The Collective Fragility Paradox

Rethinking Third-Party Risk in Software Supply Chains

Trevor Kavanaugh

Vice President, Third-Party Risk Management

January 2026

This paper represents independent analysis and is not an official position of any organization.

Table of Contents

Executive Summary: Managing Dependencies, Not Parties

The Multi-Billion Dollar Wake-Up Call

Between December 2020 and July 2024, the financial services industry experienced a series of technology incidents with combined direct costs conservatively estimated at $30-40 billion, with some analyses suggesting total economic impact—including indirect costs, opportunity losses, and long-tail remediation—exceeding $200 billion.[1] These incidents exposed a fundamental flaw in how we approach third-party risk management. The SolarWinds supply chain attack, Log4j vulnerability, Kaseya VSA ransomware campaign, 3CX supply chain compromise, MOVEit mass exploitation, xz Utils infiltration attempt, and CrowdStrike global outage were not failures of vendor oversight—they were failures of dependency management.

Traditional Third-Party Risk Management (TPRM) frameworks organize risk around legal entities: direct vendors (third parties), their vendors (fourth parties), and in sophisticated programs, fifth parties. But modern technology incidents do not respect organizational boundaries. They propagate through technical dependencies that cut across vendor relationships, regulatory jurisdictions, and risk management silos.

The industry manages "parties" but should be managing "dependencies."

Three Dimensions of Hidden Risk

This paper proposes that modern vendor dependencies create risk across three distinct but interconnected dimensions:

1. Supply Chain Risk: What's In The Product

When a critical vulnerability was discovered in Log4j in December 2021, it affected the majority of Java applications globally—with industry analyses estimating 60-64% of Java applications were vulnerable.[2] The vulnerability was not in products that financial institutions directly procured—it was embedded 5, 10, or even 20 levels deep in dependency chains.

The xz Utils backdoor attempt of 2024 revealed a qualitatively different threat: a nation-state-level actor spent 2.5 years patiently building trust as a maintainer of a widely-used compression library before inserting a backdoor targeting SSH authentication.[3] The compromise was discovered accidentally by a Microsoft engineer investigating performance anomalies—not through any systematic monitoring. Had it reached production, it would have affected every major Linux distribution. The Open Source Security Foundation warned this "may not be an isolated incident," suggesting a reusable attack pattern against volunteer-maintained critical infrastructure.[3]

Key Statistics:

- Apache Log4j appears in over 17,000 packages on Maven Central alone.[4]
- Analysis of modern applications shows approximately 60% of vulnerabilities come from transitive dependencies (dependencies of dependencies).[5]
- Most organizations lack comprehensive software composition visibility.[6]

The Challenge: Traditional vendor assessments focus on what vendors do with data and how they secure it. But they rarely examine what components vendors embed in their software, creating blind spots that span the entire industry.

2. Software Delivery Risk: How It Gets Updated

The SolarWinds attack in December 2020 demonstrated how software update mechanisms can become attack vectors. Nation-state actors compromised the build process for SolarWinds Orion software, inserting malicious code into legitimate updates distributed to approximately 18,000 organizations, including 9 U.S. federal agencies.[7]

The CrowdStrike incident of July 19, 2024, showed that even security vendors can create systemic risk through faulty updates. A defective update to CrowdStrike's Falcon Sensor crashed approximately 8.5 million Windows systems globally,[8] causing what Microsoft called "the largest outage in the history of information technology."[9]

Key Statistics:

- SolarWinds: ~18,000+ organizations installed compromised software.[10]
- SolarWinds: 9 federal agencies confirmed compromised.[11]
- CrowdStrike: 8.5 million devices affected.[12]
- CrowdStrike: Commercial banking identified as one of most impacted sectors.[13]
- Estimated $10+ billion in financial losses from CrowdStrike incident.[14]

The Challenge: TPRM programs extensively evaluate vendors' security controls but often overlook the mechanisms by which vendors distribute updates. Yet software delivery systems represent critical trust relationships that, when compromised, can impact thousands of organizations simultaneously.

3. Cloud/Infrastructure Concentration Risk: Where It Runs

Perhaps the most insidious form of hidden dependency risk is infrastructure concentration. Financial institutions may have diversified vendor portfolios with hundreds of third-party relationships, appearing to reduce concentration risk. But when examining where those vendors' applications actually run, apparent diversity collapses into concentration.

The Collective Fragility Paradox: Each individual decision to outsource to specialized providers is rational—leveraging economies of scale, accessing expertise, reducing costs. But when every organization makes the same rational decision, the industry collectively converges on a handful of critical infrastructure providers: AWS, Microsoft Azure, Google Cloud for compute; Cloudflare and Akamai for CDN; specialized payment processors; major cybersecurity vendors.

Key Insight from Recent Incidents:

- Kaseya VSA ransomware attack (July 2021): 50-60 managed service providers compromised, affecting 800-1,500 downstream businesses—demonstrating how MSP relationships create a 15-25x attack force multiplier invisible to traditional TPRM.[15]
- MOVEit Transfer vulnerability (May 2023): 2,773+ organizations affected, 95.8 million individuals' data exposed, estimated $15.8 billion in costs.[16]
- 3CX supply chain attack (March 2023): 600,000+ organizations affected, first documented "supply chain of a supply chain" attack with a five-level dependency chain.[17]
- UK FCA data (2022-2023): Third-party incidents identified as leading cause of operational incidents.[18]

Industry Statistics:

- Approximately 29% of all breaches attributable to third-party attacks.[19]

- 97% of major US banks experienced exposure through third- or fourth-party relationships in 2024—though only a handful of vendors were actually attacked, illustrating the multiplicative effect of shared dependencies.[20]

The Challenge: Traditional concentration risk management focuses on avoiding over-reliance on individual vendors. But it does not address the reality that hundreds of different vendors may all depend on the same underlying infrastructure, creating systemic concentration that vendor diversification strategies fail to mitigate.

Regulatory Frameworks Are Catching Up—Slowly

Financial regulators have evolved TPRM guidance substantially over the past decade, with major milestones including:

- **2013**: OCC Bulletin 2013-29 distinguished "critical activities" requiring heightened oversight and introduced comprehensive lifecycle management.[21]
- **2021**: Basel Committee's Principles for Operational Resilience included first explicit regulatory recognition of fourth-party risk management.[22]
- **2023**: Interagency Guidance (OCC Bulletin 2023-17 / Federal Reserve SR 23-4 / FDIC FIL-23-2023) harmonized federal banking regulator expectations and emphasized board accountability.[23]
- **2025**: EU Digital Operational Resilience Act (DORA) entered into application (adopted December 2022), creating direct oversight framework for critical ICT service providers and mandatory concentration risk management.[24]

However, regulatory guidance still exhibits a fundamental gap: It addresses third-party relationships (and increasingly, fourth-party relationships) but does not provide practical frameworks for managing the nth-party dependencies that characterize modern software supply chains, software delivery systems, and cloud infrastructure concentration.

Notable Quote from UK FCA: Following the CrowdStrike incident, the UK Financial Conduct Authority observed that "even organisations that comply with ISO 27001, SOC 2, and NIST CSF were still caught unawares."[25] Compliance with traditional frameworks did not prevent impact from dependency risks.

The Compliance Framework Problem

Financial institutions extensively rely on SOC 2 reports as a cornerstone of vendor assurance. But SOC 2 itself has limitations for addressing modern dependency risks:

What SOC 2 Addresses Well:

- Organizational security controls.
- Access management and monitoring.
- Change management processes.
- Incident response capabilities.
- Business continuity planning.

What SOC 2 Wasn't Designed to Address:

- Software composition and component dependencies.
- Security of software build and delivery mechanisms.

- Infrastructure provider dependencies (vendors' vendors).
- Open source component governance.
- Concentration risks from shared infrastructure.

The Unchanged Framework: The core Trust Services Criteria governing SOC 2 have remained largely unchanged since 2017,[26] despite the emergence of major supply chain and dependency risks. While the AICPA updated "points of focus" in 2022, the fundamental criteria do not require comprehensive dependency visibility, software supply chain security assessment, or infrastructure concentration analysis.

Historical Context: The SOC framework emerged from the AICPA's response to market misuse of SAS 70 (Statement on Auditing Standards No. 70). Companies began demanding "SAS 70 reports" from all vendors regardless of whether financial reporting was relevant, and the AICPA formalized this by creating SOC 2 in 2011.[27] This history is instructive: SOC 2 was designed to address organizational security controls, not to map complex dependency chains or assess systemic concentration risk.

The Technical SME Gap

A critical challenge facing TPRM teams is the gap between traditional vendor management expertise and the technical depth required to evaluate modern software supply chains, delivery mechanisms, and cloud architectures.

Traditional TPRM Staffing:

- Compliance professionals.
- Vendor contract specialists.
- Risk analysts.
- Business relationship managers.

Emerging Technical Requirements:

- Software composition analysis understanding.
- Cloud architecture and shared responsibility models.
- Software delivery pipeline security (CI/CD).
- Open source governance.
- Infrastructure-as-code evaluation.
- API security architecture.

Examiner Evolution: Regulatory examiners are increasingly asking technical questions about software delivery mechanisms, infrastructure dependencies, and supply chain visibility that traditional TPRM teams struggle to answer effectively. This creates examination findings and remediation expectations that require capabilities TPRM functions were not built to provide.

A Framework for Dependency Risk Management

Rather than replacing existing TPRM frameworks, this paper proposes augmenting them with three parallel dependency risk dimensions:

Risk Dimension	Traditional TPRM Question	Dependency Risk Question
Supply Chain	"Who is the vendor and what do they do?"	"What components are in the vendor's product?"
Delivery	"How does the vendor secure their environment?"	"How does the vendor distribute updates to my environment?"
Infrastructure	"What are our critical vendor relationships?"	"What infrastructure do our critical vendors depend on?"

The chapters that follow examine each dimension in detail, explore the structural limitations of current assurance frameworks, and consider what mature dependency risk management looks like in practice.

Call to Action

The tens of billions of dollars in incidents between 2020 and 2024 have demonstrated that traditional third-party risk management frameworks, while necessary, are insufficient for managing the complexity of modern technology dependencies.

This is a multi-stakeholder problem requiring a multi-stakeholder solution. Financial institutions cannot solve this alone. Progress requires coordinated evolution across:

- **Regulators**: Creating demand signals for dependency transparency (SBOM requirements, supply chain attestation).
- **Assurance frameworks**: Updating attestation scope to address software composition and delivery mechanisms.
- **Vendors and fintechs**: Generating SBOMs, adopting secure development practices, providing infrastructure transparency.
- **Financial institutions**: Building internal technical capability to consume and act on dependency information.

For individual institutions, this means augmenting traditional tools with:

- Software composition visibility.
- Software delivery risk assessment.
- Infrastructure dependency mapping.
- Technical SME integration into TPRM teams.

The good news: Many of the tools and techniques required already exist in application security, DevSecOps, and cloud architecture disciplines. The challenge is organizational—integrating technical depth into traditionally compliance-focused TPRM functions.

The question for financial institutions: Will you wait for the next major incident to expose a dependency risk you did not know you had, or will you proactively build the capability to see beyond vendor relationships to the dependencies that determine your actual risk exposure?

The chapters that follow trace how we arrived at this structural mismatch—and what it will take to close the gap.

Endnotes - Executive Summary

[1] Aggregate cost estimate methodology: SolarWinds (~$100M total, combining $18-19M direct remediation and $90M+ in cumulative legal/regulatory/insurance costs per Arnold & Porter analysis); Log4j (estimates range from $12B to $100B+ depending on methodology and scope—lower bound based on conservative industry estimates of global enterprise response costs, upper bound includes indirect impacts across the estimated 64% of Java applications affected); MOVEit ($15.8B, per Emsisoft breach cost analysis covering 2,773+ organizations and 93M+ individuals); CrowdStrike ($10B+ for Fortune 500 companies alone, per Parametrix study; ~$5.4B for top 500 US companies excluding Microsoft). Conservative estimates placing emphasis on documented direct costs total approximately $30-40B; upper-bound estimates incorporating indirect impacts, long-tail remediation, and opportunity costs across all affected organizations exceed $200B.

[2] Impact estimates vary by methodology. CISA Cyber Safety Review Board, "Review of the December 2021 Log4j Event," July 2022, documented that 35,000+ Java packages (8% of Maven Central) were affected. Industry analyses from Contrast Security and others estimated 60-64% of Java applications contained the vulnerability. The exact percentage depends on methodology (analyzing packages vs. deployed applications vs. vulnerable code paths), but all sources agree the impact was extraordinarily widespread. Available at: https://www.cisa.gov/sites/default/files/publications/CSRB-Report-on-Log4-July-11-2022_508.pdf

[3] The xz Utils backdoor incident (CVE-2024-3094, disclosed March 2024) involved a sophisticated supply chain attack where a malicious actor gained maintainer access to the xz Utils compression library through years of social engineering, then inserted a backdoor targeting SSH authentication on Linux systems. The backdoor was discovered by a Microsoft engineer investigating performance anomalies before widespread deployment to production systems. See: CISA Alert, "Reported Supply Chain Compromise Affecting XZ Utils Data Compression Library," March 29, 2024; Ars Technica, "What we know about the xz Utils backdoor that almost infected the world," April 2024.

[4] Apache Log4j Maven Central package statistics cited in: "From SolarWinds to Log4j," Check Point Security blog. Available at: https://blog.checkpoint.com/security/from-solarwinds-to-log4j-the-global-impact-of-todays-cybersecurity-vulnerabilities/

[5] Endor Labs, "State of Dependency Management 2024," available at: https://www.endorlabs.com/state-of-dependency-management. Multiple industry analyses confirm that the majority of vulnerabilities in modern applications originate in transitive dependencies rather than direct dependencies. See also: Snyk, "State of Open Source Security Report 2024" for corroborating data on transitive dependency risk.

[6] Synopsys 2024 Open Source Security and Risk Analysis (OSSRA) Report: While 96% of codebases contain open source, comprehensive software composition visibility remains limited. See also: Endor Labs, "State of Dependency Management 2024" survey findings.

[7] "Lessons Learned from the SolarWinds Cyberattack," Arnold & Porter, June 2021. Available at: https://www.arnoldporter.com/en/perspectives/advisories/2021/06/lessons-learned-from-the-solarwinds-cyberattack

[8] "2024 CrowdStrike-related IT Outages," Wikipedia. Available at: https://en.wikipedia.org/wiki/2024_CrowdStril
related_IT_outages

[9] Microsoft statement on CrowdStrike incident, July 2024, cited in multiple sources including UK FCA analysis.

[10] SolarWinds incident statistics from Arnold & Porter analysis and CISA reporting.

[11] U.S. Government Accountability Office (GAO), "SolarWinds Cyberattack Demands Significant Federal and Private-Sector Response," April 2021. GAO-21-501.

