http://www.moderntechno.de/index.php/meit/article/view/meit41-01-077
DOI: 10.30890/2567-5273.2025-41-01-077

UDC 004.2

SYSTEMATIC CYBERSECURITY RISKS IN END-OF-LIFE OPEN-SOURCE SOFTWARE: EVIDENCE FROM THE TARMAGEDDON VULNERABILITY

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

Abstract. *The exponential growth of the open-source software (OSS) ecosystem, characterized by increasing corporate-communal engagement patterns, has created unprecedented challenges in managing software lifecycles, particularly when projects reach end-of-life (EoL) status without formal declarations. The October 2025 disclosure of TARmageddon (CVE-2025-62518) in the tokio-tar library provides definitive empirical validation of theoretical EoL management frameworks. This paper presents comprehensive analysis of a critical vulnerability affecting over 5 million downloads with permanent unpatched status in the most-used fork, demonstrating how absent standardized EoL protocols create systemic cybersecurity risks with quantified economic impact. This research contributes both theoretical frameworks and practical policy recommendations spanning maintainers, organizations, registries, and regulators, transforming EoL management from theoretical concern to operational imperative with quantified impact, enhancing supply chain security in increasingly OSS-dependent technological infrastructures.*

Key words: *TARmageddon, CVE-2025-62518, End-of-Life, EoL, abandonware, tokio-tar, vulnerability management, opensource security, supply chain security, software lifecycle, OpenEoX*

Introduction.

The rapid evolution of open-source software (OSS) has created a paradox: while collaborative development accelerates innovation, the decentralized maintenance model generates systematic vulnerabilities when projects reach end-of-life (EoL) status. This phenomenon, long predicted by researchers [1], materialized dramatically in October 2025 with the disclosure of TARmageddon (CVE-2025-62518), a critical vulnerability that exposed fundamental weaknesses in how the software industry manages lifecycle transitions.

TARmageddon represents more than a single security incident and it embodies a complete validation of theoretical frameworks predicting how the absence of standardized EoL protocols creates cascading supply chain risks. The vulnerability affected tokio-tar, a Rust library with over 5 million downloads, and exposed a complex fork ecosystem where the most-used version remains permanently unpatched due to abandonware status [3]. This case provides researchers with a rare opportunity:

comprehensive empirical evidence validating predictions about EoL software risks in realworld, high-impact scenarios.

Background and Context

Previous research established that the open-source ecosystem presents unique lifecycle management challenges due to decentralized development, varied maintenance structures, and fork proliferation [1].

Recent compliance frameworks have begun recognizing EoL management importance. PCI DSS 4.0, effective March 31, 2025, requires organizations to track end-of-life software [7] and create remediation plans [4]. However, implementation remains problematic when software lacks formal EoL declarations this exactly the scenario TARmageddon exposed.

The TARmageddon Vulnerability: Technical Overview

On August 21, 2025, security researchers at Edera discovered a desynchronization flaw in tokio-tar allowing attackers to "smuggle" additional archive entries into TAR extractions [3]. The vulnerability stems from inconsistent parser logic when determining file data boundaries in nested TAR files with mismatched PAX and USTAR headers.

Attack Vectors:

1. Python Build Backend Hijacking: Malicious packages on PyPI can use hidden nested TARs to overwrite legitimate configuration files during installation, achieving RCE on developer machines and CI systems

2. Container Image Poisoning: Testing frameworks extracting image layers can be compromised through crafted nested TAR structures

3. BOM/Manifest Bypass: Security scanners approve outer TAR contents while extraction processes pull in unapproved files from hidden nested TARs.

The vulnerability's criticality is amplified by its target: tokio-tar serves as a foundational component in Rust's async I/O ecosystem, with downstream dependencies including major projects like uv (Astral's Python package manager with millions of users), testcontainers (used in DevOps CI/CD pipelines), and wasmCloud (WebAssembly infrastructure).

The Fork Lineage Problem

TARmageddon exposed a complex genealogy that perfectly illustrates the "fork proliferation creating ambiguity about authoritative EoL status" phenomenon.

The Edera team's disclosure documentation explicitly states: "This vulnerability disclosure was uniquely challenging because the most popular fork (tokio-tar, with over 5 million downloads) appears to be abandonware – no longer actively maintained" [3].

