Essential Eight on Linux, Part 2 of 8: Patch Applications on Ubuntu 26.04 LTS

April 27, 2026 / Gavin Jackson

essential-eight asd ism ubuntu ubuntu-pro landscape linux patching security

When people talk about patching on Linux, they often default to "just run `apt update && apt upgrade`."

That is not enough for the Essential Eight.

The ASD expectation is broader: patch internet-facing applications fast, patch office productivity tools, browsers, email clients, PDF readers, and security products, and prove you have a process to identify what is vulnerable in the first place.

On Ubuntu 26.04 LTS, this mitigation is one of the cleaner ones to implement, provided you standardise how software is delivered.

What ASD is trying to achieve

The risk here is not the kernel. It is the software users touch every day:

- web browsers
- email clients
- PDF readers
- collaboration tools
- locally installed runtimes
- public-facing application stacks

Attackers like application vulnerabilities because they are reachable and common. The more ad hoc your Linux software estate is, the harder this mitigation becomes.

Ubuntu 26.04 LTS reference implementation

Resolute Raccoon highlights

Resolute Raccoon improves the patching story in a few practical ways:

- the App Center has a more unified software management experience and better Debian package support
- NVIDIA CUDA is now distributed natively through Ubuntu repositories
- AMD ROCm is now available through Ubuntu repositories as well

If you support AI or HPC workloads on Linux, moving CUDA and ROCm into trusted Ubuntu package channels makes application patch governance much cleaner than vendor-managed side paths.

1. Standardise application delivery

The best Linux patching strategy is boring on purpose:

- use Ubuntu repositories wherever possible
- prefer Canonical-delivered snaps for desktop applications that already fit that model
- minimise manually installed `.deb` files
- aggressively reduce standalone tarballs, shell installers, and random vendor repos

If an application is outside your package governance model, it is outside your patch assurance model too.

2. Use Ubuntu Pro to extend security coverage

Ubuntu Pro matters here because it expands the vulnerability and patch coverage available for Ubuntu packages beyond the small base set most people think about.

For enterprises, **ESM Apps** is especially relevant because a lot of useful application packages live outside the core base OS set. If you are serious about patching Linux applications against the Essential Eight timelines, Ubuntu Pro is not just nice to have.

3. Use Landscape as the patching control plane

Landscape gives you the fleet-level mechanics:

- package inventory
- staged rollout
- reporting
- repository governance
- grouping systems by role or risk

That is important because patching internet-facing application servers and patching user endpoints should not be the same workflow. Landscape lets you separate those rings and prove what happened.

4. Focus on high-risk application classes first

For Ubuntu desktops and jump hosts, the patching priority should usually be:

1. Firefox and Chromium-based browsers
2. email clients and collaboration apps
3. PDF readers and document tooling
4. remote access clients
5. security agents

For servers, the first wave is different:

1. reverse proxies and web servers
2. language runtimes exposed to the internet

3. identity and access components
4. internet-facing management portals

5. Measure coverage, not just success

A patching job that succeeds against the packages you know about is not enough.

You also need to know about the software that escaped your standard channels. On Linux, that usually means:

- manually installed vendor binaries
- developer-downloaded runtimes
- container images that have not been rebuilt
- old utility packages sitting on bastions and admin hosts

Inventory discipline is half of this mitigation.

ISM control mapping

The October 2024 Essential Eight to ISM mapping links this mitigation to these controls:

ISM control	Linux implementation on Ubuntu 26.04 LTS
`ISM-1807`	Patch or remove vulnerable applications on workstations, with priority for internet-facing and user-facing tools.
`ISM-1808`	Patch or remove vulnerable applications on internet-facing servers, including exposed web components and management portals.
`ISM-1698`	Apply vendor application patches within required timeframes using standard package sources and Landscape rings.
`ISM-1699`	Apply rapid remediation for internet-facing applications, especially reverse proxies, browsers, and exposed services.
`ISM-1876`	Patch office productivity suites, email clients, web browsers, and PDF software on user systems.
`ISM-1690`	Maintain an accurate application inventory so vulnerable software can be found quickly.
`ISM-1691`	Verify that application updates have been applied successfully and are not just approved on paper.
`ISM-1905`	Patch or replace unsupported application versions before they become a standing exception.
`ISM-1704`	Apply higher-maturity application patching disciplines consistently across all relevant asset classes.
`ISM-1700`	Govern exceptions and risk acceptance where vendor patches do not exist.
`ISM-1693`	Remove or isolate applications that cannot be patched in a reasonable timeframe.
`ISM-1692`	Use compensating controls when immediate application patching is not possible.
`ISM-1901`	Ensure vulnerable application exposure is reduced through additional controls when patching is delayed.

Where Linux gets awkward

Ubuntu handles repo-managed software well.

The pain starts when organisations allow:

- vendor-specific shell installers
- manually unpacked browsers or agents
- unmanaged developer toolchains
- one-off binaries copied into `/usr/local/bin`

The Essential Eight does not really care that the app was "just a handy Linux utility." If it is vulnerable and reachable, it is in scope.

Compensating controls for third-party software

When an application cannot be patched quickly, use one or more of these mitigations:

- remove internet exposure
- put the application behind a reverse proxy or VPN boundary
- limit access through Teleport or another identity-aware broker
- confine the process with AppArmor
- isolate it in a container or VM
- move users to a supported package source
- remove the software entirely if it no longer has a business case

That last option is underrated.

Commercial and enterprise-friendly additions

Where native Ubuntu controls need help, these are the usual force multipliers:

- **Ubuntu Pro** for broader package security coverage
- **Landscape** for fleet-scale rollout and reporting
- **Tenable**, **Qualys**, or **Rapid7** for vulnerability discovery if you need a dedicated scanning and reporting layer
- **Teleport** to reduce exposure for unpatched administrative or high-risk services while remediation is underway

Teleport is especially useful when the vulnerable component is an admin surface that should not be directly reachable anyway.

A practical Ubuntu pattern

If I were standardising this control for an Ubuntu 26.04 estate, I would start here:

- all standard software delivered via APT or snap

- Ubuntu Pro enabled across the fleet
- Landscape groups for workstations, internal servers, and internet-facing servers
- emergency patch ring for browsers, reverse proxies, and remote access software
- inventory review for manually installed binaries
- AppArmor or isolation for software that cannot be patched immediately

Once that is in place, the patching conversation becomes much more manageable.

The bottom line

Patching applications on Linux is easy right up until you lose packaging discipline.

Ubuntu 26.04 LTS gives you a solid reference stack for this mitigation: **APT**, **snap**, **Ubuntu Pro**, and **Landscape**. Resolute Raccoon improves the story further by pulling more high-value software into official package channels. The hard part is not the tooling. It is insisting that the fleet stays inside a governed software supply chain.

References

- ASD Essential Eight maturity model and ISM mapping (October 2024)
- Ubuntu Pro
- Expanded Security Maintenance
- Landscape documentation
- Ubuntu security notices