[12] Microsoft Azure Status blog post on CrowdStrike incident impact assessment, July 2024.

[13] UK Financial Conduct Authority, "CrowdStrike Outage: Lessons for Operational Resilience," October 31, 2024. Available at: https://www.fca.org.uk/firms/operational-resilience/crowdstrike-outage-lessons-operational-resilience

[14] CrowdStrike incident cost estimates from: "How the 2024 CrowdStrike Outage Revealed Glaring Gaps in Risk and Incident Management," OneClickComply. Available at: https://oneclickcomply.com/blog/how-the-2024-crowdstrike-outage-revealed-glaring-gaps-in-risk-and-incident-management

[15] Kaseya VSA ransomware attack (July 2, 2021): REvil ransomware gang exploited vulnerabilities in Kaseya's VSA remote monitoring software used by managed service providers (MSPs). Approximately 50-60 MSPs were directly compromised, cascading to 800-1,500 downstream businesses. The attack demonstrated how MSP relationships create attack force multiplication invisible to traditional TPRM—a single MSP compromise affecting 15-25+ downstream organizations. Notable impact: Coop Sweden's 800 grocery stores were closed when their POS system vendor's MSP was compromised. Source: CISA, "Kaseya VSA Supply-Chain Ransomware Attack," July 2021; Huntress Labs incident analysis.

[16] Emsisoft, "Unpacking the MOVEit Breach: Statistics and Analysis," updated December 2023. Available at: https://www.emsisoft.com/en/blog/44123/unpacking-the-moveit-breach-statistics-and-analysis/

[17] 3CX supply chain attack (March 2023): First documented "supply chain of a supply chain" attack. The 3CX desktop application was compromised through a prior compromise of X_TRADER trading software used by a 3CX employee, creating a five-level dependency chain. Coalition Inc. concluded that "complete supply chain visibility is practically impossible" given such attack complexity. Source: SentinelOne, "SmoothOperator | Ongoing Campaign Trojanizes 3CXDesktopApp in Supply Chain Attack," March 2023. Available at: https://www.sentinelone.com/blog/smoothoperator-ongoing-campaign-trojanizes-3cxdesktopapp-in-supply-chain-attack/

[18] UK FCA operational resilience data (2022-2023) cited in CrowdStrike lessons learned publication, October 2024.

[19] SecurityScorecard, "The State of Third-Party Risk Management," 2024. Available at: https://securityscorecard.c
party-risk-management/

[20] SecurityScorecard and The Cyentia Institute, "Close Encounters of the Third (and Fourth) Party Kind" (2024). The report analyzed security incidents affecting financial services organizations and found that 97% of major US banks experienced exposure through third- or fourth-party relationships in 2024, despite the actual number of directly compromised vendors being relatively small. This demonstrates the multiplicative effect of shared dependencies. Note: This figure represents exposure through vendor relationships (having a connection to a breached vendor's network), not necessarily confirmed data compromise. Vendor security rating methodologies vary.

[21] OCC Bulletin 2013-29: "Third-Party Relationships: Risk Management Guidance," October 30, 2013. Available at: https://lenderscompliancegroup.com/wp-content/uploads/2020/02/OCCBulletin2013-292CThirdPartyRelationships28RiskManagement29.pdf

[22] Basel Committee on Banking Supervision, "Principles for Operational Resilience and Revised PSMOR," March 2021. BIS Newsletter on Third and Fourth-Party Risk. Available at: https://www.bis.org/publ/bcbs_nl28.htm

[23] OCC Bulletin 2023-17 / Federal Reserve SR 23-4 / FDIC FIL-23-2023: "Interagency Guidance on Third-Party Relationships: Risk Management," June 6, 2023. Available at: https://www.occ.gov/news-issuances/bulletins/2023/bulletin-2023-17.html

[24] EU Digital Operational Resilience Act (DORA), Regulation (EU) 2022/2554, adopted December 14, 2022, application date January 17, 2025. Available at: https://www.eiopa.europa.eu/digital-operational-resilience-act-dora_en

[25] UK Financial Conduct Authority, "CrowdStrike Outage: Lessons for Operational Resilience," October 31, 2024.

[26] AICPA, "2017 Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy" (with Revised Points of Focus – 2022). Core criteria unchanged since 2017; points of focus updated September 2022. Available at: https://www.aicpa-cima.com/resources/download/2017-trust-services-criteria-with-revised-points-of-focus-2022

[27] Secureframe, "The History of SOC 2," 2023. Available at: https://secureframe.com/hub/soc-2/history. See also: PKF AvantEdge, "A Brief History of SOC and SAS."

Part I: The Historical Context

Chapter 1: From Vendor Management to TPRM

The Semantic Misinterpretation That Created a Structural Gap

The 1990s: Simple Relationships

In the early 1990s, vendor management was an administrative function. Banks maintained most critical systems in-house, and third-party relationships followed a straightforward pattern: one vendor provided one product or service. A payroll processor handled payroll. A benefits administrator managed employee benefits. A software vendor sold a license for on-premise deployment.[1]

The risk profile was equally straightforward. Vendor management focused on contractual compliance, service delivery, and pricing. The function typically reported to procurement or operations. Risk, in the sense we understand it today, was not the primary lens through which these relationships were evaluated.

This simplicity was a function of the technology landscape. Outsourcing was a relatively new practice.[2] Most organizations retained data processing and IT operations internally. When they did engage third parties, the relationships were discrete and bounded. If a vendor failed, the impact was localized to that specific function. Dependencies rarely extended beyond the direct contractual relationship.

The Shift to "Third-Party Risk Management"

As outsourcing accelerated through the 1990s and into the 2000s, the terminology began to change. "Vendor management" became "third-party risk management." This shift reflected a fundamental evolution in how organizations viewed these relationships. Vendors were no longer simply suppliers to be managed—they were sources of risk requiring assessment, monitoring, and oversight.

Regulatory attention followed. The FFIEC issued guidance on technology outsourcing in November 2000.[3] The OCC published Advisory Letter 2000-9 introducing "third-party risk" terminology.[4] By November 2001, with OCC Bulletin 2001-47, the foundational principles of third-party risk management were established: banks must develop risk management plans before contracting with third parties, conduct due diligence, structure contracts appropriately, and maintain ongoing oversight.[5]

The industry embraced the new paradigm. Risk management functions began taking ownership of vendor relationships. Assessment questionnaires proliferated. Due diligence processes formalized. The recognition that third parties created operational, compliance, and information security risks became standard operating procedure.

The Misinterpretation

Here is where the critical misinterpretation occurred—subtle, understandable, and consequential.

The term "Third-Party Risk Management" was intended to mean: **managing all risks associated with third-party relationships, including the dependencies those third parties rely upon**.

When a bank engages a vendor, it assumes not just the direct risks that vendor creates, but also the inherited risks from that vendor's own third-party dependencies. The vendor's cloud provider, their software libraries, their security tools, their subcontractors—all become part of the risk profile the bank must manage.

However, the industry adopted a weaker interpretation: **managing the risks that directly come from our third party**. This framing positioned the third party as the risk boundary. TPRM programs focused on assessing the vendor's controls, reviewing their SOC 2 reports, evaluating their security posture, and monitoring their performance. But the dependencies that vendor relied upon—their "fourth parties"—remained largely invisible.

This was not unreasonable. In the early 2000s, those fourth-party dependencies were fewer and more manageable. A vendor's critical dependencies typically consisted of their data center or colocation provider. Infrastructure-level visibility was sufficient. Application-level dependencies—the software libraries embedded in their code, the third-party components integrated into their systems—were not yet recognized as material risk drivers.

Cloud Computing Forces Recognition of Fourth Parties

The emergence of cloud computing in the late 2000s and early 2010s made fourth-party risk impossible to ignore.[6] As vendors migrated from on-premise data centers to cloud infrastructure providers like AWS, Azure, and Google Cloud, financial institutions began to realize that their apparent vendor diversification masked underlying concentration.

An institution might have 100 different SaaS vendors, creating the appearance of a diversified vendor portfolio. But if 80 of those vendors run on AWS, the institution has significant concentration risk on a single infrastructure provider with whom they have no direct contractual relationship. This was the fourth-party problem made visible.

Regulatory guidance evolved to address this. OCC Bulletin 2013-29 introduced the concept of "critical activities" requiring heightened oversight.[7] The 2013 Federal Reserve guidance (SR 13-19) explicitly identified concentration risk as an area of concern.[8] The framework expanded: institutions were now expected to identify their vendors' critical fourth parties, particularly cloud infrastructure providers.

But even this evolution remained bounded. TPRM programs built processes to identify "critical fourth parties"—which in practice meant asking vendors: "What cloud provider do you use? Do you have a colocation provider?" The focus remained at the infrastructure layer. The inventory was still organized around **parties** (legal entities), not **dependencies** (technical components).

The Current State: Built for n=4 at Best

Today's TPRM programs are optimized for managing third-party relationships with visibility into critical fourth parties. This represents significant maturation from the procurement-focused vendor management of the 1990s. Institutions maintain vendor inventories, conduct risk assessments, require SOC 2 attestations, monitor for incidents, and track critical dependencies like cloud providers.

Yet the framework stops there. Fifth-party visibility—the subcontractors and service providers that a vendor's vendors rely upon—is rarely collected or analyzed.[9] The data is not readily available, and even when institutions request it, the information is often incomplete or quickly becomes stale.

More fundamentally, this entire paradigm is organized around **legal entities**. Third party = the vendor (a company). Fourth party = the vendor's vendor (another company). Fifth party = their vendor (yet another company). The framework assumes risk can be managed by understanding the chain of contractual relationships between organizations.

Why This Matters: Application-Layer Dependencies Remain Invisible

This legal entity focus creates a profound blind spot: application-layer dependencies.

Modern SaaS applications are built on foundations of external code libraries—components embedded in the application code itself, not the infrastructure layer where cloud providers operate. These components have no Legal Entity Identifiers (LEIs) and are not "vendors" in the traditional sense. Many are open-source projects maintained by volunteers; others are commercial libraries from companies with whom the financial institution has no direct relationship.[10] (Chapter 4 examines the full taxonomy of these invisible dependencies and their risk implications.)

When the Log4j vulnerability was discovered in December 2021 (detailed in Chapter 7),[11] financial institutions faced a simple but devastating question: "Which of our vendors use Log4j?" Most could not answer. The dependency existed deep within transitive dependency chains, did not appear on any vendor assessment questionnaire, and was not listed as a "critical fourth party." Traditional TPRM frameworks had no mechanism to discover it, inventory it, or monitor it. The same pattern repeated in 2024 with the xz Utils backdoor attempt,[12] where a widely-used compression library nearly became a vector for supply chain compromise—again, a dependency invisible to traditional vendor management processes.

This is the core problem: **TPRM programs are built to manage parties, but modern technology risk propagates through dependencies.**

The Original Intent vs. Weak Interpretation

Returning to the original terminology: "Third-Party **Risk** Management." The emphasis should have been on **risk**—all risks emanating from the third-party relationship, regardless of how many levels deep the dependency chain extends.

If a bank's payment processing vendor relies on a specific logging library, and that library has a critical vulnerability, the bank faces risk. The fact that the library is not a "party" in the contractual sense is irrelevant. The risk exists. It can be exploited. It can cause material harm.

The weaker interpretation—managing only the direct risks from our third party—was never the intent. But it became the practice. And as technology dependencies have grown exponentially more complex, the gap between intent and practice has widened into a fundamental gap.

Implications

This chapter has traced the evolution from administrative vendor management to risk-based TPRM, highlighting the critical semantic misinterpretation that shaped how frameworks developed. The shift to "third-party risk management" should have meant comprehensive management of all dependency risks. Instead, it was interpreted as managing risks from direct vendor relationships, with limited visibility into fourth parties and virtually no visibility beyond.

The result is a set of programs built for a simpler technological era—one where vendors provided bounded services, dependencies were few and visible, and infrastructure-level oversight was suffi-

cient. Modern technology operates differently. Applications are composed of hundreds of external components.[13] Vendors rely on dozens of service providers. Infrastructure converges on a handful of cloud platforms. The dependency chains extend 10, 20, 50 levels deep.

Until frameworks close this gap, incidents like Log4j will continue to expose the fundamental mismatch between entity-based oversight and dependency-level risk.

The answer lies in an institutional accident: how the accounting profession became the steward of technology risk assurance when no technology-focused organization stepped forward.

Endnotes - Chapter 1

[1] Federal Financial Institutions Examination Council (FFIEC), "Outsourcing Technology Services" IT Examination Handbook, 2000 (noting that outsourcing of technology services was "relatively new" in the early 1990s and describing the evolution from administrative vendor management to risk-focused TPRM). The OCC's 1996 Banking Circular 196 and 1997 updates marked the beginning of formal regulatory expectations for vendor management in financial services.

[2] Federal Financial Institutions Examination Council (FFIEC), "FFIEC IT Examination Handbook: Outsourcing Technology Services," June 2004, https://ithandbook.ffiec.gov/it-booklets/outsourcing-technology-services (noting that in the early 1990s, outsourcing was a relatively new practice and most organizations retained data processing and IT operations internally)

[3] Federal Financial Institutions Examination Council (FFIEC), "Risk Management of Outsourced Technology Services," November 28, 2000 (rescinded by 2004 FFIEC Outsourcing Technology Services Booklet)

[4] Office of the Comptroller of the Currency (OCC), "Advisory Letter 2000-9: Third-Party Risk," 2000 (rescinded by OCC Bulletin 2013-29)

[5] Office of the Comptroller of the Currency (OCC), "Bulletin 2001-47: Third-Party Relationships: Risk Management Principles," November 2001 (rescinded by OCC Bulletin 2013-29), https://corpgov.law.harvard.edu/2013/12/01/occ-updates-guidance-on-third-party-risk-management/

[6] Amazon Web Services launched in 2006 with S3 and EC2. Microsoft Azure followed in 2010, and Google Cloud Platform in 2012. By 2015, enterprise cloud adoption had reached mainstream status. See: Gartner, "Cloud Computing Timeline and Market Evolution," and IDC, "Worldwide Public Cloud Services Market Share" reports, 2010-2015.

[7] Office of the Comptroller of the Currency (OCC), "Bulletin 2013-29: Third-Party Relationships: Risk Management Guidance," October 30, 2013 (rescinded by OCC Bulletin 2023-17), https://sharedassessments.org/blog/occ-releases-guidance-third-party-relationships-occ-2013-29/

[8] Board of Governors of the Federal Reserve System, "SR 13-19: Guidance on Managing Outsourcing Risk," December 5, 2013, https://securityscorecard.com/blog/sr-13-19-provides-guidance-service-provider-risk-management

[9] Industry surveys consistently show limited visibility beyond fourth parties. The Shared Assessments Program's 2023 Third Party Risk Management Survey found that only 23% of organizations had formal processes for assessing risks beyond their direct vendors' subcontractors. KPMG's 2024 Third Party Risk Management Outlook similarly noted that "nth-party risk remains a significant blind spot" for most financial institutions.