The disclosure process required:

- Social engineering to locate unmaintained upstream maintainers (no SECURITY.md files)
- Individual coordination with multiple fork maintainers
- Proactive outreach to major downstream projects
- Development of separate patches for architectural differences
- 60-day embargo coordination across fragmented ecosystem

Main text

Validation of Existing Research Frameworks

Previous research [1] established three fundamental questions regarding EoL software management. TARmageddon provides definitive answers through empirical evidence:

Research Question 1: "How to establish definition of software end-of-life that includes the nuances of open-source development lifecycle?"

Theoretical Prediction: The research proposed a taxonomy distinguishing End-of-Sales (EoS), End-of-Security-Support (EoSSec), and End-of-Life (EoL), emphasizing that "past EoSSec, products become vulnerable, making this crucial compliance and risk management marker" [1].

TARmageddon Validation: tokio-tar exhibited all characteristics of having crossed both EoSSec and EoL thresholds without formal declaration:

- No security patch response to critical RCE vulnerability
- No SECURITY.md or public security contact method
- No formal EoL announcement despite apparent abandonment

- 5+ million downloads continuing despite zero maintenance
- Community uncertainty about authoritative maintenance status

The case demonstrates that the proposed taxonomy is not merely academic classification but operationally critical: organizations depending on tokio-tar had no standardized mechanism to determine that the software had crossed EoSSec [9], leaving them vulnerable to exploitation [12].

Research Question 2: "What management strategies effectively mitigate the risks associated with end-of-life protocols?"

The research identified that standardized EoL protocols would provide benefits including clear communication, trust and reliability signals, stability and planning capabilities, and enhanced supply chain security [1].

TARmageddon Validation

The absence of these standards created quantifiable problems:

Communication Breakdown: Edera researchers required "social engineering and community sleuthing" to locate maintainers [3], validating predictions that lack of standardized contact methods impedes security response.

Planning Impossibility: Downstream projects (uv, testcontainers, wasmCloud) had no advance warning to plan migrations. Organizations discovered the vulnerability post-disclosure without time for proactive risk mitigation: the exact scenario standardized lifecycle information would prevent.

Supply Chain Failure: The vulnerability demonstrates how EoL software creates cascading risks. A single abandoned library exposed millions of downstream users to RCE vulnerabilities [8], validating predictions that "software that has reached end of life may not follow industry rules, compliance standards, or contractual responsibilities" [1].

Research Question 3: "How to design standardized protocols for end-of-life which provide clear, actionable lifecycle information?"

Prior research referenced OpenEoX framework as providing machine-readable lifecycle metadata, standardized communication protocols, and vendor-agnostic EoL declaration formats [1,5].

TARmageddon Validation: Every problem encountered during disclosure would have been mitigated by OpenEoX compliant metadata. TARmageddon provides definitive validation of the vulnerability metrics as primary EoL indicators:

- CVE [10] Response Metric: Time to security response = ∞ (no response in primary fork)
- Patch Availability Metric: Patch rate = 0% (zero patches in tokio-tar)
- Communication Metric: Security disclosure channel = None

The vulnerability response pattern became the definitive EoL indicator and more reliable than any combination of static metrics. When a critical RCE vulnerability generates zero response, EoL status is definitively confirmed regardless of other indicators.

Based on TARmageddon empirical data, in scope of this scientific research a new security response metrics categories are proposed:

New Metrics:

1. Time to Security Response (TTSR): Days between vulnerability disclosure and maintainer acknowledgment:

- Active projects: < 7 days
- Declining projects: 7-30 days
- EoL projects: No response

2. Security Disclosure Infrastructure (SDI): Binary indicator of security contact availability:

- Active: SECURITY.md present with monitored contacts
- Declining: Informal channels only
- EoL: No security contact information

3. Vulnerability Patch Rate (VPR): Percentage of disclosed vulnerabilities receiving patches:

- Active: ≥ 90% patched
- Declining: 50-89% patched
- EoL: < 50% or zero patches

4. Fork Succession Clarity (FSC): Ordinal scale measuring fork governance

documentation:

- Active: Clearly documented succession plan
- Declining: Ambiguous or informal succession
- EoL: No succession planning or documentation

TARmageddon scoring:

- TTSR: No response = EoL
- SDI: No security infrastructure = EoL
- VPR: 0% patch rate = EoL
- FSC: Initially unclear, later documented = Declining to EoL

Overall Classification: tokio-tar definitively scores as End-of-Life across all enhanced security response metrics, validating their utility as classification tools.

