PatchMon PATCHMON BLOG

Debian 11 & OpenSSL 3.0 EOL: Audit Your Fleet

Debian 11 LTS ends 31 August 2026, OpenSSL 3.0 on 7 September. How to audit your fleet for every instance hiding in container images, LXC and appliance VMs.

AUTHOR

PatchMon Team

PUBLISHED

14 August 2026

CATEGORIES

Security, Patch Management

READ ONLINE

https://patchmon.net/blog/debian-11-openssl-3-end-of-life

Contents

You have an inventory of servers somewhere. It is probably accurate for the machines someone deliberately built, named and logged. What it almost certainly does not contain is the Debian 11 root filesystem inside the container image your CI pipeline rebuilds every night, the LXC container somebody stopped in 2023 and never deleted, or the vendor appliance VM nobody has logged into since it was commissioned. Those are the ones that stop receiving security updates on 31 August.

Two support windows close in quick succession. Debian 11 "bullseye" reaches the end of Long Term Support on **31 August 2026**; OpenSSL 3.0 reaches the end of its LTS window on **7 September 2026**. You already knew both dates, or could have looked them up in eight seconds. The question this post is about is the one that takes a fortnight to answer: how many of each do you have, and where are they?

The Short Version

> **Debian 11 "bullseye" LTS ends 31 August 2026.** Full support ended two years ago, on 14 August 2024. After 31 August, no more DLAs, no more security-patched packages, nothing.
>
> **OpenSSL 3.0 (LTS) support ends 7 September 2026.** Upstream ships nothing after that, and every subsequent fix becomes a distro-backport question. Note the trap: upgrading Debian 11 to Debian 12 moves you from OpenSSL 1.1.1 onto 3.0.
>
> **The software does not stop working.** Nothing breaks on 1 September, which is precisely why these deadlines get ignored.
>
> **Who is affected:** anyone running Debian 11, anyone on Ubuntu 22.04 (pinned to OpenSSL 3.0.2), and anyone whose container images, LXC templates or appliance VMs were built on either.
>
> **The sprawl problem is real.** OpenSSL 3.2 and 3.3 have already expired; 3.4 expires 22 October 2026 and 3.6 on 1 November. This is a staircase, not a cliff.
>
> **What to do:** audit first, upgrade second. You cannot plan a migration for hosts you have not found.

What Actually Stops When Debian 11 and OpenSSL 3.0 Go EOL

End of life is a supply announcement, not a shutdown. The imprecise version ("Debian 11 dies on 31 August") is easy to dismiss when your Debian 11 boxes are still serving traffic on 1 September.

For **Debian 11**, what stops is the flow of security updates: no more CVE triage against bullseye, no more patched packages, no more Debian LTS Advisories. Your `apt update` will still succeed and your packages will still install. What changes is that from 1 September, every new

vulnerability on that host is permanently unpatched, and the gap between your installed version and a safe one only widens. The contrast is the point: when Copy Fail landed in April (https://patchmon.net/blog/copy-fail-cve-2026-31431-linux-kernel-patch), distro patches followed within days. On an EOL release that pipeline does not run.

The exact wording from the Debian 11 release page (https://www.debian.org/releases/bullseye/) is worth quoting, because people misread the lifecycle: "The Debian 11 life cycle encompasses five years: the initial three years of full Debian support, until August 14th, 2024, and two years of Long Term Support (LTS), until August 31st, 2026."

There is a detail buried in that page which catches people out. Bullseye shipped with ten architectures, but LTS covers only four (https://wiki.debian.org/LTS): i386, amd64, armhf and arm64. If you run Debian 11 on ppc64el, s390x, mips64el, mipsel or armel, your security support did not end this month. It ended on 14 August 2024, and you have been unpatched for two years. Worth checking before you assume you have until month end.

For **OpenSSL 3.0**, what stops is upstream. That does not immediately strand distro users, because distributions backport fixes into the version they shipped, which is covered in detail below. Ubuntu 22.04 is the clearest case: jammy ships openssl 3.0.2 with a long tail of Ubuntu-specific security revisions (https://launchpad.net/ubuntu/jammy/+source/openssl), and standard security maintenance runs to May 2027 (https://ubuntu.com/about/release-cycle). What changes on 7 September is who does the work: from then on Canonical is deriving fixes for a branch upstream no longer touches, for another eight months.

Why This One Is Harder Than It Looks

Debian 11 hides inside things you did not install. Nobody decided to run bullseye in 2026. What happened is that a Dockerfile pinned a base image tag five years ago and the tag still resolves; an LXC container was created once, stopped, and never deleted; a vendor shipped an OVA with a Debian userland inside and called it an appliance; a cloud instance was launched from a 2021 AMI and never rebuilt. In each case the operating system is an implementation detail of something else, which means it is not on the list of operating systems you maintain. Many official language runtime images are Debian-based, so check rather than assume.

