https://www.proconference.org/index.php/gec/article/view/gec40-00-026
DOI: 10.30890/2709-1783.2025-40-00-026

UDC 004.2

CYBERSECURITY-DRIVEN APPROACH TO END-OF-LIFE SOFTWARE MANAGEMENT: ADDRESSING VULNERABILITY RISKS THROUGH STANDARDIZED EOL PROTOCOLS

Demianchuk S.
Independent researcher
ORCID: 0009-0000-2838-9052
USA, Cary NC 27513

Abstract. *Software End-of-Life (EoL) management represents a critical yet often overlooked aspect of the software development lifecycle, particularly in the open-source ecosystem where decentralized development and varied maintenance models create unique challenges. The lack of standardized approaches exposes organizations to significant security, compliance, and sustainability risks. This paper examines the definitions, taxonomy, and challenges of EoL in software, with particular emphasis on open-source contexts. It highlights the benefits of a standardized framework, including improved communication, trust, planning, and supply chain security. The study demonstrates that the standardization of software EoL represents a critical advancement for ensuring sustainable open-source ecosystems, regulatory compliance, and enhanced resilience across the software supply chain.*

Key words: *EoL, end-of-life, end of support, end of security support, end of sales, software lifecycle, cybersecurity, vulnerability management*

Introduction.

The rapid development of a new software for the critical infrastructure and enterprise systems has created an urgent need for standardized approaches to software lifecycle management, particularly regarding End-of-Life (EoL), End-of-Security-Support (EoSSec), End-of-Sale (EoS) and other states of hardware, software, services and specifications. Organizations today face significant risks when software reaches its end-of-life status, as it no longer receives critical security updates or patches. The open-source ecosystem, characterized by its distributed development model and varied maintenance structures, presents unique challenges that existing proprietary software lifecycle frameworks cannot adequately address due to lack of the standardized way to report the end-of-life information. Software is malleable and resistance to physical decay - fundamentally challenge traditional lifecycle management approaches. Unlike hardware, software evolves iteratively through updates, patches, and modifications to align with changing user needs and technological landscapes. This dynamic nature complicates the identification of

deterioration indicators and safety thresholds [1], requiring novel frameworks for assessing when software has truly reached its end of life rather than simply needing another update.

Recent compliance frameworks have begun recognizing the importance of EOL management. PCI DSS 4.0, effective March 31st, 2024, requires programs to track end-of-life software and create remediation plans [2]. However, the open-source community lacks a unified approach to communicating and managing lifecycle information, leading to security vulnerabilities, compliance failures, and inefficient resource allocation across the software supply chain.

This research addresses three fundamental questions. First, how to establish definition of software end-of-life that includes the nuances of open-source development lifecycle. Second, what management strategies effectively mitigate the risks associated with end-of-life protocols. Third, how to design a standardized protocols for end-of-life which provide clear, actionable lifecycle information.

Main text

Defining Software End-of-Life

The terminology surrounding software lifecycle transitions varies significantly across the industry. End-of-life refers to software products that are no longer sold or renewed, while end of support marks the cessation of support services, including patches for critical vulnerabilities. This distinction becomes particularly complex in open-source contexts where traditional sales models don't apply, and support may come from community volunteers, commercial vendors, or hybrid models. The software industry lacks formal decommissioning practices, typically retiring products by terminating support and creating "abandonware" that persists in use despite vendor abandonment. Netscape Communicator exemplifies this phenomenon: following Mozilla's 2006 EOL declaration, the community fork SeaMonkey continued development [3], highlighting the disconnect between vendor-defined EOL and actual software utility. This practice, replicated across numerous platforms through emulators and unofficial maintenance, demonstrates that ceasing support fails to ensure retirement while creating security vulnerabilities in still active but

unsupported systems [4].

The software lifecycle follows predictable patterns from inception through retirement. This lifecycle starts with the software development phase, followed by deployment, maintenance to fix bugs and improve functionality, and eventually reaches a stage where maintenance is no longer feasible or cost-effective.

End-of-Life taxonomy

The following taxonomy for a shared understanding of the definition is proposed:

1). Vendor: Any entity (organization, community, or individual) responsible for creating or maintaining a product. This includes open-source projects, not just commercial companies.

2). Product: Any named deliverable (software, hardware, services, specifications, etc.), regardless of origin, license, or distribution model.

3). Product Lifecycle: The full journey of a product from release (General Availability) to retirement (End-of-Life). Lifecycles may include different stages of support (full, maintenance, security-only), vary by vendor/product type, and evolve over time.

