



The .NET Cliff: What Happens When .NET 8 and .NET 9 Stop Getting Patched

A straight-talk guide to the deadline, the rules that get harder to meet, and the two ways to stay ahead of it.

The .NET Cliff: What Happens When .NET 8 and .NET 9 Stop Getting Patched

A straight-talk guide to the deadline, the rules that get harder to meet, and the two ways to stay ahead of it.

CONTENTS

1. The short version	3
2. Why this deadline is unusual	4
3. What actually changes on November 10	5
4. Where this actually bites first	7
5. The way out of that bind	8
6. Your realistic options	9
7. Turning this into a plan	10
8. Where HeroDevs fits	12
Sources & methodology	13

1. The short version

One date. Two frameworks. Zero patches after that.

On **November 10, 2026**, Microsoft ends support for both **.NET 8** (the current Long-Term Support release) and **.NET 9** (the current Standard-Term Support release), on the same day. That's not a scheduling coincidence you can wait out. It's the date both patch streams stop, permanently, for every application still running on them.

If your organization runs .NET 8 or .NET 9 in production, this guide walks through what "end of support" actually changes, why regulated teams tend to feel it first through compliance and insurance rather than a breach, and the two realistic paths forward: migrate to .NET 10, or bridge the gap with continued security support.



Nov 10

2026, the last day Microsoft ships security patches for .NET 8 and .NET 9



2

frameworks losing support the same day: an LTS and an STS release together



56

CVEs disclosed after .NET 6 reached end of support, fixed in NES for .NET 6

What this guide covers: what "end of support" really stops, why the compliance and insurance exposure usually shows up before an actual breach does, how a single unpatched framework compounds risk across everything built on it, and the two paths off the cliff, including a 90-day plan.

2. Why this deadline is unusual

LTS and STS releases don't normally expire together

Microsoft's .NET release cadence is built to stagger risk: Long-Term Support (LTS) releases like .NET 8 get three years of patches, while Standard-Term Support (STS) releases like .NET 9 got 18 months, so an STS release went dark six months before the LTS release that came before it.

November 10, 2026 is the first time that offset disappears. .NET 8 (shipped November 2023) and .NET 9 (shipped November 2024) both hit their support ceiling on the same calendar day, because Microsoft extended STS support from 18 to 24 months in September 2025, starting with .NET 9. That moved .NET 9's end of support from May 12, 2026 to the day .NET 8's three-year window closes. The practical effect is that teams who spread risk by running a mix of the two most recent releases (a common, previously reasonable strategy) get no staggering benefit this cycle. Whatever your .NET 8/.NET 9 footprint looks like today, all of it needs a plan for the same date. It will not be the last time, either. Under the 24-month policy, the next STS release, .NET 11, is on track to reach end of support alongside .NET 10 in November 2028.

End of support itself isn't new: .NET 6 reached end of support on its own on November 12, 2024, six months after .NET 7. Teams that treated it as a soft deadline, assuming Microsoft or the ecosystem would extend it, or that unpatched CVEs on an EOL runtime wouldn't attract attention, spent the following year explaining an unremediated risk to auditors, insurers, and in some cases customers. November 10, 2026 is that same deadline, with twice the surface area.

What "end of support" does *not* mean

It doesn't mean .NET 8 or .NET 9 stop running. Applications keep executing exactly as they did the day before. That's what makes the deadline easy to underweight internally: nothing visibly breaks on November 11. What changes is quieter, and for most regulated organizations, more consequential. That's covered next.

3. What actually changes on November 10

The patch stream stops. The vulnerabilities don't.

Security researchers and attackers don't stop looking for .NET vulnerabilities because Microsoft stopped patching them. New CVEs affecting .NET 8 and .NET 9 code paths will continue to be discovered. The only thing that changes is that there's no longer an official fix. November 10 is itself a Patch Tuesday, and Microsoft normally ships one final release of an ending version on its last Patch Tuesday. Anything disclosed after that has no Microsoft fix. HeroDevs' security team calls this the **forever-day problem**: once a runtime goes end-of-life, every newly disclosed vulnerability against it has no path to a fix.