Your OpenSSL version is a property of the distro, not the application. It is set by whatever `libssl` the package manager installed, which is set by the distro release, which was set by whoever built the base image. That holds only for software linking against the system library. Node.js embeds its own OpenSSL, some Python builds do too, and plenty of vendor software ships a private `libcrypto.so` where no package manager will ever update it. A host can be fully patched by every measure your tooling checks and still have three different OpenSSL versions resident in memory.

This is a staircase, not a cliff. The OpenSSL release strategy (https://openssl-library.org/policies/releasestrat/index.html) publishes support end dates per branch, and a fleet with mixed versions inherits all of them. 3.2 (23 November 2025) and 3.3 (9 April 2026) are already expired; 3.0 ends on 7 September 2026, 3.4 on 22 October and 3.6 on 1 November. There is a counter-intuitive trap: 3.6 is newer than 3.5, but 3.5 is LTS and supported until 8 April 2030 while 3.6 expires in under three months. Under the current policy, "non-LTS releases after 3.5 will be fully supported for 13 months", so upgrading to the newest branch can shorten your runway. If you are choosing a target this autumn, 3.5 has the long horizon; 4.0 (supported until 14 May 2027) does not.

Finding Debian 11 In Your Estate: Hosts, Docker, LXC

Hosts

`/etc/os-release` exists on essentially every modern distribution. Prefer sourcing it over `lsb_release`, which is a package that frequently is not installed.

```
. /etc/os-release && echo "$ID $VERSION_ID ($VERSION_CODENAME)"
# debian 11 (bullseye)

cat /etc/debian_version      # 11.11, tells you how stale the install is
dpkg --print-architecture    # amd64, tells you whether LTS ever covered it
```

That only answers for hosts already on your list, and only for today; see how PatchMon tracks Debian releases (https://patchmon.net/integrations/debian) for keeping it current. The rest of this section is about the hosts that are not on the list.

Docker

Two questions matter, and only one is about what is running. Start with what your next build will pull, because fixing a host recreated tomorrow achieves nothing:

```
grep -rn --include='Dockerfile*' -iE '^\s*FROM' . | grep -iE 'bullseye|debian:11'
```

Then enumerate the images on disk, where tags pinned years ago quietly persist:

```
docker image ls --format '{{.Repository}}:{{.Tag}}' | while read -r img; do
  printf '%s\t' "$img"
  docker run --rm --entrypoint sh "$img" \
    -c '. /etc/os-release 2>/dev/null && echo "$PRETTY_NAME" || echo "no os-
release"' \
    2>/dev/null || echo "no shell in image"
done
```

That executes each image, so run it on a scratch host; distroless and scratch-based images will report no shell. For running containers, `docker exec "$c" sh -c '. /etc/os-release; echo "$PRETTY_NAME"'` answers directly. See also our Docker monitoring notes (https://patchmon.net/integrations/docker).

LXC and Proxmox

This layer gets missed, and deserves more of your time than Docker: the containers are longer-lived, treated as pets rather than cattle, and frequently older than the hosts running them.

The first mistake is auditing only running containers. `pct list` reports every container with its status, and a stopped container is not a decommissioned one. It is one somebody will start again in six months, rejoining your estate with a 2021 userland and no security updates since August:

```
pct list    # includes stopped containers; audit all of them

for id in $(pct list | awk 'NR>1 {print $1}'); do
  status=$(pct status "$id" | awk '{print $2}')
  printf '%s\t%s\t' "$id" "$status"
  if [ "$status" = "running" ]; then
    pct exec "$id" -- sh -c '. /etc/os-release; echo "$PRETTY_NAME"' 2>/dev/null
  else
    echo "STOPPED, needs offline inspection"
  fi
done
```

For the stopped ones, `pct mount` reads the filesystem without booting the container. Note the caveat from the pct documentation (https://pve.proxmox.com/pve-docs/pct.1.html) before scripting this across a live node: it "will hold a lock on the container and is meant for emergency maintenance only as it will prevent further operations on the container other than start and stop". Do it deliberately, and unmount immediately.

```
pct mount "$id"
. "/var/lib/lxc/$id/rootfs/etc/os-release" && echo "$PRETTY_NAME"
pct unmount "$id"
```

Then audit the templates, because those are what the next container gets built from. Proxmox caches them on disk with the distro version in the filename, so this costs nothing:

```
pveam list local                  # cached templates, version is in the filename
pveam available | grep -i debian  # what is currently on offer upstream
```

On plain LXD or Incus the same two questions apply: `lxc list` includes stopped instances, and the cached image list is the Proxmox template cache equivalent:

```
lxc list -c ns
lxc image list
lxc exec "$NAME" -- sh -c '. /etc/os-release; echo "$PRETTY_NAME"'
```

If you run Proxmox at any scale, our Proxmox integration notes (https://patchmon.net/integrations/proxmox) cover reporting this automatically rather than sweeping by hand.