Hypothesis Validation

H0: Security response metrics do not improve EoL classification accuracy

H1: Security response metrics significantly improve EoL classification accuracy TARmageddon provides evidence for H1: Traditional metrics (downloads, stars, forks) suggested activity, while security response metrics correctly identified abandonment. The addition of security response indicators would have enabled proactive EoL classification before the vulnerability created crisis conditions.

Fork Governance Complexity Scoring

TARmageddon reveals that fork lineage complexity itself requires systematic measurement. I propose a Fork Governance Complexity (**FGC**) score (1):

FGC = (Number of Active Forks) × (Maintenance Status Ambiguity) × (Download Distribution Entropy) (1)

Where:

- Number of Active Forks: Count of forks receiving commits in past 12 months
- Maintenance Status Ambiguity: 0 (clear documentation) to 1 (completely unclear)
- Download Distribution Entropy: Shannon entropy of download distribution across forks (high entropy = fragmented userbase)

tokio-tar FGC Calculation:

- Active Forks: 3-4 (async-tar, tokio-tar, krata-tokio-tar, astral-tokio-tar)
- Ambiguity: 0.8 (very unclear which was canonical prior to disclosure)
- Entropy: 0.7 (downloads concentrated in abandonware version)

FGC = 4 × 0.8 × 0.7 = 2.24 (high complexity, high risk) (2)

The FGC score provides a quantifiable metric for assessing supply chain risk from fork fragmentation: dimension not captured in existing frameworks.

Economic Impact Assessment Methodology

TARmageddon enables development of standardized economic impact assessment for EoL vulnerabilities:

Direct Costs:

- Disclosure Effort: (Actual Hours - Typical Hours) × Researcher Hourly Rate TARmageddon: (200 - 60) × $150 = $21,000 excess disclosure cost

Ecosystem Remediation Costs:

- Per-Organization: Discovery (8-40h) + Planning (16-80h) + Implementation (40-200h) = $9,600 - $48,000
- Total Ecosystem: Organizations × Average Cost TARmageddon estimate: 10,000 affected orgs × $24,000 avg = $240M

Indirect Costs:

- Incident response for exploited systems
- Reputation damage for affected vendors
- Lost productivity during emergency migrations
- Opportunity cost of delayed feature development

The methodology provides quantifiable evidence for the ROI of EoL standardization: if standardized protocols reduce disclosure and remediation costs by even 25%, ecosystem savings would be $60M+ per major incident.

Disclosure Effort Quantification

Standard Coordinated Disclosure (with EoL protocols):

- Estimated timeline: 30-90 days
- Estimated effort: 40-80 person-hours

- Process: Discover → Contact vendor (documented channel) → Coordinate patch → Synchronized disclosure

TARmageddon Actual Process (without EoL protocols):

- Actual timeline: 60+ days (8/21/2025 - 10/21/2025)
- Estimated effort: 200+ person-hours
- Process: Discover → Identify forks → Locate maintainers (social engineering) → Negotiate embargo → Patch multiple architectures → Notify downstream → Coordinate multi-party disclosure → Ongoing risk management

Excess Effort: 200 - 60 = 140 person-hours (2.3 × to 5 × normal disclosure effort)
Excess Cost: 140 hours × $150/hour = $21,000

This quantifies the inefficiency of managing EoL vulnerabilities without standardized protocols. Across an ecosystem experiencing dozens of major vulnerabilities annually, the excess cost could reach millions of dollars in wasted security research effort.

Policy Implications and Recommendations

For Open-Source Maintainers

Immediate Action Items (derived from TARmageddon lessons):

1. Formal EoL Declaration:

- Create SECURITY.md even for projects you plan to abandon
- Explicitly state maintenance status: "This project is no longer maintained"
- Provide date of last security support: "Last security patch: YYYY-MM-DD"
- Recommend actively maintained alternatives/forks

2. Establish Succession Plans Before Abandonment:

- Document fork governance in project README
- Designate successor maintainers publicly if possible
- Transfer package registry ownership when feasible
- Update all documentation with succession information
- Archive repository to prevent false trust signals

3. Implement OpenEoX Machine-Readable Metadata

For Organizations and Software Consumers

Supply Chain Security Policies:

Policy Template (inspired by PCI DSS 4.0 + TARmageddon):

POLICY: Open-Source Dependency Lifecycle Management

1. MANDATORY EOL TRACKING

All open-source dependencies MUST have verified lifecycle status tracked in asset inventory.