[10] The Linux Foundation and Laboratory for Innovation Science at Harvard's 2020 "Census II of Free and Open Source Software" found that 60% of the most critical open source components are maintained by small teams of unpaid volunteers. The Tidelift "2023 Open Source Maintainer Survey" reported that 60% of maintainers are unpaid and 46% have considered quitting due to burnout.

[11] Cybersecurity & Infrastructure Security Agency (CISA), "CSRB Report on Log4j," July 2022, https://www.cisa.gov/sites/default/files/publications/CSRB-Report-on-Log4-July-11-2022_508.pdf (noting that the Log4Shell vulnerability CVE-2021-44228 was publicly disclosed in December 2021)

[12] The xz Utils backdoor incident (CVE-2024-3094, disclosed March 2024) involved a sophisticated supply chain attack where a malicious actor gained maintainer access to the xz Utils compression library and inserted a backdoor targeting SSH authentication. The backdoor was discovered by a Microsoft engineer before widespread deployment, but the incident demonstrated the vulnerability of widely-used open source dependencies to social engineering and maintainer compromise. See: Ars Technica, "What we know about the xz Utils backdoor that almost infected the world," April 2024; CISA Alert, "Reported Supply Chain Compromise Affecting XZ Utils Data Compression Library," March 29, 2024.

[13] Software composition analysis industry research indicates modern applications average 100-500 direct dependencies, with transitive dependencies often reaching into the thousands. See: Synopsys Open Source Security and Risk Analysis (OSSRA) Report 2024; Sonatype State of the Software Supply Chain 2024.

Chapter 2: The Accountants Stepped In

How Financial Auditors Became the Stewards of Technology Risk Assurance

SAS 70's Original Purpose

In April 1992, the American Institute of Certified Public Accountants (AICPA) introduced Statement on Auditing Standards No. 70 (SAS 70).[1] Its purpose was narrow and specific: to provide guidance for CPAs reporting on controls at service organizations that affected their clients' financial statements.

The context matters. Organizations were beginning to outsource financial transaction processing—payroll, benefits administration, transaction processing—to specialized service providers (see Chapter 1 for the broader history of outsourcing's emergence in the 1990s). External auditors needed a way to assess the controls at these third-party organizations without having to audit each one individually for every client that used them. SAS 70 solved this problem. A service organization could engage a CPA firm to issue a report on its internal controls over financial reporting. User auditors (the auditors of companies using the service) could then rely on that report rather than conducting their own assessment.

This was an auditor-to-auditor communication mechanism. SAS 70 reports were designed for a technical audience of accounting professionals. They addressed a specific concern: do the controls at this service organization provide reasonable assurance that financial data processed on behalf of our client is accurate and complete?

The standard served its intended purpose well—for about a decade.

The Market Misuse Problem

As outsourcing expanded through the 1990s and into the 2000s, companies began outsourcing not just financial processing but core business functions: IT infrastructure, security services, cloud hosting, managed services, customer data processing.[2] The scope and complexity of outsourced relationships grew exponentially.

When companies engaged these new categories of service providers, they naturally asked: "How do we know you have adequate controls?" The service providers, seeking a credible answer, pointed to the only standardized framework available: SAS 70. "We have a SAS 70 report," became the response.

This was, technically, a misuse. SAS 70 was designed for controls over financial reporting, not information security, not data privacy, not operational resilience.[3] But the market had no alternative. There was no equivalent framework for technology controls. So SAS 70 became the catch-all assurance mechanism, regardless of whether financial reporting was even relevant to the service being provided.

The AICPA observed this trend with growing concern. Chuck Landes, vice president of professional standards and services for the AICPA, noted that vendors were claiming "SAS 70 certification" when no such certification existed.[4] Marketing materials implied that a SAS 70 report guaranteed security or compliance, when the report merely confirmed that stated controls were being followed. There was no minimum standard, no benchmark for adequacy—only verification that the organization did what it said it did.[5]

Auditors found themselves complicit. Firms began preparing SAS 70 reports for organizations whose services had nothing to do with financial statement preparation—HR software, communications tools, cloud hosting. The scope creep was undeniable, but the demand was real.[6]

The AICPA's Response: SSAE 16 and the SOC Framework

In 2011, the AICPA made a decisive move. Rather than fight the market misuse of SAS 70, they formalized it. They introduced Statement on Standards for Attestation Engagements No. 16 (SSAE 16) and created the Service Organization Control (SOC) framework, effective June 15, 2011.[7]

The new framework separated financial reporting controls from broader technology and security controls through a three-tier structure:

SOC 1: The successor to SAS 70, focused exclusively on controls relevant to financial reporting. Designed for user auditors and restricted in distribution—cannot be used as a marketing tool.[8]

SOC 2: The innovation. Designed specifically to address controls relevant to security, availability, processing integrity, confidentiality, and privacy. This was the framework for technology risk. Unlike SOC 1, SOC 2 reports could be shared with management, regulators, and business partners (under non-disclosure agreements).

SOC 3: A public-facing summary report. The only SOC report explicitly designed for marketing purposes.[8] Provides a high-level seal indicating that an organization has been examined against the Trust Services Criteria, without disclosing detailed control descriptions.

To provide substantive criteria for SOC 2 examinations, the AICPA developed the Trust Services Principles[9] (later rebranded as Trust Services Criteria in 2017).[10] These defined the five categories: security, availability, processing integrity, confidentiality, and privacy. For each category, the criteria specified what controls organizations should have in place.

Two Perspectives on the AICPA's Decision

The creation of SOC 2 can be viewed in two ways, both of which contain truth.

Perspective 1: Productizing the Misuse

The AICPA recognized that companies were using SAS 70 for technology controls despite it being inappropriate. Rather than correcting this behavior, they monetized it by creating SOC 2. Accounting firms, which had built lucrative practices around SAS 70 audits, now had an even larger market: every technology service provider needed SOC 2 attestation. The misuse became the product.

Perspective 2: Addressing an Unmet Need

The market genuinely needed a standardized framework for technology control assurance. Companies were demanding "show us your controls" from vendors. Procurement processes required some form of independent validation. The AICPA, seeing this need and possessing the institutional infrastructure to address it, stepped forward when no one else did.

Both perspectives have merit. The AICPA did see a market opportunity created by the misuse of SAS 70. But they also filled a genuine void. Why did the accounting profession fill this void rather than technology-focused organizations?

Why Accounting Firms Stuck

The dominance of accounting firms in technology risk attestation is not accidental. They possess several structural advantages that technology consulting firms, cybersecurity specialists, or information security professionals cannot easily replicate.

1. Existing Audit Infrastructure

Accounting firms already had deep relationships with the companies that would need SOC reports. CFOs, controllers, and audit committees were existing touchpoints. Firms like Deloitte, PwC, EY, and KPMG had global presence, enabling them to serve multi-national organizations.[11] They had established audit methodologies for evaluating internal controls—frameworks that could be adapted from financial controls to technology controls.[12]

When SOC 2 emerged, accounting firms did not need to build client relationships from scratch. They were already in the building.

2. Professional Standards and Independence

A pillar of AICPA standards is independence. CPAs conducting attestation engagements must be independent of the entity they are auditing.[13] This is not merely a professional guideline—it is enforced through state licensing boards and peer review processes. A CPA firm that violates independence standards faces professional consequences, including potential license revocation.

Technology consulting firms often provide implementation services to the same clients they might assess. This creates inherent conflicts of interest. A firm that helps design a security architecture and then attests to its adequacy has compromised independence. The AICPA has clear guidance on this: providing certain non-attest services—including penetration testing, vulnerability management, and incident response—to an audit client creates independence threats that must be carefully managed or avoided entirely.[14]

Independence is not just a technical requirement; it is a market signal. When a company tells a prospective customer "we have a SOC 2 report," the credibility of that report depends on the independence of the auditor. "Audited by Deloitte" carries weight in procurement decisions in a way that "assessed by XYZ Consulting" does not, precisely because the market understands that Deloitte has professional obligations and oversight mechanisms that a consulting firm may lack.

3. Regulatory Authority

Here is the critical structural advantage: the AICPA created and codified the SOC standards. SSAE 18 (the current attestation standard) and the Trust Services Criteria were developed by the AICPA.[15] The standards explicitly state that only licensed CPA firms may perform SOC 1 and SOC 2 audits.[16]

This is a regulatory moat. It is not merely that CPA firms are well-positioned to perform these audits—they are the **only** organizations authorized to do so. A technology consulting firm, no matter how technically proficient, cannot issue a SOC 2 report. The AICPA controls the standard, and compliance with the standard requires using a licensed CPA firm.

This was not a nefarious power grab. The AICPA had the authority to create attestation standards because it is the professional body governing the accounting profession. When they developed SOC 2 to address market demand, they naturally embedded it within their existing professional standards framework. The result, however, is that technology risk attestation became structurally dependent on the accounting profession.

4. Risk Management Expertise Transfer

From a non-technical perspective, CPAs are experts in risk management. Auditing is fundamentally about evaluating risk and controls. The transition from auditing financial risk to auditing cybersecurity risk was conceptually straightforward.[17] CPAs understand internal control concepts, evidence requirements, risk-based audit approaches, and control framework evaluation. These skills transfer across domains.

Of course, this does not mean CPAs are cybersecurity experts. Most are not. But CPA firms addressed this by hiring technology specialists—professionals with IT, security, and engineering backgrounds—and integrating them into audit teams. The firms leveraged their institutional infrastructure (methodology, quality control, client relationships, independence frameworks) and supplemented it with technical expertise.

5. "Audited by Big 4" Market Weight

The prestige and market recognition of being audited by a Big 4 firm carries significant weight.[11] This creates a self-reinforcing cycle:

- Companies seeking vendors prefer those with Big 4 SOC 2 reports
- Service organizations seek Big 4 auditors to satisfy customer demands
- Big 4 firms gain more experience and market share
- The cycle continues

Smaller CPA firms and regional practices also perform SOC 2 audits, and many do so with equal rigor. But in high-stakes procurement decisions, the brand recognition of a Big 4 auditor can be the deciding factor.

The Counterfactual: What If Technology Firms Had Led?

It is worth considering why technology consulting firms, cybersecurity specialists, or professional associations like ISACA did not create competing frameworks.

Lack of Independence Standards: Technology consultants often implement the very systems they might assess, creating conflicts of interest with no professional framework to manage them.

No Unified Professional Body: Unlike the AICPA, which represents all CPAs under consistent standards, the technology industry lacks a single authoritative body with the power to create and enforce attestation standards.

Implementation vs. Attestation: Technology firms excel at building and securing systems but lack the attestation tradition and methodology that accounting firms have developed over decades. Attestation is a distinct competency—not simply technical assessment, but independent verification according to established criteria with defined reporting formats and professional liability frameworks.

Client Skepticism: Would CFOs and audit committees trust a technology vendor to objectively assess another technology vendor? The perception of independence matters as much as actual independence. Accounting firms benefit from a century of credibility in independent verification.

The Answer to "Why Accountants?"

The AICPA and accounting firms ended up governing technology risk assurance because:

1. They were already there (existing client relationships)
2. They had the infrastructure (audit methodology, quality control, peer review)
3. They had independence (professional requirements and enforcement mechanisms)
4. They created the standards (AICPA codified the rules, creating a regulatory moat)
5. They had market credibility ("Audited by [Big 4]" carries weight)
6. No one else stepped up (no competing professional body created a credible alternative)

This was not nefarious. It was institutional advantage meeting market need.

Implications

The AICPA deserves credit for recognizing a market gap and filling it with a structured framework when chaos threatened to persist. SOC 2 is far superior to the pre-2011 environment of misapplied SAS 70 reports and meaningless certification claims.

However, this history reveals an important dynamic: technology risk attestation is governed by a profession whose core expertise is financial auditing, not technology. The Trust Services Criteria are principle-based and abstract because that is how accountants approach standards—they favor broad applicability over technical prescriptiveness.

This matters because the governance structure affects what evolves and what does not. An accounting body is not naturally positioned to drive innovation in software supply chain attestation, CI/CD pipeline security, or application-layer dependency transparency. These are not areas where the profession has deep native expertise.

CPAs can competently assess technology risks, especially when supported by technical specialists. What's less clear is whether the institutional incentives and expertise of the accounting profession will naturally drive the evolution needed to address gaps that are becoming increasingly visible: software bill of materials disclosure, software delivery mechanism attestation, application-layer dependency transparency.

So far, the answer appears to be no. SOC 2's Trust Services Criteria have remained largely unchanged since 2017, with only revised points of focus in 2022 that explicitly did not alter the core criteria.[18] The framework has not required SBOMs, has not mandated disclosure of CI/CD pipeline security controls, has not addressed open source dependency transparency. These gaps persist not because of negligence, but because the governance structure does not naturally incentivize their closure.

The accountants stepped in when no one else did. That was the right thing to do in 2011. The question now is whether the same governance structure will prove sufficient for the challenges ahead. But the accountants were not the only profession positioned to create technology risk frameworks. Several technology-focused organizations had the expertise and credibility—yet none built competing alternatives.

Endnotes - Chapter 2

[1] AICPA, "Statement on Auditing Standards No. 70: Service Organizations," issued April 1992. Historical context from PKF AvantEdge, "A Brief History of SOC and SAS," and AICPA professional standards documentation. See also: Journal of Accountancy, "Replacing SAS 70," August 2010, https://www.journalofaccountancy.com/issues/2010/aug/20103009/

[2] The evolution of outsourcing scope is documented in FFIEC IT Examination Handbooks across editions (2000, 2004, 2013). The 2004 edition explicitly noted expansion beyond "data processing and related technology services" to include "networking, Web hosting, software development and maintenance, and other technology-based products and services." See also: Deloitte Global Outsourcing Survey series (2000-2012) documenting the shift from cost-driven tactical outsourcing to strategic technology partnerships.

