability would.

Expiry is **enforced automatically and audited** the moment it passes — there is no separate reminder email or notification, so track upcoming expiries the same way you track anything else on a deadline: check the Exceptions list itself periodically, particularly before it lapses and that finding's normal, non-excepted workflow resumes.

15. Software Catalog, Supported Software Catalog & Baselines

Three related, but distinct, pages worth telling apart:

Software Catalog

Your organization's own library of installer packages that back a remediation fix — products, versions, architectures, silent-install commands, and a real per-package approval workflow (uploaded → checksum + signature verified → malware-scanned → Approved) before anything in it can ever be deployed. Nothing in this catalog installs or executes anything by itself; it's the repository a remediation's package is pulled from.

Supported Software Catalog

A separate, platform-level list of "IQDesk knows how to deploy this safely" software — this is what a Software Baseline (below) actually picks its required items from.

Software Baselines

A Baseline is a named list of "every matching endpoint should have these" — one or more required products (from the Supported Software Catalog above), each with an optional minimum version. Compliance against a baseline is computed live every time you view it, never a stale cached number. A baseline can also be marked as the **default for new enrollments** — when it is, a brand-new endpoint automatically gets the baseline's required software installed the moment it enrolls, with no one needing to remember to push it by hand.

16. Data Sources (Administrators)

Administration → Patch Management → Data Sources lists every vulnerability feed described in [How Detection Works](#), each with its own last-sync time and item count, and a manual **Sync now** to force an immediate refresh of one feed rather than waiting for its normal schedule. This is also where any feed that needs an API key (some threat-intel-style feeds elsewhere in IQDesk work the same way) gets configured — most core feeds used by Patch Management work out of the box with no key needed.

17. Roles & Permissions

ACTION	MINIMUM ROLE
View Dashboard, Findings, Deployments, Playbooks, Policies, Rollouts, Exceptions, Catalog, Baselines	Manager
Approve a remediation	Admin
Schedule or start a deployment (single or multi-endpoint)	Admin
Start, pause, resume, or cancel a playbook	Admin
Create/edit a Policy, Maintenance Window, Exception, or Rollout	Admin
Approve/reject a Software Catalog package	Admin
Cancel, retry, or roll back a deployment job	Admin

In short: a Manager has full visibility across this entire module and can make the day-to-day decision of what needs fixing, but every action that actually changes a production endpoint's state requires an Administrator — by design, matching the real change being made.

18. FAQ & Troubleshooting

Why does one product show hundreds of vulnerabilities?

A long-unpatched, older piece of software genuinely accumulates that many published CVEs over its lifetime — it's real public data, not a bug in detection. Use the [By Product](#) view to see it as one manageable row instead of hundreds.

A finding disappeared — was it ignored?

No: a finding automatically resolves the moment a re-scan no longer detects it — because the software was upgraded, removed, or because an active Exception now excludes it. Nothing is ever silently dropped without one of those three reasons.

Why can't I click Approve/Deploy even though I'm a Manager?

Those specific actions require Admin — see [Roles & Permissions](#). You can still see the button and everything else on the page; only the action itself is gated.

Will this reboot my users' machines unexpectedly?

Only when the specific patch that just ran genuinely needed one — see [Reboots — Only When Actually Required](#) — and only ever inside whatever your Policy's reboot setting and Maintenance Window actually allow.

A deployment failed — is my endpoint left in a broken state?

If backup-before-install was enabled for that policy, no — IQDesk automatically restores the pre-patch state. See [Deployment Safety](#).