Appliance and vendor VMs

You cannot upgrade these yourself, because the vendor owns the userland. The action is not `apt`, it is an email asking their plan for 31 August, sent this week rather than in October.

Finding Your OpenSSL Versions

The obvious command is the least complete one:

```
openssl version -a
```

That reports the `libcrypto` the `openssl` CLI binary itself links against. Useful data point, bad summary: nothing guarantees your web server, database or application runtime uses the same library.

Package manager queries, by distribution family:

```
dpkg -l | grep -E '\s(openssl|libssl[0-9])'      # Debian / Ubuntu
rpm -qa | grep -E '^(openssl|compat-openssl)'    # RHEL / Rocky / Alma / Fedora
apk info -v | grep -E 'openssl|libssl'           # Alpine
zypper search -is | grep -i openssl              # SUSE
pacman -Q | grep -i openssl                      # Arch
```

Now the part the package manager cannot tell you. To see what a running process actually has mapped, which is the only answer true at runtime:

```
pid=$(pgrep -f nginx | head -1)
grep -oE '/[^ ]*lib(ssl|crypto)[^ ]*' "/proc/$pid/maps" | sort -u
```

And to find bundled copies no package owns, which are the ones still sitting at 3.0 long after everything else has moved:

```
find / -xdev \( -name 'libssl.so*' -o -name 'libcrypto.so*' \) 2>/dev/null |
  while read -r f; do
    dpkg -S "$f" >/dev/null 2>&1 || echo "UNOWNED: $f"
  done
```

Anything printed there is an OpenSSL copy your patching process does not touch. Runtimes embedding their own are worth checking explicitly:

```
node -p "process.versions.openssl"
python3 -c "import ssl; print(ssl.OPENSSL_VERSION)"
```

A Java service with a bundled native library, a Go binary using cgo against a vendored build, and a commercial agent shipping its own `lib/` directory will not appear in the package queries above. They are found by the `find` sweep or not at all.

What To Do About What You Find

For **Debian 11**, the supported destination is Debian 12 "bookworm", and the upgrading chapter of the bookworm release notes (https://www.debian.org/releases/bookworm/amd64/release-notes/ch-upgrading.en.html) covers the sequence: get to the latest bullseye point release first, remove non-Debian and obsolete packages, change your sources, run a minimal upgrade before the full one, install a kernel metapackage afterwards. Do it in that order: the steps people skip are the pre-upgrade cleanup ones, and those are what turn a routine `dist-upgrade` into an evening.

Two caveats worth planning around.

Bookworm is not a fresh horizon. Debian 12's full support ended on 11 July 2026 (https://www.debian.org/releases/bookworm/) and it has been handed to the LTS team, with LTS running until 30 June 2028 (https://wiki.debian.org/LTS) across i386, amd64, armhf, arm64 and ppc64el. An 11 to 12 upgrade therefore moves you from an expiring LTS release to a live one, buying a little under two years. If you are rebuilding from scratch, weigh going straight to current stable for a full support window instead.

The upgrade lands you on OpenSSL 3.0. Debian 11 ships `openssl 1.1.1w-0+deb11u8`. Debian 12 ships `openssl 3.0.20-1~deb12u2`. So the migration you are running to escape the 31 August deadline puts you on the branch whose upstream support ends on 7 September.

Do not panic about that, and do not let anyone tell you Debian 12 is unsupported a week after you migrate. It is not. Debian's security FAQ (https://www.debian.org/security/faq) is explicit that the project backports fixes into the version it shipped rather than moving to a new upstream release, to avoid API and ABI churn mid-release: "the most important guideline when making a new package that fixes a security problem is to make as few changes as possible". Those

`deb11u8` and `deb12u2` suffixes are that policy in action, and Debian 11 users have relied on it for a while already: OpenSSL 1.1.1 is no longer supported upstream either, and bullseye has been carrying it on Debian's backports without most operators noticing.

What genuinely changes on 7 September is who does the work, and what the arrangement does not cover:

> Every subsequent fix becomes a distro-backport question, so you inherit a dependency on your vendor's effort and its timeliness.
>
> For bookworm that effort sits with the LTS team rather than the main security team, which the Debian LTS wiki (https://wiki.debian.org/LTS) describes as a responsibility handover.
>
> Anything you built, vendored or statically linked against OpenSSL 3.0 yourself is genuinely on its own. Nobody is backporting into your build.
>
> Containers carrying their own OpenSSL rather than the host's are in the same position.