[3] CFO.com, "The Truth About SAS 70," September 2010, https://www.cfo.com/news/the-truth-about-sas-70/669107/

[4] CFO.com, "The Truth About SAS 70," September 2010, https://www.cfo.com/technology/2010/09/the-truth-about-sas-70/ (Chuck Landes quotes and auditor complicity discussion)

[5] Vibato Blog, "SAS 70 to SSAE 16: How to Review your Vendors Internal Control Report," https://www.vibato.com/blog/bid/79403/SAS-70-to-SSAE-16-How-to-Review-your-Vendors-Internal-Control-Report

[6] CFO.com, "The Truth About SAS 70," September 2010, https://www.cfo.com/technology/2010/09/the-truth-about-sas-70/ (auditor complicity discussion)

[7] Vibato Blog, "SAS 70 to SSAE 16: How to Review your Vendors Internal Control Report," https://www.vibato.com/blog/bid/79403/SAS-70-to-SSAE-16-How-to-Review-your-Vendors-Internal-Control-Report (SSAE 16 effective date: June 15, 2011)

[8] PKF AvantEdge, "A Brief History of SOC and SAS," https://www.pkfavantedge.com/it-compliance/a-brief-history-of-soc-and-sas/

[9] Compass IT Compliance, "A Detailed History of SOC 2 Compliance," https://www.compassitc.com/blog/a-detailed-history-of-soc-2-compliance

[10] Secureframe, "2025 Trust Services Criteria for SOC 2," https://secureframe.com/hub/soc-2/trust-services-criteria (2017 Trust Services Criteria rebrand from Trust Services Principles)

[11] Schellman, "Which Big 4 Auditing Firm Should Perform Your SOC Audit?" https://www.schellman.com/blog/so examinations/which-audit-big-4-should-perform-your-soc-audit

[12] Linford & Co., "Who Can Perform a SOC Audit? CPA, non-CPA, SOC 1 & SOC 2," https://linfordco.com/blog/who-can-perform-soc-audit/ (CPA audit methodology and training)

[13] Linford & Co., "Who Can Perform a SOC Audit? CPA, non-CPA, SOC 1 & SOC 2," https://linfordco.com/blog/who-can-perform-soc-audit/ (Independence requirements)

[14] Sprinto Blog, "SOC 2 Myths and Malpractices Busted: Be Wary Of These Red Flags," https://sprinto.com/blog/soc-2-myths/ (Independence threats from nonattest services including penetration testing, vulnerability management, incident response)

[15] Linford & Co., "Who Can Perform a SOC Audit? CPA, non-CPA, SOC 1 & SOC 2," https://linfordco.com/blog/who-can-perform-soc-audit/ (AICPA codification of SOC standards)

[16] Secureframe, "Who Performs a SOC 2 Audit?" https://secureframe.com/hub/soc-2/who-performs-a-soc-2-audit (Only licensed CPA firms can perform SOC audits)

[17] Schneider Downs, "Why Do CPA Firms Perform SOC 2 Audits?" https://schneiderdowns.com/our-thoughts-on/why-cpa-firms-perform-soc-2-audits/ (Natural progression from financial risk to cybersecurity risk)

[18] AICPA, "2017 Trust Services Criteria (With Revised Points of Focus – 2022)," https://www.aicpa-cima.com/resources/download/2017-trust-services-criteria-with-revised-points-of-focus-2022

Chapter 3: Who Should Have Followed (But Didn't)

The Technology-Focused Organizations That Could Have Created Alternatives

The Vacuum

By the mid-2000s, the limitations of SAS 70 were evident. It was being used for purposes it was never designed to address. The market needed a technology-specific attestation framework—one designed by and for technology professionals, with criteria that addressed modern software development practices, security architectures, and operational resilience.

This need was recognized. Companies were asking vendors for assurance. Procurement teams were demanding independent validation. Risk managers needed some basis for evaluating third-party technology controls. The demand was real and growing.

The AICPA filled this vacuum with SOC 2 in 2011. But that should not have been the only option. Several technology-focused organizations had the expertise, credibility, and market presence to create competing frameworks. None did. Understanding why reveals structural constraints that persist to this day.

ISACA: The Professional Association That Could Not Compete

ISACA (formerly the Information Systems Audit and Control Association), established in 1969,[1] is the most prominent professional association dedicated to managing and auditing information systems. It is best known for the COBIT framework[2] and for administering certifications including CISA (Certified Information Systems Auditor), CRISC (Certified in Risk and Information Systems Control), and CISM (Certified Information Security Manager).[3]

If any organization had the expertise to create a technology-specific attestation framework, ISACA did. COBIT provides comprehensive IT governance and management guidance. ISACA's certifications are widely recognized in the information security and audit community. The organization publishes standards, guidelines, and best practices specifically for IT audit and assurance.[4]

Yet ISACA did not create an attestation framework to compete with SOC 2. Why?

Structural Mission Constraint: ISACA operates as a professional association and standards development organization, not as an attestation service provider. This creates a fundamental separation:

- **What ISACA does**: Certify individuals (CISA, CRISC, CISM), develop frameworks and standards (COBIT), publish audit guidance
- **What ISACA does not do**: Perform organizational attestations, certify companies, issue audit reports

This is not accidental. ISACA's mission is to support the profession, not compete with it. If ISACA created an attestation service, it would be competing with the audit firms and internal audit departments whose professionals hold ISACA certifications. This creates an insurmountable conflict of interest.

ISACA publishes standards that **others** use to perform attestations.[5] Individual auditors may hold ISACA certifications, but the attestation is performed by their firm (a CPA firm or internal audit department), not by ISACA. The organization certifies practitioners, not companies.

The Business Model Problem: Professional associations derive revenue from certifications, memberships, training, and conferences. Adding attestation services would require building an entirely different business—one that competes with members. The economics do not work.

The Independence Question: Even if ISACA wanted to provide attestation, the independence concerns would be significant. If ISACA develops the standard (COBIT), trains professionals in that standard, and then attests that companies comply with the standard, where is the independence? The AICPA model works because different CPA firms compete to perform SOC audits—the standard-setter does not perform the attestations. ISACA would be both.

What ISACA Actually Contributed: ISACA's frameworks and guidance are used by auditors performing SOC 2 assessments. The Trust Services Criteria align with COBIT principles. Many SOC 2 auditors hold CISA certifications. ISACA's contribution is indirect but real—just not in the form of a competing attestation framework.

NIST: The Government Agency Barrier

The National Institute of Standards and Technology (NIST) is a U.S. federal government agency with a statutory mission to develop standards and guidance.[6] NIST has produced extensive frameworks relevant to technology risk, including the Cybersecurity Framework (CSF),[7] SP 800-53 (Security and Privacy Controls),[8] SP 800-171 (Protecting Controlled Unclassified Information),[9] and SP 800-218 (Secure Software Development Framework).[10]

These frameworks are widely respected. Federal agencies rely on them. Many financial institutions use NIST CSF as a foundational element of their cybersecurity programs. The frameworks are technically robust, comprehensive, and continuously updated.

Yet NIST does not offer certification or attestation. There is no "NIST CSF certification" issued by NIST.[11] Why?

Explicit Policy of Non-Certification: NIST intentionally maintains separation between standards development and compliance verification. The agency develops frameworks that **others** use for attestation. This prevents conflicts of interest and maintains objectivity in the standards development process.

NIST's website explicitly states that the Cybersecurity Framework is voluntary and does not establish a certification program.[12] Organizations may self-assess, may be assessed by third parties, or may face regulatory requirements based on NIST standards—but NIST itself does not certify compliance.

Government Agency Mission Constraint: As a federal agency, NIST's role is to develop standards, conduct research, and provide guidance. Creating a certification or attestation business would fundamentally change its mission and raise questions about government involvement in commercial markets. Would a NIST certification give certified companies a competitive advantage? Would foreign entities trust a U.S. government certification? These questions illustrate why government agencies typically do not perform attestation.

Self-Attestation Model: For federal software procurement, NIST frameworks are used via self-attestation or third-party assessment organizations (3PAOs). Executive Order 14028 (2021)[13] required software producers selling to the federal government to attest compliance with NIST Secure Software Development Framework (SSDF). But the attestation is provided by the software producer or an independent assessor—not by NIST.[14]

Third-Party Ecosystem: NIST's approach has enabled an ecosystem of third-party assessors. FedRAMP uses NIST standards but has its own authorization process.[15] CMMC (Cybersecurity Maturity Model Certification) for Department of Defense contractors requires third-party assessments against NIST 800-171.[16] Private organizations have created certification programs based on NIST standards (such as the SCF Conformity Assessment Program).[17] NIST provides the criteria; others provide the attestation.

What NIST Actually Contributed: NIST frameworks serve as the foundation for many attestation programs, including FedRAMP and CMMC. They are referenced in SOC 2 assessments and incorporated into risk management programs globally. NIST's contribution is profound—just not in the form of a NIST-issued attestation.

ISO/IEC: Organizational Scope, Not Product-Specific

ISO/IEC 27001 is an internationally recognized standard for information security management systems (ISMS).[18] It provides a framework for managing and protecting sensitive information and is widely adopted globally. Organizations can become ISO 27001 certified through independent certification bodies.

This sounds like exactly what the market needed—a technology-specific certification alternative to SOC 2. Yet ISO 27001 did not fill the gap. Why?

Organizational vs. Product-Level Scope: ISO 27001 certifies an organization's security management system, not specific technology products or services.[19] The scope is enterprise-wide governance and process maturity, not the security controls of a particular application, system, or service offering.

A SaaS vendor might achieve ISO 27001 certification for their overall ISMS, but that certification does not attest to the specific security controls of their software product. It confirms they have a structured approach to information security management—policies, risk assessments, training, incident response processes. It does not confirm that their application is free of Log4j vulnerabilities, that their CI/CD pipeline is secured, or that they generate SBOMs.

Broad vs. Technical Focus: ISO 27001 addresses general information security management. It is process-oriented and technology-agnostic. While this broadness enables global applicability, it also means the standard does not address technology-specific risks like software supply chain dependencies, cloud architecture patterns, or infrastructure concentration.

Complementary, Not Competing: ISO 27001 and SOC 2 serve different purposes. ISO 27001 provides organizational-level assurance. SOC 2 provides service-level assurance based on specific Trust Services Criteria. Many organizations pursue both: ISO 27001 to demonstrate enterprise security maturity, and SOC 2 to provide detailed control attestation for their service offerings.

Cloud-Specific Extensions (ISO 27017/27018): ISO developed cloud-specific guidance with ISO 27017 (cloud services information security controls) and ISO 27018 (protection of PII in cloud environments).[20] However, these remain organizational certifications, not product or service-level attestations. They extend ISO 27001 to cloud contexts but do not fundamentally change the scope.

What ISO Actually Contributed: ISO 27001 provides a globally recognized standard for ISMS certification. It is referenced in procurement requirements, integrated into risk management programs, and recognized by regulators. But it addresses a different assurance need than SOC 2—organizational governance rather than service-specific control validation.

Cloud Security Alliance: Enhancement, Not Alternative

The Cloud Security Alliance (CSA), founded in 2008,[21] is a nonprofit organization dedicated to defining standards, certifications, and best practices for cloud security. Given its focus and timing (emerging alongside cloud adoption), CSA seemed positioned to create a cloud-specific attestation framework that could serve as an alternative to SOC 2.

CSA did create the Security Trust Assurance and Risk (STAR) Program, which "encompasses key principles of transparency, rigorous auditing, and harmonization of standards."[22] STAR operates on three levels: self-assessment (Level 1), independent third-party certifications (Level 2), and continuous monitoring (Level 3).[23]

This appears to be exactly what was needed. Yet STAR did not become an alternative to SOC 2. Instead, it became a complement. Why?

STAR Attestation = SOC 2 + Cloud Controls Matrix: The critical finding is that CSA STAR Attestation is not an independent framework. It is an enhancement to SOC 2. According to CSA's own description, STAR Attestation is "a collaboration between CSA and the AICPA to provide guidelines for CPAs to conduct SOC 2 engagements using criteria from the AICPA (Trust Service Principles) and the CSA Cloud Controls Matrix."[24]

Translation: STAR Attestation is a SOC 2 Type II report with additional criteria from CSA's Cloud Controls Matrix. The same CPA firms perform the audits. The same SSAE 18 attestation standards apply. CSA added cloud-specific controls to the assessment, but the foundational framework remained SOC 2.

STAR Certification = ISO 27001 + Cloud Controls Matrix: Similarly, CSA STAR Certification builds atop ISO 27001. It acknowledges that an organization has implemented an ISMS per ISO 27001 and also features cloud-specific controls from the Cloud Controls Matrix.[25] Again, CSA enhanced an existing framework rather than creating an independent one.

Why CSA Chose This Approach: CSA's decision to augment SOC 2 and ISO 27001 rather than create a competing standard was pragmatic:

1. **Leverages existing audit infrastructure**: CPAs already perform SOC 2; certification bodies already perform ISO 27001. No need to build a new auditor ecosystem.
2. **Increases adoption**: Organizations already pursuing SOC 2 or ISO 27001 can add STAR more easily than switching to a new framework.
3. **Avoids market fragmentation**: Does not force organizations to choose between competing frameworks; instead, allows layering of assurance.

The Trade-Off: This approach increased STAR adoption but meant CSA STAR inherited the limitations of its underlying frameworks. SOC 2's technology gaps (no SBOM requirements, limited software supply chain coverage) carry forward into STAR Attestation. ISO 27001's organizational focus (rather than product-specific attestation) carries forward into STAR Certification.

What CSA Actually Contributed: The Cloud Controls Matrix (CCM) provides a comprehensive controls framework with 197 control objectives across 17 domains,[26] mapped to multiple standards including ISO 27001, NIST SP 800-53, PCI DSS, and AICPA Trust Services Criteria. CSA provided a valuable mapping and harmonization layer, enabling organizations to see how different frameworks align. But STAR did not replace SOC 2—it enhanced it.

Emerging Frameworks: Specialized, Not Comprehensive

In recent years, technology-focused frameworks have emerged to address specific gaps, particularly in software supply chain security.

in-toto: A framework designed to ensure the integrity of a software product from initiation to end-user installation.[27] It provides a generalized workflow to secure software supply chains and a framework for software attestations—signed documents that associate metadata with artifacts. Hosted by the Cloud Native Computing Foundation (CNCF), in-toto provides the technical foundation for supply chain attestation.

SLSA (Supply-chain Levels for Software Artifacts): A set of incrementally adoptable guidelines for supply chain security, established by industry consensus and backed by The Linux Foundation.[28] SLSA focuses on protecting software from source through deployment and allows users to make automated decisions about artifact integrity. SLSA recommends in-toto attestations as the vehicle to express provenance and other supply chain attributes.[29]

SBOM (Software Bill of Materials): While not an attestation framework per se, SBOMs provide transparency into software dependencies.[30] Executive Order 14028 (2021) mandated SBOMs for software sold to the federal government. Technical standards exist (CycloneDX, SPDX, SWID), and tools can auto-generate SBOMs in CI/CD pipelines.

Why These Don't Replace SOC 2: These frameworks address **specific technical gaps** but remain narrowly focused:

1. **Specialized scope**: Focus on software supply chain and artifact integrity, not broader operational controls
2. **Different use case**: Address build/release pipeline security, not organizational governance or service operations
3. **Early adoption stage**: Not yet widely accepted by procurement/risk teams as SOC 2 alternatives
4. **Complementary nature**: Designed to work alongside organizational attestations, not replace them

Example Layering:

- **SOC 2**: Attests that a SaaS vendor has appropriate security controls, incident response, access management, etc.
- **SLSA**: Attests that the software artifacts the vendor deploys were built securely with proper provenance
- **SBOM**: Lists what third-party components are in the software

All three address different aspects of risk. An ideal future state might include all three, but they serve distinct purposes.