The taxonomy of the key lifecycle milestones is proposed:

1). End-of-Sales (EoS): The last date a product can be purchased directly from vendor channels. After EoS, support may continue, but no new sales.

2). End-of-Security-Support (EoSSec): The last date the vendor provides security patches. Past this point, products become vulnerable, making this a crucial compliance and risk management marker.

3). End-of-Life (EoL): The final point when the vendor ends all support (development, updates, security fixes, technical assistance). Customers must migrate to supported alternatives before this date.

Benefits and Challenges of Standardized EoL

The open-source ecosystem presents unique lifecycle management challenges. Organizations must track various releases and updates for each OSS technology, determining appropriate responses when OSS is no longer supported while

maintaining security and compliance. Community-driven development models mean that lifecycle decisions may be distributed across multiple stakeholders with varying priorities and resources [5].

Main challenges:

- ✓ Diverse Structures: Projects differ widely, making one-size-fits-all frameworks difficult.
- ✓ Limited Resources: Many projects lack funding or staff to manage lifecycle programs.
- ✓ Decentralized Decisions: Consensus is harder in distributed, volunteer-driven projects.
- ✓ - Awareness & Adoption Gaps: Some maintainers may be unaware, hesitant, or resistant to formalized approaches.
- ✓ Resistance to Change: Preference for flexibility may slow adoption of standards.

The following are a few benefits of a standardized EoL in the supply chain:

- ✓ Clear Communication: Improves coordination between maintainers, contributors, users, and customers.
- ✓ Trust & Reliability: Signals transparency and responsible management, boosting confidence in projects and vendors.
- ✓ Stability & Planning: Helps organizations plan roadmaps, allocate resources, and reduce risks.
- ✓ Sustainability: Encourages long-term project health, attracts contributors, and supports funding.
- ✓ Business Advantages: Streamlines product management, builds customer trust, and ensures smoother transitions to new solutions.

Discussion.

The standardization of the software End-of-Life (EoL) addresses fundamental sustainability challenges in open-source software development. By providing clear lifecycle information, projects can better manage contributor expectations and resource allocation. Organizations can make informed decisions about dependency

adoption and support investments, potentially directing resources toward critical projects approaching EOL. Software that has reached end of life may not follow industry rules, compliance standards, or contractual responsibilities, standardization of the EoL data directly addresses these concerns by providing auditable lifecycle tracking that satisfies regulatory requirements while enabling proactive security management.

The standardization of the software End-of-Life (EoL) represents a significant advancement in supply chain security. By automatically linking lifecycle status with vulnerability databases, organizations can prioritize remediation efforts and allocate resources more effectively.

Summary and conclusions

This research presents a comprehensive analysis of software End-of-Life (EoL) management challenges and proposes the standardization of the end-of-life software lifecycle. By bridging the gap between open-source development practices and enterprise lifecycle management requirements, the standardized End-of-Life (EoL) approach facilitates sustainable software ecosystem growth while enhancing security posture across the software supply chain. Adoption of such approach represents a crucial step toward professionalizing open-source lifecycle management without sacrificing the flexibility and innovation that characterize open-source development. As software increasingly underpins critical infrastructure and business operations, standardized lifecycle management becomes not just beneficial but essential for ecosystem health and security.

References:

1. Assaad, Z., & Henein, M. (2022). End-of-life of software: How is it defined and managed? arXiv. https://doi.org/10.48550/arXiv.2204.03800

2. XEOL, "End-of-Life Software and Compliance," XEOL Blog. [Online]. Available: https://www.xeol.io/post/end-of-life-software-and-compliance.

3. Jeffrey, C., & Franco, J. (2020). What Ever Happened to Netscape Navigator? Techspot. https://www.techspot.com/article/2077-netscape-navigator/

4. McGraw, G. (2004). Software Security. IEEE Computer Society, 4, 1540–7993.

5. Santos, O. (2023). Establishing standardized end-of-life and end-of-support programs for software and hardware. *Becoming a Hacker*. https://becomingahacker.org/establishing-standardized-end-of-life-and-end-of-support-programs-for-software-and-hardware-e3e231898e02

6. Santos, O., Schmidt, T., Roguski, P., Middlekauff, A., Cao, F., Demianchuk, S., Rock, L., Murphy, J., Hagen, S., Chari, S., & Schaffer, T. (2025, April 24). OpenEoX: A standardized framework for managing End of Life and other product lifecycle information [Technical report]. OASIS Open. https://docs.oasis-open.org/openeox/standardization-framework/openeox-standardization-framework-technical-report.pdf

Article sent: 25.08.2025