QUANTIFYING THE REAL RISKS OF UNSUPPORTED OPEN SOURCE SOFTWARE

herodevs

@herodevs

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Open source software (OSS) forms the backbone of enterprise applications, digital infrastructure, and development pipelines. Once considered a cost-saving supplement to commercial tooling, OSS has become essential for innovation, time-to-market acceleration, and platform flexibility. However, alongside its growth lies an underaddressed risk: the long-term lifecycle management of these open source components.

When open source projects reach end-of-life (EOL), organizations are left without access to security patches, bug fixes, or compliance coverage. In the absence of formal support, the responsibility to maintain, secure, and document these components shifts directly to the organization—often without internal capacity, visibility, or process maturity to do so effectively.

This white paper examines the strategic, operational, and compliance risks associated with unsupported OSS, provides a quantitative view of the cost of inaction, and outlines a path forward using third-party long-term support (LTS) as a risk mitigation and modernization tool. It is written for technical and business leaders responsible for platform integrity, security posture, and technology strategy.

THE LIFECYCLE BLIND SPOT IN ENTERPRISE OPEN SOURCE

The explosion of open source adoption has fundamentally reshaped enterprise software delivery. According to the 2024 Synopsys Open Source Security and Risk Analysis (OSSRA) report, 96% of audited codebases contain open source components, and in many cases, more than 70% of the code in an application is open source.

What's less understood, however, is how these components are maintained over time—and who assumes responsibility when upstream projects no longer provide support. The open source community, despite its innovation velocity and collaborative strength, is not structured to offer long-term maintenance guarantees. Individual contributors and maintainers, often unpaid, operate with limited resources and without contractual obligations to the enterprises that depend on their work.

This creates a fundamental misalignment. Enterprises require software systems to remain stable and secure for 5–10 years. The open source ecosystem, in contrast, operates on faster, more flexible lifecycles, with no obligation to support older versions beyond an established end-of-life date.

UNDERSTANDING END-OF-LIFE: WHAT IT REALLY MEANS

When an OSS component reaches its declared end-of-life, it enters a state where:

- No further security patches or CVE remediations will be released
- No bug fixes or compatibility updates are provided
- Community engagement, forum activity, and issue tracking sharply decline
- Projects may be archived or formally deprecated

The component may continue functioning technically, but operational and security guarantees cease. This often goes unnoticed by engineering teams focused on delivery—until a compliance audit, security scan, or vulnerability disclosure forces the issue.

The impact of running EOL OSS is not merely theoretical. The Synopsys OSSRA report noted that 45% of analyzed applications in 2024 contained at least one open source component that was no longer actively maintained. These components are often embedded deep within dependency trees, making them difficult to detect and replace without significant refactoring.

FOUR CATEGORIES OF RISK INTRODUCED BY UNSUPPORTED OSS

Security Vulnerabilities Without Remediation

Unsupported OSS cannot be updated through official channels. When a new CVE is identified in an EOL component, no upstream fix is issued. This means vulnerabilities remain unpatched, indefinitely.

In a 2023 study of software supply chain risk, 48% of known vulnerabilities in enterprise environments were traced back to unmaintained or EOL open source libraries. These vulnerabilities are particularly dangerous because they are publicly known and easily exploitable, yet often remain in production for months—if not years—due to migration complexity.

Security teams face a difficult decision: manually patch an outdated component and assume the engineering burden, or leave it in place and accept the exposure. Neither path is sustainable at scale.

Compliance and Audit Failures

Modern compliance frameworks (e.g., SOC 2, ISO 27001, HIPAA, PCI-DSS, NIST 800-53) increasingly include mandates requiring that organizations demonstrate usage of actively supported and secure software. End-of-life components, by definition, fail to meet these criteria.

This risk has become more visible in the past 18 months, particularly in industries subject to regulatory scrutiny. A 2023 Linux Foundation report revealed that over 70% of federal agencies identified unsupported OSS components as a material risk to their compliance standing3.

Independent software vendors are also impacted. When onboarding large enterprise customers or government clients, vendors are routinely asked to submit Software Bill of Materials (SBOM) documentation and verify support status. The inclusion of EOL components can delay or disqualify contracts.

Operational Fragility

Unsupported software components increase time-to-remediation when incidents occur. As upstream support and documentation wane, internal knowledge silos emerge. Engineering teams must dedicate resources to reverse-engineering patches, resolving dependency conflicts, or rewriting integrations—distracting from core development objectives.

Moreover, the longer an organization remains on an EOL version, the more difficult future upgrades become. Dependency lock-in and diverging API contracts introduce fragility into the stack, increasing the likelihood of cascading failures or downtime during modernization.