The Window That Closed

Between 2000 and 2010, there was a window of opportunity. SAS 70's limitations were evident. Cloud computing was emerging. SaaS business models were proliferating. Software supply chain risks were becoming visible. Technology dependency chains were deepening.

This was the moment when a technology-focused organization could have created a comprehensive attestation framework purpose-built for modern technology risks. Such a framework might have

addressed software supply chain from the start, incorporated delivery pipeline security, included infrastructure dependency mapping, and prevented SOC 2's dominance in technology attestation.

It did not happen. ISACA could not, due to structural mission constraints. NIST would not, due to government agency boundaries. ISO addressed organizational governance, not product-specific attestation. CSA chose enhancement over competition. Emerging frameworks (SLSA, in-toto) arrived a decade later and addressed specialized needs, not comprehensive attestation.

By the time these technology-focused frameworks emerged, SOC 2 had achieved market dominance. The network effects were too strong. Procurement teams asked for SOC 2 by name. Vendors invested in achieving SOC 2 compliance. The ecosystem had formed around the accounting profession's framework.

Implications

The absence of a technology-focused alternative to SOC 2 is not the result of negligence or lack of expertise. It is the result of structural constraints:

Mission limitations: Professional associations (ISACA) and government agencies (NIST) cannot provide attestation without conflicting with their core missions.

Market dynamics: Network effects and first-mover advantage made it difficult for later entrants to compete with an established standard.

Infrastructure requirements: Creating a new attestation framework requires building an entire ecosystem—auditor training, accreditation mechanisms, market acceptance, professional liability frameworks. This is a massive undertaking.

Scope challenges: Comprehensive attestation must address organizational controls, operational processes, and technical specifics. Balancing breadth and depth while remaining technology-agnostic is extraordinarily difficult.

The result is that SOC 2—designed by accountants, governed by an accounting body, and performed by CPA firms—remains the dominant framework for technology risk attestation. Technology-focused organizations contributed frameworks, guidance, and specialized standards. But none created a comprehensive alternative.

This is the historical accident: not that SOC 2 exists, but that it is the only option. The market needed technology risk attestation. The AICPA provided it when no one else did. That was valuable. But the governance structure—accountants governing technology attestation—has implications for what evolves and what does not.

The gaps in SOC 2 (software supply chain, delivery mechanisms, application-layer dependencies) persist not because CPAs are incapable—they demonstrably have the technical resources and institutional infrastructure—but because the structure does not naturally prioritize innovation in those areas. Technology-focused organizations could have created alternatives but were structurally constrained from doing so.

The window closed. The market settled. And financial institutions now manage technology risk using frameworks that apply auditing methodology developed for financial controls to an increasingly technical domain.

The absence of technology-focused alternatives meant TPRM frameworks remained anchored to the accounting profession's entity-based worldview. Yet even as guidance evolved, a more fundamental

question remained unaddressed: were regulators—and institutions—managing the right unit of risk?

Endnotes - Chapter 3

[1] Information Systems Audit and Control Association, ScienceDirect Topics, https://www.sciencedirect.com/topics/science/information-systems-audit-and-control-association

[2] ISACA, "COBIT Framework," https://www.isaca.org/resources/cobit

[3] ISACA, "Certifications," https://www.isaca.org/credentialing

[4] ISACA, "IT Audit Resources," https://www.isaca.org/resources/it-audit

[5] ISACA, "Standards, Guidelines, Tools and Techniques," ISACA Journal, Volume 6, 2020, https://www.isaca.org/resources/isaca-journal/issues/2020/volume-6/standards-guidelines-tools-and-techniques

[6] "National Institute of Standards and Technology," Government Contracts Law Blog, https://www.governmentcont institute-of-standards-and-technology-nist/

[7] NIST, "Cybersecurity Framework: Assessment & Auditing Resources," https://www.nist.gov/cyberframework/ass auditing-resources

[8] NIST Special Publication 800-53, "Security and Privacy Controls for Information Systems and Organizations"

[9] NIST Special Publication 800-171, "Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations"

[10] CISA, "Secure Software Development Attestation Common Form," April 2023, https://www.cisa.gov/sites/defaul 04/secure-software-self-attestation_common-form_508.pdf (referencing NIST SP 800-218)

[11] Security Compass, "NIST CSF Compliance & Certification: Everything You Need to Know," https://www.securitycompass.com/blog/nist-csf-compliance-certification/

[12] NIST, "Cybersecurity Framework," https://www.nist.gov/cyberframework/assessment-auditing-resources

[13] Executive Order 14028, "Improving the Nation's Cybersecurity," May 12, 2021

[14] K&L Gates, "Secure Software Regulations and Self-Attestation Required for Federal Contractors," May 19, 2023, https://www.klgates.com/Secure-Software-Regulations-and-Self-Attestation-Required-for-Federal-Contractors-5-19-2023; NASA SEWP, "Software Attestation," https://www.sewp.nasa.gov/software_atte

[15] Microsoft Azure Compliance, "NIST CSF," https://learn.microsoft.com/en-us/azure/compliance/offerings/offeri nist-csf

[16] PreVeil, "NIST 800-171 Compliance: Third-Party Assessments Now Required," https://www.preveil.com/blog/ni 800-171/

[17] ComplianceForge, "NIST CSF 2.0 Certification," https://complianceforge.com/scf-certifications/nist-csf-certification/

[18] ISMS.online, "ISO 27001 Certification vs SOC 2 Attestation: What Are the Key Differences?" https://www.isms.online/iso-27001/iso-27001-certification-vs-soc-2-attestation/

[19] ISMS.online, “ISO 27001 Certification vs SOC 2 Attestation”

[20] ISO/IEC 27017:2015 (cloud services information security controls); ISO/IEC 27018:2019 (protection of PII in public clouds)

[21] Cloud Security Alliance, “About CSA,” https://cloudsecurityalliance.org/about/. CSA was founded in December 2008 following a session at ISSA CISO Forum and launched publicly at RSA Conference 2009.

[22] Cloud Security Alliance, “STAR Program,” https://cloudsecurityalliance.org/star

[23] Schellman, “Understanding the CSA STAR Program,” https://www.schellman.com/blog/pci-compliance/understanding-csa-star-program

[24] Microsoft Azure Compliance, “CSA STAR Attestation,” https://learn.microsoft.com/en-us/azure/compliance/offerings/offering-csa-star-attestation

[25] TÜV SÜD, “CSA STAR Certification,” https://www.tuvsud.com/en-us/services/auditing-and-system-certification/csa-star

[26] Cloud Security Alliance, “STAR Program Overview,” https://cloudsecurityalliance.org/artifacts/star-program-overview (Cloud Controls Matrix details)

[27] in-toto Framework, https://in-toto.io/

[28] SLSA Framework, “Frequently Asked Questions,” https://slsa.dev/spec/v1.0/faq

[29] SLSA Blog, “in-toto and SLSA,” May 2023, https://slsa.dev/blog/2023/05/in-toto-and-slsa

[30] SLSA Blog, “SLSA and SBOM,” May 2022, https://slsa.dev/blog/2022/05/slsa-sbom

Part II: The Framework Gap

Chapter 4: Parties vs Dependencies

The fundamental problem with contemporary third-party risk management is architectural. TPRM programs are built to see, assess, and manage **legal entities**—parties with contracts, attestations, and business relationships. But software supply chain risk propagates through **technical dependencies**—code libraries, infrastructure services, build pipelines, and deployment mechanisms that exist largely independent of organizational boundaries.

The problem is not insufficient effort or inadequate resources—it is a structural blind spot embedded in how TPRM frameworks conceptualize the vendor ecosystem. The industry manages "parties" but should be managing "dependencies."

The Entity-Centric View

Modern TPRM programs organize the world hierarchically by contractual relationships:

- **Third-party**: The vendor with whom you have a direct contract
- **Fourth-party**: The vendor's vendor, typically disclosed through SOC 2 subservice organization carve-outs or contractual notifications
- **Fifth-party**: The fourth-party's vendor, identified occasionally in comprehensive due diligence

This model maps cleanly to questions TPRM teams are trained to answer: Who are we doing business with? What services do they provide? What data do they access? What are our contractual protections? Do they have appropriate insurance? Have they provided a SOC 2 report?

At the fourth-party level, visibility narrows substantially. Critical fourth-party relationships identified in most financial institution TPRM programs consist primarily of cloud infrastructure providers—AWS, Microsoft Azure, Google Cloud—and occasionally colocation service providers. This gives TPRM teams an infrastructure-level view of vendor dependencies: where systems run, where data is stored, what network connectivity exists.

This infrastructure view is useful. It is also radically incomplete.

What the Entity View Cannot See

Consider a typical fintech SaaS vendor serving community banks. The TPRM program sees:

- **Third-party**: The fintech company itself
- **Fourth-party**: AWS (disclosed on the fintech's SOC 2 as subservice organization)
- Possibly: Twilio (if SMS notifications are mentioned in the system description)
- Possibly: Stripe (if payment processing is disclosed)

What the TPRM program does not see, because no framework requires disclosure:

Application-layer dependencies: The 200+ open source libraries imported into the fintech's codebase—logging frameworks, authentication libraries, data validation utilities, API clients, cryptographic implementations, date/time handlers, JSON parsers, HTTP request handlers. According to the 2024 Synopsys Open Source Security and Risk Analysis (OSSRA) report, 96% of commercial

codebases contain open source components, with the average codebase comprising 77% open source code.[1.1] Each of these components is a dependency. Each is maintained by individuals or organizations the fintech has no contractual relationship with. Each could contain vulnerabilities affecting your data.

Software delivery dependencies: The CI/CD pipeline through which code is built, tested, and deployed—GitHub Actions, Jenkins, GitLab CI, CircleCI. The artifact repositories storing compiled code—Docker Hub, npm registry, PyPI. The code signing infrastructure verifying package integrity. The configuration management tools controlling deployment. A compromise anywhere in this chain means malicious code reaches production, regardless of the fintech's application security practices.

Security tooling dependencies: The SIEM platform collecting logs, the EDR solution monitoring endpoints, the vulnerability scanner identifying weaknesses, the secrets management system protecting API keys, the identity provider controlling access. If these tools are managed by external providers—Datadog, CrowdStrike, Qualys, HashiCorp, Okta—each represents a dependency with privileged access to sensitive systems. Yet they rarely appear in TPRM inventories because they're not "vendors" in the traditional procurement sense.

Development and consulting relationships: The software development firm the fintech contracted to build v2.0 of their platform. The specialized consultants who configured the authentication system. The offshore development team handling maintenance. These parties directly influence the design and implementation of systems processing your customer data, yet they are invisible to TPRM programs focused on the fintech as a unified legal entity.

Operational systems of the vendor: The fintech's own third-party relationships—their payroll system, CRM, compliance management platform, ticketing system, communication tools. A breach of the vendor's HR system could expose employee credentials used to access production environments. Yet fourth-party risk assessment in practice rarely extends to vendors' own operational systems.

The nth-Party Dependency Concept

The depth at which TPRM programs can see is, at best, n=4 (fourth-party), and only then for specific categories of relationships—primarily infrastructure. But the actual dependency chains extend to n=5, n=10, n=20, or effectively infinite depth as dependencies recursively pull in their own dependencies.

The term **nth-party dependency** captures this reality: every third-party relationship is its own cascading set of technical, organizational, and operational dependencies. When these dependency chains are mapped across an entire vendor portfolio, they converge at points no current TPRM framework is equipped to detect.

This convergence is where concentration risk hides.

The Log4j Validation

The Log4j incident (CVE-2021-44228, detailed in Chapter 7) validated the nth-party dependency problem empirically.[1] Apache Log4j is a Java logging library, open source, maintained by volunteers, used directly or transitively in thousands of enterprise applications. It is not a vendor. It does not have a SOC 2 report. It does not appear in any TPRM register. It has no contract, no SLA, no business entity identifier.

Yet when the vulnerability was disclosed in December 2021, organizations with diversified vendor portfolios discovered they had catastrophic concentration risk. Applications from vendors with no apparent relationship—different industries, different technology stacks, different hosting environments—all pulled in Log4j, either directly or through other libraries that depended on it.

The impact was widespread across the software ecosystem, affecting both directly declared dependencies and code that had been copied into vendor codebases. No dependency scanner could find all instances. No SBOM would disclose everything.

Financial institutions with comprehensive TPRM programs, detailed vendor inventories, and SOC 2 reports for every critical vendor spent weeks trying to answer a single question: "Where are we exposed to Log4j?" The programs built to manage third-party risk could not answer the question because the dependency did not map to a party.

This was not a theoretical edge case. The Cybersecurity and Infrastructure Security Agency (CISA) stated Log4Shell was "one of the most serious flaws seen in recent years."[4] Exploitation began within hours of public disclosure. Nation-state actors, ransomware groups, and cryptocurrency miners all targeted the vulnerability.[4.1] Organizations without dependency visibility operated blind.

The MOVEit Pattern

MOVEit—Progress Software's managed file transfer solution—illustrated the same blind spot at the application layer. The CL0P ransomware group exploited a zero-day SQL injection vulnerability in MOVEit Transfer in May 2023, ultimately affecting over 2,700 organizations and 95.8 million individuals.[5]

For financial institutions, the challenge was not that they used MOVEit directly (though some did). The challenge was that their vendors used MOVEit, often without disclosure. MOVEit is application-level infrastructure—a tool embedded in vendor operations to move files securely between systems. It is not a "subservice organization" in SOC 2 terminology. It does not fit cleanly into fourth-party risk frameworks. It simply existed, undisclosed, within vendor environments.

When the vulnerability became public, TPRM teams faced the same question they faced with Log4j: "Which of our vendors are exposed?" The answer required direct outreach to every vendor asking "Do you use MOVEit?" Many vendors did not know immediately. Some learned they were exposed only after their own subcontractors notified them. The dependency chain—financial institution → vendor → vendor's file transfer tool—was invisible until it became a crisis.

Victims included major financial services providers: PwC, Ernst & Young, TIAA, T. Rowe Price, pension systems, and payroll processors.[6] The estimated cost reached $15.8 billion.[7] The incident was entirely foreseeable from a technical perspective—file transfer applications are high-value targets, SQL injection vulnerabilities are a known threat class, and defense-in-depth principles should have limited exposure. But TPRM frameworks do not ask "What file transfer solutions do your vendors use?" They ask "Do you have a SOC 2?" and move on.

The SolarWinds and 3CX Software Delivery Problem

SolarWinds (2020) and 3CX (2023) demonstrated dependency risk at a different layer entirely: the software delivery pipeline. Both incidents involved attackers compromising the build process to inject malicious code into legitimate software updates. Customers received signed, validated, legitimate-looking updates that contained sophisticated backdoors.