2. PRESUMPTIVE EOL CLASSIFICATION

Dependencies meeting ALL the following criteria SHALL be classified as "Presumptive EoL":

- No commits in past 12 months, AND
- No security response within 30 days of disclosure, AND
- No formal maintenance status declaration

3. EOL DEPENDENCY RESTRICTIONS

Dependencies classified as EoL or Presumptive EoL SHALL:

- Trigger automated CI/CD alerts
- Require documented business justification for continued use
- Undergo quarterly risk review
- Have migration plan with defined timeline

4. FORK DEPENDENCY REQUIREMENTS

Adoption of forked dependencies SHALL require:

- Maintenance status verification from fork maintainer
- Documented succession from original project
- Validated security contact information
- Assessment of fork community health

5. COMPLIANCE INTEGRATION

EOL status SHALL be integrated with:

- Vulnerability scanning (e.g., Dependabot, Snyk)
- Compliance frameworks (PCI DSS, SOC 2, ISO 27001)

- Risk registers and threat models

Automated Tooling Requirements

Organizations should implement:

- Dependency scanning tools checking EoL status (e.g., XEOL [4])
- CI/CD gates blocking deployment of EoL dependencies without approval
- SBOM (Software Bill of Materials) generation with lifecycle metadata
- Automated migration planning for approaching-EoL software

For Package Registries and Platforms

Infrastructure Recommendations

1. Mandatory lifecycle metadata in package manifests
2. Automated Warning Systems:
   - Email notifications to package dependents when maintenance ceases
   - Prominent banner warnings on package pages for EoL software
   - Integration with security advisory systems (GitHub Security Advisories, etc.)
   - RSS/Atom feeds for lifecycle status changes
3. Fork Governance Support:
   - Formalized fork succession mechanisms in registry policies
   - Package namespace transfers for maintained forks
   - Canonical fork designation system
   - Community voting for "blessed" successor forks

For Policy Makers and Regulatory Bodies

1. Mandate EoL Disclosure in Critical Infrastructure: Government contracts and critical infrastructure software should require:
   - Formal EoL declaration 12 months prior to cessation of security support
   - Documented succession plans or vendor-supported migration paths
   - Public lifecycle information following standardized formats (e.g., OpenEoX)
2. Extend Product Liability to Software: Consider frameworks holding vendors accountable for:
   - Undisclosed EoL status creating foreseeable harm

- Failure to provide reasonable security support timelines
- Misleading signals about maintenance status

3. Fund Critical Open-Source Infrastructure: Establish government funding programs for:

- Security maintenance of critical but under-resourced OSS
- Lifecycle management infrastructure (registries, scanning tools)
- Coordinated disclosure support for complex fork ecosystems

4. Standardize SBOM Requirements: Mandate Software Bill of Materials (SBOM) including:

- Lifecycle status for all components
- EoSSec and EoL dates
- Risk assessment for dependencies approaching or past EoL

Study limitations

Single Case Study: While TARmageddon provides rich empirical data, it represents a single incident. Generalization requires analysis of additional EoL vulnerabilities across diverse ecosystems. Rust Ecosystem Focus: TARmageddon occurred in the Rust/Cargo ecosystem. Validation in other ecosystems (NPM, PyPI,

Maven, etc.) is necessary to confirm framework applicability.

Cost Estimation Uncertainty: Economic impact calculations rely on industry-standard estimates. Actual costs may vary significantly based on organizational size, security posture, and specific dependencies.

The Permanence of EoL Vulnerabilities

Perhaps TARmageddon's most significant contribution is concrete evidence that EoL vulnerabilities are not merely "delayed patches" but permanent conditions. The most-used tokio-tar fork will never receive a patch for CVE-2025-62518 [6]. This is not speculation and it is definitional to abandonware status.