Strategic and Business Disruption

In high-stakes settings—mergers and acquisitions, due diligence processes, cybersecurity insurance renewals—running unsupported OSS can be viewed as a material liability. Organizations unable to produce documentation proving ongoing support are often subject to elevated risk ratings or contract delays.

In one case study from 2023, a SaaS company pursuing enterprise partnerships failed a vendor security review due to the presence of AngularJS and Lodash 4—both of which had reached EOL. Although the applications were stable, the security team at the prospective client required assurance of active support or formal LTS coverage. The absence of either delayed onboarding by 90 days and introduced significant unplanned remediation work.

WHY TRADITIONAL MITIGATIONS FAIL

Upgrading Immediately

In theory, migrating to the latest version of an OSS project addresses the risk. In practice, this approach is often unworkable—especially when breaking changes are introduced across versions, or when systems are tightly coupled to legacy behavior.

Frameworks such as Spring, Angular, and Node.js have introduced major architectural changes between versions. These changes require months of planning, QA, and staff re-training. In a 2023 Gartner study, over 65% of software modernization efforts exceeded budget, and more than 40% failed to meet scope objectives.

Independent software vendors are also impacted. When onboarding large enterprise customers or government clients, vendors are routinely asked to submit Software Bill of Materials (SBOM) documentation and verify support status. The inclusion of EOL components can delay or disqualify contracts.

Forking and Maintaining Internally

Some teams attempt to fork unsupported libraries and self-patch. While this may suffice for short-term fixes, it introduces long-term burdens:

- Engineering teams become responsible for tracking vulnerabilities and writing patches
- Integration with external tooling (e.g., scanners, CD pipelines) degrades
- Knowledge is concentrated among a few team members, creating key-person risk

Forking also increases divergence from the community version, complicating any future migration or rebase.

Accepting the Risk

The third (and most common) approach is passive acceptance. Teams continue running EOL software, often without full awareness of the implications, until a triggering event occurs—an audit failure, a breach, or a dependency conflict that halts development.

This reactive posture is both operationally and reputationally costly. The financial cost of remediating a known but unaddressed vulnerability is typically 4–6x higher post-incident than during a planned upgrade or support engagement.

THE ROLE OF COMMERCIAL LONG-TERM SUPPORT

Long-term support (LTS) provided by a commercial partner offers a structured alternative to these inadequate options. This approach delivers sustained security coverage and risk mitigation for OSS components that are no longer maintained by their original authors or foundations.

At HeroDevs, we call this **Never-Ending Support**.

Our model provides:

- **SLA-backed security patching** for EOL OSS versions
- **Ongoing vulnerability monitoring** and response
- **Audit-grade documentation,** including patch histories and compliance reports
- **Compatibility-preserving updates,** tested against your environment
- **Version-specific support timelines,** enabling planned modernization

This allows organizations to remain on their current version while maintaining compliance, security integrity, and development velocity.

STRATEGIC BENEFITS OF NEVER-ENDING SUPPORT

Organizations that adopt commercial LTS gain:

- **Reduced upgrade pressure,** enabling modernization on their timeline, not the community's
- **Improved audit readiness,** with documented patch coverage and SLA references
- **Decreased breach risk,** through proactive CVE remediation
- **Higher engineering efficiency,** by eliminating unplanned upgrade work
- **Financial predictability,** compared to multi-quarter migration budgets

These benefits are most pronounced in complex environments with multiple applications or significant legacy footprint.

END-OF-LIFE IS A BUSINESS RISK, NOT JUST A TECHNICAL ONE

The widespread reliance on open source software has outpaced most organizations' ability to maintain it responsibly. As OSS projects reach end-of-life, the absence of formal support structures introduces silent, compounding risks—across security, compliance, and operations.

HeroDevs provides a path forward. Our Never-Ending Support program helps engineering, security, and compliance teams maintain control over their OSS environments long after official support ends.

In a world where every system relies on open source, the ability to extend support for critical components is no longer a convenience—it's a strategic imperative.

ABOUT HERODEVS

HeroDevs delivers commercial-grade, SLA-backed support for end-of-life open source software. Our Never-Ending Support offering helps organizations secure, maintain, and document their legacy OSS environments. We serve customers across industries including finance, healthcare, infrastructure, and software.

For more information, visit herodevs.com or contact us at support@herodevs.com.

REFERENCES

1. Synopsys. 2024 Open Source Security and Risk Analysis Report.
2. Forrester. Securing the Software Supply Chain. 2023.
3. Linux Foundation. The State of Open Source in Government. 2023.
4. Gartner. Software Modernization Trends Survey. 2023.