SolarWinds held SOC 2 Type II certification at the time of the attack.[8] The company had access controls, change management processes, vulnerability scanning, and incident response plans. None of these prevented Russian intelligence operatives (SVR) from injecting the SUNBURST malware into Orion platform updates. The compromise occurred in what SolarWinds described as a "millisecond window" during the build process—an attack vector SOC 2 controls are not designed to address.[9]

To be clear: this was a nation-state attack with essentially unlimited resources and sophistication. Better auditing almost certainly would not have detected it. The SVR maintained access for over a year while evading detection by one of the most security-conscious customer bases in the world. The lesson here: software delivery mechanisms represent a category of dependency risk that traditional TPRM frameworks do not address, and adversaries with nation-state capabilities can exploit gaps that even sophisticated security programs cannot close.

Approximately 18,000 organizations installed trojanized Orion updates, including nine federal agencies and numerous Fortune 500 companies.[10] The malware established command-and-control communications, exfiltrated data, and enabled lateral movement within victim networks. From a TPRM perspective, the incident was invisible until after the fact. Customers had performed due diligence, received attestations, and monitored vendor security posture. The dependency that mattered—the integrity of SolarWinds' build environment—was never in scope.

3CX extended the pattern to cascading supply chain compromise.[11] Attackers first trojanized X_TRADER, a trading software package from Trading Technologies. A 3CX employee downloaded the compromised software on a personal device, which led to VPN credential theft and ultimately compromise of both Windows and macOS build servers. Malicious updates were signed with valid 3CX certificates and distributed to customers.

This is supply chain attack as dependency chain: Attacker → Trading Technologies software → 3CX employee → 3CX build environment → 3CX customers. The dependency depth exceeded what any TPRM program is structured to see. Trading Technologies was not a fourth-party to 3CX's customers. The employee's personal software was not a disclosed subcontractor. Yet the dependency path was real and exploitable.

The xz Utils Pattern: Deliberate Open Source Infiltration

The incidents above involve external attackers compromising vendor systems. But open source supply chain risk includes a more insidious pattern: patient adversaries who spend years building trust before attacking from within.

In early 2024, the xz Utils backdoor (CVE-2024-3094) exposed what security researchers called the most sophisticated open source supply chain attack ever documented.[12.1] An individual using the handle "Jia Tan" began contributing to the xz data compression library in October 2021. Over the next two years, Jia Tan submitted legitimate patches, fixed bugs, and built credibility with the project maintainer.

The social engineering was methodical. Multiple sockpuppet accounts—likely the same actor—pressured the original maintainer, who had publicly mentioned experiencing burnout, to accept more help. By November 2022, Jia Tan had been granted maintainer status. By January 2024, Jia Tan controlled the project's website. In February and March 2024, the backdoor was inserted—sophisticated code that would have enabled remote code execution on millions of Linux systems.

The attack was discovered only by accident: a Microsoft engineer noticed unusual CPU patterns during routine system monitoring. If not for this anomaly, the backdoor could have propagated to

virtually every major Linux distribution.

The xz Utils pattern reveals that supply chain risk includes not just accidental vulnerabilities or external compromises, but deliberate infiltration by sophisticated adversaries willing to invest years building trust. The OpenSSF and OpenJS Foundation warned in April 2024 that xz Utils-style social engineering "may not be an isolated incident" and that similar infiltrations may already exist undetected.[12.2]

This attack vector is fundamentally different from the others discussed. Log4j was an accidental vulnerability. SolarWinds was an external compromise of build systems. xz Utils was insider access obtained through patient social engineering. No amount of vendor due diligence or SOC 2 review would detect an adversary who spends two years building legitimate credibility before attacking.

The CrowdStrike Update Delivery Risk

The July 2024 CrowdStrike incident demonstrated a related but distinct problem: software update mechanisms as concentration points. CrowdStrike Falcon is an endpoint detection and response (EDR) solution deployed widely across enterprise environments, including financial institutions. On July 19, 2024, a faulty sensor configuration update caused Windows systems to crash with kernel errors, affecting approximately 8.5 million devices globally.[12]

The estimated cost exceeded $10 billion.[13] Airlines grounded flights, hospitals canceled procedures, banks experienced operational disruptions. The incident was not a security compromise—it was a quality assurance failure. But from a dependency risk perspective, the pattern is instructive.

CrowdStrike Falcon operates at the kernel level with privileged system access. Configuration updates deploy automatically to ensure rapid threat response. Customers cannot stage updates or test them in isolated environments before production deployment because the value proposition is real-time threat protection. This creates a structural dependency: organizations rely on CrowdStrike's internal quality assurance to prevent disruptions.

TPRM programs treat security tools as solutions that reduce risk, not as dependencies that introduce it. Vendor assessments focus on whether the security tool is effective—does it detect threats, respond to incidents, meet compliance requirements? They do not systematically assess the risk introduced by the tool itself—what happens if an update is faulty, if the vendor experiences an outage, if the software conflicts with system updates?

The CrowdStrike incident revealed concentration risk that was technically visible but operationally invisible. Many financial institutions use CrowdStrike across their entire Windows infrastructure. This concentration exists by design—it is efficient, provides unified visibility, and simplifies management. But it also means a single vendor update failure creates enterprise-wide disruption. TPRM frameworks missed this concentration because the dependency was on update delivery infrastructure, not on a "party."

Why the Entity Model Persists

If the entity-centric TPRM model has such fundamental blind spots, why does it persist?

The answer is that it maps to the world as it existed when third-party risk management was formalized. In the 1990s, vendor management was an administrative function handling contracts and procurement.[14] Vendors provided discrete products or services—a payroll system, a card

processor, a loan origination platform. Each vendor was a distinct entity with clear boundaries. Assessing vendor risk meant assessing the vendor as an organization.

Cloud computing introduced the concept of critical fourth parties, primarily infrastructure providers. TPRM evolved to recognize that a vendor's reliance on AWS or Azure was a risk factor worth evaluating. But this evolution maintained the entity-centric model—it simply added another layer of organizations to assess.

The transformation from "Vendor Management" to "Third-Party Risk Management" carried an implicit promise: by managing the risk of third parties, organizations would inherently manage the risks of their fourth parties, fifth parties, and all downstream dependencies. But this promise was based on a misinterpretation.

"Third-Party Risk Management" does not mean "management of third parties." It means "management of risks associated with third-party relationships." Those risks include the technical dependencies, software supply chains, operational systems, and delivery mechanisms that third parties rely upon. The entity model treats these as details within vendor risk assessment. In reality, they are the risks that matter most.

The Dependency Model Alternative

A dependency-centric TPRM model would start from a different question: "What does our organization depend on to deliver services?" rather than "Who are our vendors?"

The answer includes:

- Infrastructure: cloud platforms, data centers, network providers, DNS services
- Applications: software systems, APIs, databases, middleware
- Code libraries: open source components, commercial SDKs, internal shared libraries
- Delivery mechanisms: CI/CD pipelines, package registries, artifact repositories, update channels
- Security tooling: monitoring platforms, EDR agents, SIEM systems, identity providers
- External expertise: development firms, consultants, managed service providers
- Operational systems: both organizational (our systems) and vendor-side (their systems)

Some of these map to legal entities that can be contracted with, assessed through due diligence, and held accountable through SLAs. Many do not. Open source libraries have communities, not vendors. Build pipelines have operators, not contracts. Update channels have trust models, not attestations.

A dependency-centric model would:

- **Inventory technical dependencies** across the application stack, not just contractual relationships
- **Map convergence points** where multiple critical systems depend on the same component or service
- **Assess substitutability** and switching costs for concentrated dependencies
- **Monitor vulnerability disclosures** for both direct and transitive dependencies
- **Evaluate delivery mechanisms** as part of vendor security assessment
- **Distinguish between parties and platforms**—AWS is both a vendor (legal entity) and a platform (dependency shared across many vendors)

This model does not replace entity-based TPRM. Contracts, attestations, and vendor governance remain essential. But it recognizes that legal relationships are an incomplete proxy for technical

dependencies. Risk propagates through code, not through corporate structures.

The industry has spent twenty years refining entity-based TPRM. It has built sophisticated frameworks, hired specialized teams, and invested in vendor management platforms. Yet the incidents that cause the most damage—Log4j, SolarWinds, MOVEit, CrowdStrike—trace to dependencies the frameworks were never designed to see.

TPRM programs clearly need to evolve beyond the entity model. What's less clear is how quickly they can build the capability to see and manage dependencies at the scale and depth where risk actually exists.

The next chapter examines specifically where current compliance frameworks fall short against these dependency-level risks.

Endnotes

[1] For detailed Log4j statistics and impact analysis, see Chapter 7. Medium analysis: "Throwback on the Log4Shell vulnerability," https://medium.com/@dolor3sh4ze/throwback-on-the-log4shell-vulnerability-5383d05c5e72

[1.1] Synopsys, "2024 Open Source Security and Risk Analysis (OSSRA) Report," https://www.synopsys.com/softwar integrity/resources/analyst-reports/open-source-security-risk-analysis.html. The report found that 96% of audited commercial codebases contained open source components, with open source comprising 77% of total code on average.

[4] Cybersecurity and Infrastructure Security Agency (CISA): "Apache Log4j Vulnerability Guidance," https://www.cisa.gov/news-events/news/apache-log4j-vulnerability-guidance

[4.1] Cloudflare reported exploitation attempts began within 9 minutes of the Log4j vulnerability disclosure on December 9, 2021. Check Point documented over 800,000 attack attempts within the first 72 hours. Microsoft Threat Intelligence Center identified exploitation by nation-state actors from China, Iran, North Korea, and Turkey, as well as ransomware operators and cryptomining groups. See: Cloudflare Blog, "Exploitation of CVE-2021-44228 before public disclosure"; Check Point Research, "Log4j Exploitation Trends"; Microsoft Security Blog, "Guidance for preventing, detecting, and hunting for CVE-2021-44228 exploitation," December 2021.

[5] Emsisoft: "Unpacking the MOVEit Breach: Statistics and Analysis," https://www.emsisoft.com/en/blog/44123/u the-moveit-breach-statistics-and-analysis/

[6] American Banker: "15 banks, credit unions confirm MoveIt data breaches," https://www.americanbanker.com/new banks-credit-unions-confirm-moveit-data-breaches

[7] Wikipedia: "2023 MOVEit data breach," https://en.wikipedia.org/wiki/2023_MOVEit_data_breach

[8] Venminder: "SolarWinds Data Hack Is a Reminder Why Third-Party Risk Management Is Important," https://www.venminder.com/blog/solarwinds-hack-third-party-risk-importance

[9] TechTarget: "SolarWinds hack explained: Everything you need to know," https://www.techtarget.com/whatis/fea hack-explained-Everything-you-need-to-know

[10] U.S. Senate Republican Policy Committee: "The SolarWinds Cyberattack," https://www.rpc.senate.gov/policy-papers/the-solarwinds-cyberattack

[11] Sonatype Blog: "Another SolarWinds? The Latest Software Supply Chain Attack on 3CX," https://www.sonatype.com/blog/another-solarwinds-the-latest-software-supply-chain-attack-on-3cx

[12.1] Wikipedia: "XZ Utils backdoor," https://en.wikipedia.org/wiki/XZ_Utils_backdoor; Securelist: "Social engineering aspect of the XZ incident," https://securelist.com/xz-backdoor-story-part-2-social-engineering/112476/; Dark Reading: "Attacker Social-Engineered Backdoor Code Into XZ Utils," https://www.darkreading.com/application-security/attacker-social-engineered-backdoor-code-into-xz-utils

[12.2] Nextgov: "Linux backdoor was a long con, possibly with nation-state support, experts say," https://www.nextgov.com/cybersecurity/2024/04/linux-backdoor-was-long-con-possibly-nation-state-support-experts-say/395511/; OpenSSF and OpenJS Foundation joint warning issued April 2024.

[12] Wikipedia: "2024 CrowdStrike-related IT outages," https://en.wikipedia.org/wiki/2024_CrowdStrike-related_IT_outages

[13] Worldwide financial damage from the CrowdStrike incident estimated at $10 billion or more. Fortune 500 companies alone faced $5.4 billion in direct losses per Parametrix study. Sources: Wikipedia, "2024 CrowdStrike-related IT outages," https://en.wikipedia.org/wiki/2024_CrowdStrike-related_IT_outages; Fortune, "CrowdStrike outage will cost Fortune 500 companies $5.4 billion," August 2024.

[14] The evolution from "Vendor Management" to "Third-Party Risk Management" is documented in OCC guidance spanning 1996-2023. OCC Banking Circular 196 (1996) focused on basic vendor oversight; OCC Bulletin 2001-47 introduced "risk management" framing; OCC Bulletin 2013-29 formalized comprehensive TPRM expectations. Federal Reserve SR Letters followed similar evolution. See: OCC Comptroller's Handbook, "Third-Party Relationships: Risk Management Guidance," 2023.

Chapter 5: Framework Gaps by Design

The structural mismatch between entity-based oversight and dependency-level risk is not an accident. It is embedded by design in every major third-party risk management and assurance framework currently in use. These frameworks were created to address legitimate risk management concerns, but they were created for a different technological landscape. As software supply chains have grown in complexity and concentration, the frameworks have remained largely unchanged—or their updates have not bridged the fundamental gap.

Three frameworks dominate the TPRM landscape in financial services: DORA (Digital Operational Resilience Act), SOC 2 (Service Organization Control Type 2), and SBOM/CISA guidance derived from Executive Order 14028. Each addresses a portion of the supply chain risk spectrum. None addresses the application-layer dependency problem.

DORA: Entity-Focused Supply Chain Transparency

The Digital Operational Resilience Act, which became applicable January 17, 2025, represents the most comprehensive regulatory framework for ICT third-party risk management ever implemented in financial services.[1] DORA applies to over 22,000 financial entities across the EU and brings critical ICT third-party providers under direct regulatory oversight for the first time.[1.1]

DORA requires financial institutions to maintain a **Register of Information** cataloging all ICT third-party service provider relationships.[2] This register must document the vendor name, services provided, data accessed, criticality classification, contract dates, and dependencies. Article 29(2) explicitly requires assessment of "long or complex chains of subcontracting," and Article 28(3) mandates identification of "material sub-contractors" that "effectively underpin" critical services.[3]

This is meaningful progress beyond status quo TPRM programs. DORA forces systematic inventory, criticality classification, and subcontractor visibility. Financial institutions can no longer limit vendor oversight to direct third-party relationships and ignore downstream dependencies.

But DORA remains fundamentally entity-focused. It addresses the question "who are your subcontractors?" not "what's in your code?"

Consider Article 30(3)(f), which requires contracts to include provisions on "the use of subcontractors, including that the financial entity is informed of any planned change concerning the use of subcontractors."[3] This requirement is valuable for maintaining visibility into outsourcing chains—if a vendor shifts from self-hosting to AWS, DORA requires notification. But it does not reach application-level dependencies.