The patch stream covers more than the runtime. A .NET 8 or .NET 9 install is made of four parts, and all four stop getting fixes: the .NET runtime, the ASP.NET Core shared framework, the Windows Desktop runtime for WPF and Windows Forms, and the .NET SDK, including MSBuild. .NET 6 shows what that looks like after end of support. Fixes HeroDevs shipped in NES for .NET 6 landed in every one of those parts: a WebSocket decompression denial of service and a Linux diagnostics socket exposure in the runtime (6.0.44), an HTTP/3 control-stream denial of service in Kestrel (6.0.45), heap out-of-bounds writes in WPF font and glyph rendering (6.0.44), and an MSBuild download filename spoofing fix and a dotnet watch origin check in the SDK (6.0.45 and 6.0.46).

Four practical consequences

- **Vulnerability scanners keep flagging you, indefinitely.** Your SCA and vulnerability-management tooling will continue to surface CVEs against .NET 8/9, and unlike a supported runtime, none of them will ever move to "patched." Every scan after November 10 adds to a backlog that only grows.
- **Scanners can also miss real exposure.** Microsoft does not publish CVEs or patches for end-of-life versions of .NET, so a new CVE lists only the supported versions as affected, and a scanner matching against that record has nothing to flag on .NET 8 or .NET 9. CVE-2025-55315, the 9.9-rated request smuggling flaw in the ASP.NET Core Kestrel server disclosed on October 14, 2025, is the example: .NET 6 was vulnerable, the CVE did not list it, and Microsoft did not patch it. HeroDevs shipped the fix in NES for .NET 6.0.39 three days later.
- **Severity scores don't stay put.** A CVE that looked low-priority at CVSS 4 or 5 can become urgent overnight if it's added to CISA's Known Exploited Vulnerabilities (KEV) catalog or its EPSS exploitation-probability score spikes, and that reclassification can happen years after initial disclosure. On a supported runtime, that just triggers a patch cycle. On an EOL runtime, there's no patch to apply.

- **Low-severity findings can chain into high-severity ones.** A modest issue in one component can become the missing link in a multi-step attack path once combined with something else in your stack. The individual CVSS score never reflects that combination, only your own risk analysis does, and auditors increasingly expect to see that analysis documented.

This isn't unique to .NET. Every open source ecosystem carries a long tail of packages that are already end of life, with known CVEs and no upstream fix coming. .NET 8 and .NET 9 are about to add two more widely used runtimes to that list, on the same day, at enterprise scale.

4. Where this actually bites first

The bill usually comes due in an audit or a renewal, not a breach

Most teams picture EOL risk as "we might get hacked." For regulated organizations, the more immediate exposure is that **running unsupported software quietly puts you out of compliance with frameworks you already have to satisfy**, discovered on a schedule you don't control.

FRAMEWORK	WHAT IT REQUIRES	WHY EOL .NET BREAKS IT
PCI DSS 4.0.1	Applying critical security patches within a defined window of their release, and confirming on a regular basis that software in scope remains supported by its vendor.	There's no patch to apply within that window, because the vendor no longer issues one. The vendor-support confirmation becomes a documented failure rather than a routine checkbox. (Confirm the current requirement numbers and exact timelines with your compliance team, since PCI DSS is periodically revised.)
HIPAA Security Rule	45 CFR §164.308(a)(1)(ii)(A) and (B): risk analysis and risk management; "reasonable and appropriate safeguards" for protected health information.	An unpatchable vulnerability in a system touching PHI is difficult to defend as a "reasonable and appropriate" safeguard once no official fix exists.
SOC 2	Documented risk analysis and compensating controls for known exposures.	Auditors treat EOL software as a known, ongoing risk decision, not an unknown gap, which raises the bar for what compensating controls you can show.
ISO/IEC 27001:2022	Annex A, Control 8.8: timely evaluation and remediation of technical vulnerabilities.	An EOL component with no remediation path is, by definition, a visible control failure.