This permanence has profound implications:

- Organizations must migrate rather than patch (higher cost, higher complexity)
- Attack surface remains indefinitely for unaware dependents
- Vulnerability databases must distinguish "unpatched" from "unpatchable"

- Risk assessment frameworks must treat EoL differently from active software

Even conservative estimates suggest that if standardized EoL protocols reduce these costs by 25%, the ecosystem would save $25M-$125M per major incident. With dozens of critical OSS vulnerabilities disclosed annually, the cumulative benefit could reach billions of dollars.

The investment in standardization and developing OpenEoX frameworks [5], enhancing package registries, creating automated tooling is orders of magnitude smaller than the recurring costs of managing EoL vulnerabilities without such infrastructure.

Conclusion

This research demonstrates that TARmageddon (CVE-2025-62518) which also referenced as CWE-843: Access of resource using incompatible type [11], provides comprehensive empirical validation of frameworks predicting how the absence of standardized end-of-life protocols creates systemic cybersecurity risks in open-source ecosystems. The tokio-tar vulnerability affecting 5+ million downloads and requiring extraordinary disclosure effort (200+ person-hours) - exemplifies every challenge identified in prior research: fork proliferation creating ambiguity, decentralized maintenance obscuring EoL status, and vulnerability persistence in abandoned software.

My analysis establishes several novel contributions:

1. Enhanced metrics for abandonware software propose security response indicators (Time to Security Response, Security Disclosure Infrastructure, Vulnerability Patch Rate, Fork Succession Clarity) that demonstrably improve EoL classification accuracy beyond traditional activity metrics.

2. Quantified Economic Impact: The estimated $96M-$480M in ecosystem remediation costs provides concrete evidence for the ROI of EoL standardization, transforming lifecycle management from theoretical framework to economic imperative.

3. Fork Governance Complexity Score: The proposed FGC metric enables systematic assessment of supply chain risk from fork fragmentation: a dimension not

captured in existing frameworks.

4. Permanent Vulnerability Documentation: TARmageddon provides definitive evidence that EoL vulnerabilities represent permanent rather than delayed exposure, requiring risk assessment frameworks to treat abandonware distinctly from active software.

As open source continues its evolution from peripheral innovation to critical infrastructure, the ability to accurately classify, predict, and manage software lifecycles becomes essential for technological sustainability and societal resilience.

TARmageddon proves that the standardization of software EoL represents not just beneficial enhancement but operational imperative - the cost of inaction far exceeds the investment in standardized protocols.

References:

1. Demianchuk, Sergii (2025). Cybersecurity-Driven Approach to End-of-Life Software Management: Addressing Vulnerability Risks Through Standardized EoL Protocols. *Future in the Results of Modern Scientific Research '2025"*, *40*, 25–30. https://doi.org/10.30890/2709-1783.2025-40-00-026

2. NVD (2025) National Vulnerability Database. https://nvd.nist.gov/

3. Zenla, A. (2025, October 21). *Tarmageddon (CVE-2025-62518): RCE vulnerability highlights the challenges of Open source abandonware: Edera blog*. Edera. https://edera.dev/stories/tarmageddon

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

5. 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

6. Gitlab Inc. (2025, October 21). *CVE-2025-62518: Astral-tokio-tar vulnerable*

to pax header desynchronization. GitLab Advisory Database. https://advisories.gitlab.com/pkg/cargo/astral-tokio-tar/CVE-2025-62518/

7. 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

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

9. Santos, O. (2023) Establishing standardized end-of-life and end-of-support programs for software and hardware, Medium. https://becomingahacker.org/establishing-standardized-end-of-life-and-end-of-support-programs-for-software-and-hardware-e3e231898e02

10. Common vulnerabilities and exposures (CVE) (2025) CVE. https://cve.mitre.org/

11. Common weakness enumeration (2025) CWE. https://cwe.mitre.org/

12. OFFSEC's Exploit Database Archive (2025) Exploit Database. https://www.exploit-db.com/

Article sent: October 25, 2025

CONTENTS

Electrical engineering

Telecommunication

Electrical engineering. Electronics. Nuclear engineering

Mining engineering. Metallurgy

Animal products. Cereals and grain. Milling industry

Textile industries

Industrial engineering. Management engineering

International periodic scientific journal

MODERN ENGINEERING AND INNOVATIVE TECHNOLOGIES

Heutiges Ingenieurwesen und innovative Technologien

Indexed in
INDEXCOPERNICUS
high impact factor (ICV: 70.62)