Code libraries do not have Legal Entity Identifiers (LEIs). They do not issue subcontracting notifications. They do not fit into DORA's framework because they are not subcontractors—they are embedded components. A financial institution's vendor may use hundreds of open source libraries maintained by individuals or loosely organized communities. None of these trigger DORA's subcontracting notification requirements. None appear in the Register of Information. Yet any of them could be the next Log4j.

The European Commission itself acknowledged the limitations of entity-level oversight in an instructive way. In July 2024, the draft Regulatory Technical Standards included a proposed Article 5 that would have required financial entities to establish processes for "ongoing monitoring of the ICT third-party service providers' supply chains." This would have pushed visibility down to code-level dependencies.

The European Commission **rejected this provision** in the final RTS.[4] The official justification was that such monitoring would be "disproportionate" and that existing requirements for subcontractor notification were sufficient. In other words, even the most ambitious regulatory framework in financial services pulled back from requiring dependency-level visibility when confronted with the operational complexity.

DORA's greatest contribution to supply chain risk management may be its unintentional demonstration of where entity-based frameworks reach their limit. Financial institutions subject to DORA will have comprehensive visibility into vendors and their subcontractors. They still will not know which of those vendors are exposed to the next critical code library vulnerability until the vulnerability is disclosed and they ask.

SOC 2: Designed for Infrastructure Controls, Not Application Dependencies

SOC 2 Type II reports have become the de facto assurance baseline for SaaS vendors in financial services. Banks require them in vendor contracts, TPRM teams rely on them for due diligence, and auditors treat them as credible third-party attestations. SOC 2 provides genuine value for what it was designed to cover: access controls, change management, logical and physical security, availability, and data handling practices.

But regulators have explicitly cautioned against over-reliance on these reports. The FFIEC IT Examination Handbook states: **"Users of audit reports or reviews should not rely solely on the information contained in the report to verify the internal control environment of the TSP."**[12] This guidance applies directly to SOC 2 reports—the FFIEC specifically identifies SOC 1, SOC 2, and SOC 3 attestations as examples of the audit reports requiring supplemental verification. Financial institutions must implement additional monitoring and verification procedures beyond reviewing SOC reports.

SOC 2's scope reflects its origins and the risk landscape when it was designed. The Trust Services Criteria were finalized in 2017 and remain substantively unchanged (see Chapter 2 for the detailed SOC 2 evolution timeline).[5][6] The framework was not designed to require Software Bills of Materials, mandate CI/CD pipeline security assessment, or address application-layer dependencies—these risk categories were not yet widely recognized as material threats to financial institutions. When SOC 2 reports disclose subservice organizations, they almost universally disclose infrastructure providers (AWS, Azure, GCP) while application-layer dependencies remain invisible.

SOC 2's vendor-defined scoping creates additional gaps. A vendor could scope their SOC 2 to cover internal IT operations while excluding the customer-facing application entirely—and this would comply with AICPA standards. Even when application security is in scope, auditors test high-level process controls, not technical details like dependency vulnerability management or build pipeline isolation.

The AICPA introduced **SOC for Supply Chain** in March 2020 explicitly to address software supply chain attacks,[7] but this framework uses the same Trust Services Criteria and focuses primarily on physical manufacturing. No major audit firm has developed SOC for Supply Chain as a standard offering for software vendors. The framework has not achieved material adoption in financial services TPRM.[7.1] SOC 2 Type II remains the standard, and SOC 2 does not see application-layer dependencies.

(See Chapter 9 for detailed analysis of how SOC 2 could evolve to address emerging supply chain risks.)

SBOM/CISA Guidance: Standards Without Mandates

Executive Order 14028 (Improving the Nation's Cybersecurity), issued in May 2021, established the most comprehensive federal software supply chain security requirements ever implemented in the United States.[8] The order directed NIST to issue guidelines on software supply chain security and directed CISA to develop an attestation form for software producers.

The technical standards exist. CycloneDX and SPDX provide machine-readable SBOM formats.[9] Tools like Syft, Trivy, and native GitHub/GitLab SBOM generation can automatically produce SBOMs in CI/CD pipelines. CISA's Secure Software Development Attestation Form (March 2024) requires software producers to attest to core secure development practices including:[10]

- Separation of build environments from development and production
- Multi-factor authentication for all personnel with access to software development environments
- Encryption of data in transit
- Trusted source code supply chains
- Provenance for third-party components
- Vulnerability disclosure programs
- SBOM capability

These requirements are comprehensive and technically sound. They address many of the control gaps that enabled the SolarWinds and 3CX attacks. They create a baseline expectation for software supply chain transparency.

But they apply only to federal contractors selling software to government agencies. Financial services institutions are not covered. There is no equivalent OCC, FDIC, Federal Reserve, or FFIEC guidance requiring financial institutions to collect SBOMs from vendors or to verify CISA attestations.[11] There is no examination manual checklist item for SBOM collection. There is no regulatory expectation.

The result is a demand problem masquerading as a technical problem. For organizations with modern CI/CD infrastructure, SBOM generation is technically straightforward—a single additional pipeline step can produce a machine-readable dependency inventory. However, the reality is more complex for legacy applications, acquired codebases, products with extensive runtime dependencies, or systems built by contractors no longer engaged.[11.1] The technical barrier is real but varies dramatically by technology stack and organizational maturity. Vendors are not refusing to generate SBOMs solely because it is technically difficult. They are not generating them because no customer is requiring it and no regulator is examining it.

Why would a fintech generate an SBOM when:

- No customer requires it contractually
- No audit framework tests for it
- No regulator examines it
- It costs engineering time with no return on investment
- There is zero competitive disadvantage for not having one

The irony is profound. The technology exists. The standards exist. The federal government has demonstrated that SBOM requirements can be implemented at scale. But in financial services—the sector most dependent on SaaS vendors, most exposed to supply chain concentration risk, and most heavily regulated for third-party risk—SBOM adoption remains voluntary and rare.

The Void: Where Frameworks Do Not Meet

The gap between entity-level oversight and code-level dependency transparency is not a minor oversight that can be addressed through incremental updates. It is a structural void where no framework currently provides comprehensive assurance.

Consider how major incidents map to this void:

Incident	Risk Category	Entity-Level Framework Coverage	Dependency-Level Gap
SolarWinds (2020)	Software delivery pipeline	DORA: Would require notification of build environment changes, but compromise was undetected; SOC 2: SolarWinds held SOC 2 certification—controls did not prevent build compromise; SBOM: Would document components in delivered software, but not detect malicious injection during build	Compromise occurred in build process, not infrastructure. SOC 2 tested access controls and change management, but not build integrity or artifact signing. Attack vector outside audit scope.
Log4j (2021)	Code library vulnerability	DORA: Would require notification of subcontractor changes, but Log4j is not a subcontractor; SOC 2: CC9.2 covers vendor relationships, not embedded libraries; SBOM: Would have enabled rapid exposure identification, but not required in financial services	Embedded in applications invisibly. Not a legal entity. No contract, no SOC 2, no disclosure obligation. Organizations could not identify exposure for weeks.
Kaseya VSA (2021)	Remote management platform	DORA: Kaseya would appear in register if directly contracted, but many institutions affected through MSP relationships; SOC 2: Kaseya's SOC 2 did not prevent supply chain compromise; SBOM: Not applicable—attack targeted software delivery mechanism	Ransomware deployed through compromised software update affected 1,500+ organizations.[13.1] Fourth-party exposure invisible to institutions whose MSPs used Kaseya.
3CX (2023)	VoIP software supply chain attack	DORA: 3CX would appear in register if used directly; upstream dependency on X_TRADER software invisible; SOC 2: 3CX held SOC 2; did not cover build pipeline compromise; SBOM: Could have revealed compromised dependency if generated and monitored	Attackers compromised 3CX build process through trojanized upstream dependency. 600,000+ organizations affected.[13.2] Software signed with valid certificates passed security checks.
MOVEit (2023)	Application-layer tool	DORA: MOVEit would appear in register only if treated as "ICT third-party service provider"—likely not, since it is embedded tooling; SOC 2: Could be disclosed as subservice organization, but is not required; SBOM: Would not appear—MOVEit is operational infrastructure, not code dependency	Embedded in vendor systems for file transfer. Not disclosed as subservice organization on most SOC 2 reports. TPRM teams learned of exposure only after breach notification.
xz Utils (2024)	Code library backdoor attempt	DORA: xz Utils is not a subcontractor—it is an open source compression library; SOC 2: No visibility into individual code libraries embedded in applications; SBOM: Would have enabled detection of affected versions if	Backdoor inserted by compromised maintainer nearly reached production systems globally. Discovered only through alert engineer's investigation, not through systematic dependency monitoring.

Each incident exploited the same gap: the risk propagated through a technical dependency that did not map to an entity that frameworks are designed to assess.

The Missing Framework Layer

What would a framework that bridged this gap look like? It would need to:

1. **Mandate SBOM collection** from all critical software vendors, with standardized formats (CycloneDX or SPDX) and machine-readable data
2. **Require dependency vulnerability disclosure** including transitive dependencies, with timelines for remediation and customer notification
3. **Assess CI/CD pipeline security** including build environment isolation, artifact signing, provenance attestation, and deployment controls
4. **Evaluate software delivery mechanisms** as part of vendor security assessment—how do updates deploy, who controls them, what testing occurs before release
5. **Distinguish between infrastructure and application dependencies**—AWS is both a vendor and a platform that many vendors share; this distinction matters for concentration risk
6. **Map convergence points** where multiple critical vendors depend on the same underlying components or services
7. **Create audit accountability** for dependency-level controls through third-party attestation, not just vendor self-assessment

No current framework provides this. DORA goes furthest on entity-level visibility but stops at subcontractor chains. SOC 2 provides process-level assurance for vendor controls but does not reach code dependencies. SBOM guidance establishes technical standards but has no regulatory mandate in financial services.

The void exists because each framework was designed for a specific purpose and reflects the technological assumptions of its era. DORA reflects regulatory concern about outsourcing and cloud concentration. SOC 2 reflects audit practices for IT controls and data handling. SBOM guidance reflects federal government cybersecurity priorities. None was designed to provide end-to-end software supply chain assurance.

Financial institutions today face a choice: rely on frameworks that were not built for software supply chain risk, or build supplemental capabilities that existing frameworks do not address. The incidents of recent years demonstrate that relying on existing frameworks alone is insufficient. The question is whether institutions will build those capabilities proactively or wait for the next crisis to reveal gaps that could have been anticipated.

Framework limitations are only part of the challenge. Equally important is whether TPRM teams have the technical expertise to evaluate these risks—even if frameworks were updated tomorrow.

Endnotes - Chapter 5

[1] European Parliament and Council of the European Union, "Regulation (EU) 2022/2554 on Digital Operational Resilience for the Financial Sector (DORA)," December 14, 2022, https://eur-lex.europa.eu/eli/reg/2022/2554/oj

[1.1] European Insurance and Occupational Pensions Authority (EIOPA), "Digital Operational Resilience Act (DORA)" overview documentation. The 22,000+ figure includes banks, insurance companies, investment firms, and ICT third-party service providers subject to DORA requirements.

[2] DORA Article 28(3), "ICT Risk Management Framework - Register of Information," in Regulation (EU) 2022/2554

[3] DORA Article 29(2) and Article 30(3)(f), "Assessment of ICT Third-Party Risk and Key Contractual Provisions," in Regulation (EU) 2022/2554

[4] European Banking Authority, European Insurance and Occupational Pensions Authority, and European Securities and Markets Authority, "Final Report on Draft Regulatory Technical Standards," July 2024. The draft Article 5 requiring ongoing supply chain monitoring was removed in the final RTS published by the European Commission.

[5] American Institute of Certified Public Accountants, "2017 Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy," 2017

[6] American Institute of Certified Public Accountants, "2017 Trust Services Criteria (With Revised Points of Focus – 2022)," September 2022. The revision document explicitly states changes "do not, in any way, alter the criteria in the 2017 TSC."

[7] American Institute of Certified Public Accountants, "SOC for Supply Chain: An Examination Engagement to Report on a Subject Matter Related to the Products Produced by or Services Provided by a Producing Entity," March 2020

[7.1] Based on review of Big Four and major regional CPA firm service offerings (2024). SOC for Supply Chain engagements remain concentrated in manufacturing and distribution sectors per AICPA guidance; software vendor adoption remains limited.

[8] The White House, "Executive Order 14028: Improving the Nation's Cybersecurity," May 12, 2021, https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[9] OWASP Foundation, "CycloneDX Specification," https://cyclonedx.org/specification/overview/; Linux Foundation, "Software Package Data Exchange (SPDX) Specification," https://spdx.dev/specifications/

[10] Cybersecurity and Infrastructure Security Agency (CISA), "Secure Software Development Attestation Form," March 2024, https://www.cisa.gov/resources-tools/resources/secure-software-development-attestation-form

[11] Based on review of OCC Bulletins, FDIC Financial Institution Letters, Federal Reserve SR Letters, and FFIEC IT Examination Handbook updates through December 2025. While these agencies reference supply chain risk management principles, no guidance specifically mandates SBOM collection from vendors or CISA attestation verification for financial institutions. The June 2023 Interagency Guidance on Third-Party Relationships (OCC Bulletin 2023-17, Fed SR 23-4, FDIC FIL 29-2023) addresses supply chain risk conceptually but does not require SBOMs.

[11.1] SBOM generation challenges for complex environments documented in: NTIA Software Component Transparency, "SBOM Minimum Elements," noting that "SBOMs may require additional effort for legacy systems"; Sonatype "State of the Software Supply Chain" reports (2022-2024) discussing SBOM tooling gaps for legacy applications; and CISA "SBOM Sharing Lifecycle Report" (2024) noting "organizations with significant technical debt or acquired codebases face extended timelines for comprehensive SBOM coverage."

[12] Federal Financial Institutions Examination Council (FFIEC), "IT Examination Handbook: Outsourcing Technology Services," June 2023, Section "Third-Party Due Diligence and Monitoring." The handbook states: "Users of audit reports or reviews should not rely solely on the information contained in the report to verify the internal control environment of the TSP." Available at: https://ithandbook.ffiec.gov/

[13.1] Kaseya CEO Fred Voccola statement and industry reporting confirmed 50-60 MSPs directly affected and 800-1,500 downstream businesses. See: NPR, "Scale, Details Of Massive Kaseya Ransomware Attack Emerge," July 5, 2021; Reuters, "Up to 1,500 businesses affected by ransomware attack," July 5, 2021.

[13.2] 3CX company data and security researcher analysis. See: CPO Magazine, "Supply Chain Attack on VoIP Firm 3CX Puts 600,000 Businesses at Risk," March 2023; Krebs on Security, "3CX Breach Was a Double Supply Chain Compromise," April 2023.

Chapter 6: The Competency Gap

Even if TPRM frameworks mandated SBOM collection, dependency transparency, and supply chain attestation, a more fundamental challenge would emerge: the scope of TPRM has expanded into technical domains that were not part of vendor management a decade ago.