Cyber insurance follows the same logic

Insurers have been pricing this risk directly. Coalition's 2023 Cyber Claims Report found that organizations running end-of-life software were **three times more likely to suffer a security incident**, regardless of company size, and that policyholders with even one unresolved critical vulnerability faced **33% higher odds of filing a claim**.

Carriers can deny or limit claims tied to EOL software through several routes: named exclusions that apply regardless of negligence, rescission if the application misstated security controls, and warranty clauses that void coverage when a breach involves an unsupported component. Many policies also fund restoration only to the pre-incident state, not an upgrade, so "we'll fix it after the breach" isn't a strategy insurance will pay for.

5. The way out of that bind

Auditors and underwriters don't require you to be current. They require you to be supported.

This is the detail most teams miss under deadline pressure: none of the frameworks above actually say "you must be on the latest version." They say you must be able to demonstrate **vendor support and a patching commitment** for whatever you run. That's a lower bar than a full migration, and it's the bar extended, vendor-backed support is built to clear.

What underwriters and auditors typically accept:

- A signed support agreement that names the specific component and version
- A documented, dated record of CVE remediation for that version
- A vendor attestation letter confirming the scope and cadence of that support

In other words, the compliance and insurance exposure described above isn't resolved only by finishing a migration before November 10. It's resolved by having a supported patch stream for whatever you're running, on whatever timeline you're running it. The real choice is between migrating on a realistic timeline and converting the gap into a documented, auditable support relationship in the meantime.

6. Your realistic options

Migrate to .NET 10, bridge with extended support, or both

Every organization running .NET 8 or .NET 9 in production is choosing between these two paths right now, whether or not that choice has been made explicit. Most enterprises end up doing both: migrating the applications that can realistically move, and bridging the ones that can't.

Path A: Migrate to .NET 10

.NET 10 is the current LTS release and the intended long-term destination for anything still on .NET 8 or .NET 9. Microsoft's own release notes cite JIT and runtime performance improvements, expanded post-quantum cryptography support, modern TLS 1.3 across all major platforms, the ability for console apps to natively create container images without a Dockerfile, and Entity Framework Core 10 updates including vector search and native JSON support. Those features are worth the upgrade on their own, deadline or not. Enterprise application upgrades still routinely take longer than teams initially scope for, particularly once transitive dependencies, third-party library compatibility, and regression testing enter the picture. If migration is feasible on your compliance timeline, this is the right primary path. If it isn't, Path B covers the alternative.

Path B: Bridge with Never-Ending Support (NES) for .NET

NES for .NET is a secure, drop-in replacement for the unsupported .NET 6, 8, or 9 runtime. No code changes required. It restores the supported patch stream auditors and underwriters look for, while your migration proceeds on a timeline you control instead of one Microsoft's calendar sets for you.

- **Direct visibility into the .NET security process.** HeroDevs is a corporate sponsor of the .NET Foundation and participates in the .NET Security Group alongside Microsoft, Red Hat, IBM, and Canonical, the group responsible for coordinating pre-disclosure patching and simultaneous .NET releases. Members receive source patches about a week before public disclosure, so NES for .NET builds ship on the same Patch Tuesday as Microsoft's.
- **Already proven at this task.** NES for .NET has shipped fixes for 62 CVEs across nine .NET 6 releases, 56 of them disclosed after Microsoft stopped patching .NET 6.
- **No re-platforming required.** Same runtime behavior, same deployment model, on both Windows and Linux. The only thing that changes is that patches keep coming.
- **OpenVEX statements your scanners can read.** HeroDevs publishes OpenVEX statements with NES for .NET releases, so scanners report which CVEs a build fixes and which ones do not apply, instead of an open-ended list of findings against an end-of-life runtime.
- **Backed by ongoing ecosystem investment.** HeroDevs' \$20 million Open Source Sustainability Fund supports maintainers across open source ecosystems, including .NET.