Which is why the `find` sweep above matters more than the `dpkg` query.

Two smaller things. Install `debian-security-support`, which flags packages whose support has ended ahead of the release expiring, and subscribe whoever handles patching to debian-security-announce (https://www.debian.org/security/) so DSAs and DLAs arrive rather than being discovered.

From One-Off Audit To Ongoing Tracking

Every command above answers for one host, one node, or one afternoon. The fleet version is different: not "what is on this box" but "which of my hosts are on bullseye, on an architecture LTS never covered, with an OpenSSL 3.0 library still resident, and has anything appeared since I last checked".

That last clause defeats a manual sweep: an audit is a photograph, and estates are not static. Containers get restored from snapshots, stopped ones get started, someone redeploys from a stale image tag. The bullseye container appearing next Tuesday will not be in the spreadsheet you finish on Friday.

That is an inventory problem, and what PatchMon (https://patchmon.net) exists to solve. Agents report each host's OS release, architecture, installed packages and available updates to one place, so the audit becomes a filter rather than an SSH loop: show every host on Debian 11, group by architecture, watch the count fall as the migration progresses, with next Tuesday's container appearing on its own because collection is continuous. The OpenSSL question works the same way, where the useful output is the spread of versions across the fleet rather than one string. More on tracking patch status across every host (https://patchmon.net/linux-patch-management).

That covers every host you can put an agent on, which in most estates is nearly all of them, and it keeps answering as the estate moves rather than going stale the moment you close the spreadsheet. The `find` sweep above stays the companion for OpenSSL copies that live outside package management entirely, and appliance VMs remain a vendor conversation until they move.

If you want to work through this on a real fleet, PatchMon Cloud (https://patchmon.net/pricing) starts at $1 per host per month with a 14 day trial, and the Community Edition (https://patchmon.net/open-source) is AGPLv3 and free to self-host.

Bottom Line

Audit before you plan. Sweep `/etc/os-release` across every host, image, container and template you can reach this week. A plan built from an incomplete inventory is one that misses hosts.

Check your architectures first. If you run Debian 11 outside i386, amd64, armhf and arm64, you lost security support on 14 August 2024, not this month. Those hosts are the emergency; everything else is a schedule.

Enumerate stopped containers, not just running ones. `pct list` and `lxc list` both include them. A stopped container with a 2021 userland rejoins your estate the moment somebody starts it.

Fix the images and templates, not just the instances, or the host you upgrade today gets recreated from a bullseye image tomorrow. Email appliance vendors in the same pass.

Record the OpenSSL branch spread across the fleet, not one version on one box, paying particular attention to copies no package manager owns.

Neither deadline will cause an outage. That is exactly why they get deferred, and why estates accumulate hosts that quietly stopped receiving patches years ago. The teams that handle end of life calmly are not the ones with the earliest start date; they are the ones who can answer "how many do we have and where" in an afternoon rather than a fortnight.

Run the audit this week. There is still enough time to plan a migration, and not enough to also discover the hosts.

Sources and Further Reading

Debian 11 "bullseye" release information (https://www.debian.org/releases/bullseye/)

Debian 12 "bookworm" release information (https://www.debian.org/releases/bookworm/)

Debian Wiki: LTS, supported releases, dates and architectures (https://wiki.debian.org/LTS)

Debian Wiki: LTS/Using, configuring apt for LTS and options when support ends (https://wiki.debian.org/LTS/Using)

Debian 12 release notes: upgrading from Debian 11 (https://www.debian.org/releases/bookworm/amd64/release-notes/ch-upgrading.en.html)

Debian Security FAQ: why fixes are backported rather than rebased (https://www.debian.org/security/faq)

Debian security information and the debian-security-announce mailing list (https://www.debian.org/security/)

Debian Security Bug Tracker (https://security-tracker.debian.org/tracker/)

Debian package: openssl in bullseye (https://packages.debian.org/bullseye/openssl)

Debian package: openssl in bookworm (https://packages.debian.org/bookworm/openssl)

OpenSSL release strategy and per-branch support end dates (https://openssl-library.org/policies/releasestrat/index.html)

OpenSSL vulnerabilities index, by release branch (https://openssl-library.org/news/vulnerabilities/)

Ubuntu release cycle and maintenance windows (https://ubuntu.com/about/release-cycle)

Launchpad: openssl source package in Ubuntu 22.04 "jammy" (https://launchpad.net/ubuntu/jammy/+source/openssl)

Proxmox VE: pct container toolkit manual page (https://pve.proxmox.com/pve-docs/pct.1.html)

endoflife.date: Debian release lifecycle summary (https://endoflife.date/debian)

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