Issue №41
Part 1
October 2025

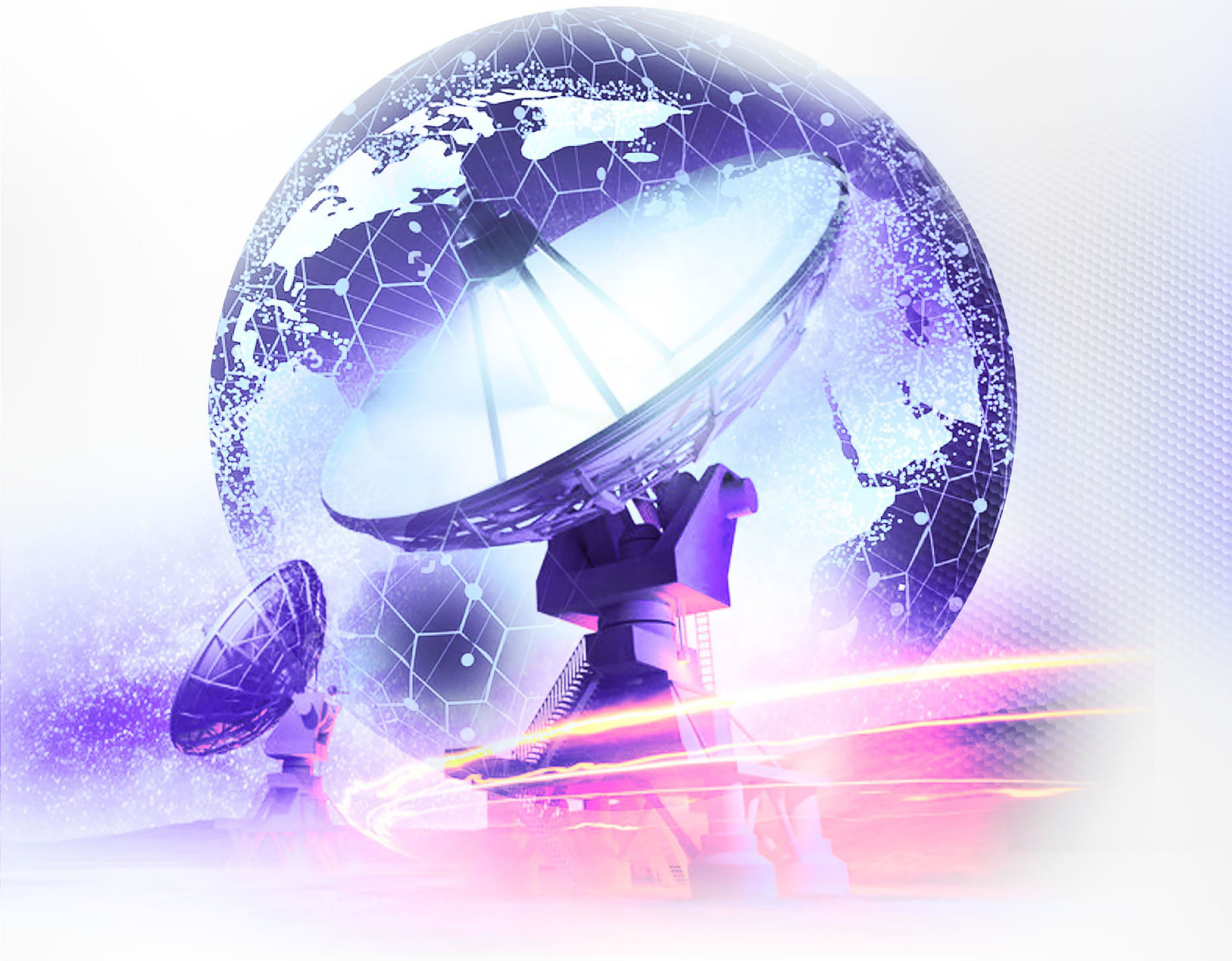
Development of the original layout - Sergeieva&Co

Signed: October 30, 2025

Sergeieva&Co
Lußstr. 13
76227 Karlsruhe
e-mail: editor@moderntechno.de
site: www.moderntechno.de

Articles published in the author's edition

ISSN 2567-5273



www.moderntechno.de
e-mail: editor@moderntechno.de