7. Turning this into a plan

A realistic runway to November 10

The teams that handle this well don't wait for a single "migrate or bridge" decision across the whole portfolio. They triage application by application, on a short, concrete timeline. Here's a workable structure regardless of how much runway is left when you start it.

STEP 1



Inventory

🕒 weeks 1-2

- List every production application on .NET 8 or .NET 9, including internal tools and vendor-delivered systems
- Flag which ones touch regulated data (PHI, cardholder data, PII) or sit inside a SOC 2/ISO 27001 audit boundary
- Note which have an active, staffed migration path to .NET 10 already underway

STEP 2



Decide, per application

🕒 weeks 3-4

- Migrate if .NET 10 compatibility work is scoped, resourced, and realistically finishable before the deadline
- Bridge with NES if the migration timeline extends past November 10, if the application is a candidate for retirement rather than a rewrite, or if compliance coverage needs to be locked in now while migration work continues in parallel

STEP 3



Execute and document

🕒 through November 10

- For migration candidates: lock scope, assign owners, and track against the deadline like any other compliance-driven project
- For bridge candidates: put a signed support agreement and CVE remediation record in place before November 10, not after. This is the artifact your next audit or policy renewal will ask for.
- Brief compliance, security, and, where relevant, your cyber insurance broker on the plan for each application, not just the ones already migrated

STEP 4**After November 10**

- Confirm every remaining .NET 8/9 application is either migrated or has an active support agreement in place. There is no third state that satisfies an auditor.
- Continue migration work for bridged applications on the timeline the bridge was meant to buy, rather than letting "supported" become the permanent answer

8. Where HeroDevs fits

If any application won't make the deadline, that's what NES for .NET is for

You don't need to have this fully sorted out before you talk to us. Most conversations start with a single application or a handful of them: the ones where migration clearly won't finish in time, or where a signed support agreement needs to exist before your next audit or policy renewal, whichever comes first.

Talk to HeroDevs about NES for .NET

Secure, drop-in patching for .NET 6, 8, and 9. No code changes, no re-platforming, and built by a team with direct visibility into the .NET Security Group's disclosure process.

Get a Custom Quote: herodevs.com/support/dot-net-nes



Sources & methodology

- Microsoft, "Announcing .NET 10," ".NET 8 and .NET 9 will reach End of Support on November 10, 2026," ".NET STS releases supported for 24 months" (September 16, 2025), and "Announcing the .NET Security Group" (October 14, 2025), .NET Blog, devblogs.microsoft.com/dotnet.
- Coalition, Inc., 2023 Cyber Claims Report, as independently reported via Businesswire (May 2023): organizations with unresolved critical vulnerabilities found to be 33% more likely to experience a cyber claim; end-of-life software linked to threefold higher incident likelihood.
- U.S. Department of Health and Human Services, HIPAA Security Rule, 45 CFR §164.308(a)(1)(ii)(A) and (B).
- ISO/IEC 27001:2022, Annex A, Control 8.8 (Management of Technical Vulnerabilities).
- CISA Known Exploited Vulnerabilities (KEV) Catalog; FIRST.org Exploit Prediction Scoring System (EPSS).
- HeroDevs, "HeroDevs Joins The .NET Foundation to Secure and Grow the Open Source Ecosystem" and the .NET Foundation Sponsor Spotlight (dotnetfoundation.org): \$20 million Open Source Sustainability Fund and .NET Security Group participation.
- HeroDevs NES for .NET 6.0.x release notes (docs.herodevs.com/net/release-notes/dotnet-6-0): 62 CVEs fixed across releases 6.0.38 (June 4, 2025) through 6.0.46 (September 9, 2026), 56 of them disclosed after .NET 6 reached end of support.
- HeroDevs, "FAQ about CVE-2025-55315, the 9.9-rated CVE in ASP.NET Core" (November 5, 2025).

This guide is provided for general informational purposes and does not constitute legal or compliance advice. Organizations subject to PCI DSS, HIPAA, SOC 2, ISO 27001, or other regulatory frameworks should confirm specific obligations, including current requirement details and effective dates, with their own compliance, legal, and audit teams before relying on this guide.