The mismatch is one of role evolution, not competency gaps. TPRM programs were built around contract analysis, financial risk assessment, operational resilience evaluation, and compliance verification. These capabilities remain essential. But the technology landscape has added new risk dimensions that require different expertise. Current TPRM professionals are adequate for traditional vendor management—what's missing is whether team compositions reflect the expanded scope of modern third-party risk.

Software supply chain analysis—dependency mapping, build pipeline security, code library vulnerability assessment—represents an adjacent discipline that has become relevant to TPRM only in the past decade. The integration of these skill sets is an organizational design challenge, not a professional development failure.

What TPRM Teams Are Built to Assess

The contemporary TPRM professional is educated and trained to evaluate:

- **Financial stability**: Does the vendor have adequate financial resources to continue operations?
- **Operational resilience**: Does the vendor have business continuity and disaster recovery plans?
- **Reputational risk**: Is the vendor involved in controversies that could harm the institution by association?
- **Legal and regulatory compliance**: Does the vendor comply with relevant laws and regulations?
- **Contractual protections**: Are service levels, liability limitations, termination rights, and audit rights adequate?
- **Information security governance**: Does the vendor have policies, training, incident response plans, and attestations?

These are the domains where TPRM programs add genuine value. They prevent institutions from engaging vendors who are financially unstable, operationally fragile, legally non-compliant, or contractually unfavorable. They ensure that vendors meet baseline information security standards as evidenced through SOC 2 reports, attestations, and questionnaire responses.

This expertise remains essential. The problem is that software supply chain risk exists in domains where this expertise does not translate.

The "Good on Paper" Solution and Why It Fails at Step Zero

The solution to nth-party dependency risk appears straightforward conceptually: Collect Software Bills of Materials from each critical SaaS vendor. Aggregate them into a centralized repository. Analyze convergence across the vendor portfolio. Identify concentrated dependencies. Prioritize monitoring accordingly.

The technology to execute this workflow exists. Commercial TPRM platforms could add SBOM ingestion and analysis modules. The technical barriers, while real for some organizations, are not insurmountable.

The solution fails at step zero: **most fintech SaaS vendors do not generate SBOMs because the market is not demanding them.**[6] TPRM teams cannot collect what vendors do not produce. (Chapter 9 examines market dynamics and multi-stakeholder solutions to this demand signal problem.)

The Competency Gap in Interpretation

Even if SBOMs existed and were collected systematically, interpreting them requires expertise that falls outside traditional TPRM training. A Software Bill of Materials is a technical artifact—meaningful to software engineers and security analysts, but opaque without that background.

A Software Bill of Materials for a modern web application contains hundreds or thousands of components. A typical entry might look like this (simplified CycloneDX format):

```
{
  "type": "library",
  "name": "notify",
  "group": "EventSMS",
  "version": "2.1.3",
  "purl": "pkg:npm/EventSMS/notify@2.1.3",
  "licenses": [{"license": {"id": "MIT"}}]
}
```

An SBOM contains dozens or hundreds of such entries. A TPRM analyst reviewing this list would see names like:

- express
- axios
- lodash
- moment
- jsonwebtoken
- bcrypt
- winston
- pg
- redis
- EventSMS/notify

Most of these are legitimate, widely used libraries. Express is a web framework. Axios handles HTTP requests. Lodash provides utility functions. Moment handles date/time operations. Each serves a clear purpose in modern web development.

But what is EventSMS/notify? Is it a legitimate third-party notification library? Is it a custom internal module the vendor wrote? Is it a typo or package name squat that could indicate malicious intent?

An experienced software developer would recognize that "EventSMS" as a package namespace is unusual—most legitimate libraries use descriptive names or developer handles. "Notify" is generic. The combination is a potential red flag worth investigating: Does this library have documentation? Who maintains it? How many weekly downloads does it have on npm? Are there known vulnerabilities?

The average TPRM analyst would not question it. Without software development context, "import

notify from EventSMS" looks like every other dependency in the list—technical details that serve some purpose the vendor's engineers determined was necessary.

Now consider the referenced scenario from the source documents: a developer who added a library that sends smartphone notifications whenever specific application events occur. This is a convenience for the developer—they get instant alerts when errors happen or when specific users log in. It is also a potential data exfiltration channel. Event details could include customer names, account identifiers, transaction amounts, or other sensitive information. The library makes network requests to external services the vendor has no contract with and the financial institution has no visibility into.

A security-trained software engineer would recognize this as a control gap warranting investigation. A TPRM analyst reviewing an SBOM would not even notice it as anomalous.

The Vulnerability Assessment Challenge

Even for well-known, legitimate libraries, assessing risk requires technical depth most TPRM teams do not possess.

When a new vulnerability is disclosed in a widely used library, TPRM teams need to answer:

1. Which of our vendors use this library?
2. Are they using a vulnerable version?
3. Is the vulnerable functionality actually exercised in the vendor's application, or is it unused code?
4. What is the exploitability in the vendor's specific deployment context?
5. What compensating controls exist (network segmentation, input validation, monitoring)?
6. What is the remediation timeline, and what customer action is required?

Questions 1 and 2 can be answered mechanically with SBOM data and vulnerability databases. Questions 3-6 require software engineering judgment. They require understanding not just that a library is present, but how it is used, what attack vectors exist, and what defenses are in place.

Consider Log4Shell as a concrete example. The vulnerability existed in Log4j versions 2.0-beta-9 through 2.14.1. Simply knowing a vendor uses Log4j is insufficient. You need to know:

- **Which version?** (Versions before 2.0-beta-9 were not vulnerable. Versions 2.15.0 and above had the vulnerability patched, though 2.15.0 itself had a bypass requiring 2.16.0 or 2.17.0 for complete mitigation.)[1.1]
- **Is it directly included or transitive?** (If transitive, which parent dependency brings it in? Updating the parent may be necessary.)
- **Where is it deployed?** (Internet-facing applications have higher exploitation risk than internal tools.)
- **What is logged?** (Applications that log user input are at higher risk than applications logging only system events.)
- **What mitigations are in place?** (Blocking specific JNDI lookup patterns, restricting outbound network traffic, applying virtual patches through WAF rules.)

Answering these questions requires software architecture knowledge, understanding of Java dependency management, familiarity with exploitation techniques, and ability to evaluate defense-in-depth controls. TPRM teams were not built for this—this is application security work.

And Log4j was a high-profile vulnerability with extensive public guidance, proof-of-concept exploits, and vendor advisories.[1] Most dependency vulnerabilities receive far less attention. CVE descriptions are technical and assume reader familiarity with the affected software. CVSS scores provide numeric severity ratings but do not account for deployment-specific mitigations. NVD data is often incomplete—64% of open source CVEs in 2025 had no CVSS score in the National Vulnerability Database.[2]

TPRM analysts tasked with dependency risk assessment would be operating in an environment where vulnerabilities are disclosed continuously, technical documentation is inconsistent, and vendor responses vary from immediate patching to prolonged delays. The skill required is not "read a report and check a box." It is "understand software architecture well enough to independently assess risk."

The Unfair Comparison

The source documents include a critical observation: "Your average programmer is rarely able to identify the vulnerabilities they are creating."[1.5]

This is true. Software development is complex. Even experienced engineers introduce security vulnerabilities through coding errors, design flaws, or incomplete understanding of attack vectors. Secure software development requires specialized training, code review, static analysis tooling, dynamic testing, and security-focused architectural review. Most development teams rely on security specialists to identify vulnerabilities that general-purpose developers miss.

If trained software engineers struggle to identify vulnerabilities in code they wrote, expecting TPRM analysts—who are not software engineers—to identify vulnerabilities in hundreds of third-party dependencies is unrealistic to the point of absurdity.

Yet this is precisely what a dependency-centric TPRM model would require without structural changes to team composition.

The Technical SME Gap in Context

The competency gap is not unique to TPRM. It reflects a broader pattern in risk management: as the domain being assessed becomes more technical, generalist risk assessment becomes inadequate.

Financial institutions have addressed this pattern in other domains:

- **Credit risk** for complex derivatives and structured products requires quantitative analysts with advanced mathematics and financial engineering backgrounds—not just credit analysts
- **Model risk management** for AI/ML systems requires data scientists who can evaluate model validity, training data quality, and algorithmic bias—not just model validators
- **Cybersecurity risk** evolved from IT audit functions to dedicated Chief Information Security Officer roles with technical depth in threat modeling, incident response, and security architecture

In each case, the evolution followed a predictable pattern: general risk assessment proved inadequate, incidents occurred, regulators increased expectations, and institutions built specialized capabilities.

TPRM is now at that inflection point for software supply chain risk. General vendor risk assessment—collecting questionnaires, reviewing SOC 2 reports, checking financial stability—is inadequate for evaluating dependency-level risks. The incidents have occurred (Log4j, SolarWinds, MOVEit, CrowdStrike). Regulators are increasing expectations (DORA, CISA guidance, FFIEC updates).

Institutions can build the specialized capability proactively or wait for regulatory enforcement to force the issue.

What Competency Actually Looks Like

The skill set needed to effectively manage software supply chain risk in a TPRM context includes:

Software development fundamentals: Understanding of programming concepts, dependency management, build processes, version control, and deployment pipelines. Not necessarily the ability to write production code, but enough familiarity to understand what developers are doing and why.

Application security knowledge: Familiarity with common vulnerability classes (injection, authentication bypass, cryptographic failures, supply chain compromise), attack vectors, and defensive controls. Ability to read CVE descriptions and assess exploitability in context.

Dependency ecosystem familiarity: Understanding of how package managers work (npm, Maven, PyPI, RubyGems), what transitive dependencies are, how version resolution works, and what software composition analysis tools do.

SBOM interpretation: Ability to read CycloneDX or SPDX formatted SBOMs, identify unusual or high-risk components, correlate SBOMs with vulnerability databases, and ask informed questions of vendors about their dependency management practices.

DevOps and CI/CD literacy: Understanding of how modern software is built and deployed—what CI/CD pipelines are, what artifact repositories do, how container registries work, what infrastructure-as-code means, and where security controls fit into these processes.

Open source ecosystem awareness: Familiarity with how open source software is developed, maintained, and funded. Understanding that "vendor" does not apply to most open source libraries. Awareness of the risks associated with unmaintained projects, single-maintainer dependencies, and malicious package injection.

This is not a list of skills that can be acquired through a two-day training course. It represents a fundamentally different professional background—one that bridges risk management and software engineering.

The Solution That Is Not a Solution

The competency gap cannot be solved by training existing TPRM teams to become software engineers. The skill development timeline is too long, the opportunity cost is too high, and the retention risk is significant. A TPRM analyst who develops deep technical skills becomes marketable for higher-paying technical roles and may leave for engineering or security positions.

Nor can the competency gap be solved by expecting vendors to provide dumbed-down explanations of their software supply chains. Vendors can produce SBOMs, but they cannot eliminate the technical complexity inherent in modern software. Dependency analysis requires technical judgment that vendors cannot provide as a service.

Commercial TPRM platforms cannot fully solve the gap either. Tools can automate SBOM ingestion, correlate components with vulnerability databases, and flag high-risk dependencies. But tools cannot provide the contextual judgment required to assess whether a flagged vulnerability is exploitable in a specific deployment, whether a mitigation is adequate, or whether an unusual dependency is benign or suspicious.

The competency gap requires a structural solution: **integrating technical subject matter expertise directly into TPRM teams.**

The Third-Party Technology Risk Analyst Role

Some institutions are addressing the competency gap by creating dedicated roles that bridge TPRM and technical functions. A Third-Party Technology Risk Analyst—or equivalent role embedded within TPRM programs—would:

- **Bridge TPRM and IT/security teams**: Act as translator between risk professionals who understand vendor management and technical professionals who understand software dependencies
- **Perform technical due diligence**: Evaluate SBOMs, assess dependency management practices, review CI/CD security controls, and analyze vendor security architecture
- **Support incident response**: When a critical dependency vulnerability is disclosed, lead vendor exposure identification, assess risk, and coordinate remediation
- **Provide technical consultation**: Assist TPRM analysts in interpreting technical findings from penetration tests, security assessments, and vendor attestations
- **Monitor dependency risk**: Track vulnerability disclosures affecting the vendor portfolio, identify concentration points, and report on emerging supply chain threats

Such roles typically command compensation above traditional TPRM analyst rates but below pure software engineering or security engineering positions—reflecting the hybrid skill set required.[3]

The scale of investment depends on portfolio complexity. Institutions with 100-500 third-party relationships may find a single dedicated analyst sufficient; larger organizations with thousands of vendors and complex technology stacks may require specialized teams.[3.1]

The Alternative: IT Resource Allocation

Not every institution can immediately create a dedicated Third-Party Technology Risk Analyst position. Headcount approval processes are slow, hiring takes time, and budget constraints are real.

The alternative is to allocate existing IT or information security resources to support TPRM on software supply chain issues. This could take the form of:

- A senior application security engineer assigned to spend 25-50% of their time supporting TPRM vendor assessments
- An IT architect designated as the TPRM technical liaison for infrastructure and application dependency questions
- Cross-functional working sessions where TPRM brings vendor assessment questions to IT/security subject matter experts on a recurring basis

This approach leverages existing technical capability without requiring new headcount. The disadvantages are that it divides attention (the IT resource has competing priorities), creates coordination overhead, and does not build dedicated TPRM-embedded technical expertise over time.

But it is better than the status quo, where TPRM teams assess software supply chain risk without technical capability and IT teams are brought in only after an incident has occurred.

Why This Is Not Optional

The most common objection to adding technical capability to TPRM teams is cost: "We can barely staff our current TPRM program. How can we add expensive technical resources?"

This objection misunderstands the risk landscape. Technology is not a peripheral concern for TPRM—it is the primary driver of third-party risk in financial services. The highest-risk vendors are SaaS applications, cloud platforms, security tools, payment processors, and core banking systems. Every one of these is a technology product or service. Assessing them without technical capability is assessing them inadequately.

The incidents validate this. Log4j caused emergency weekend response efforts across the industry.[4] SolarWinds led to federal investigations and heightened regulatory scrutiny. MOVEit resulted in breach notifications affecting millions of individuals and hundreds of institutions.[5] CrowdStrike created operational disruptions measurable in billions of dollars.

Each of these incidents was a software supply chain failure. Each exploited dependencies that TPRM programs could not see and would not have understood even if disclosed. Each demonstrated that entity-based risk assessment is insufficient for technology vendors.

The cost of adding technical capability to TPRM—whether through dedicated headcount, reallocated IT resources, or cross-functional collaboration—is a modest investment relative to the risk. The cost of a single major vendor incident—breach notification, regulatory scrutiny, customer remediation, reputational damage, operational disruption—can easily exceed $1 million for a regional institution and tens of millions for larger organizations.

The real question is not whether institutions can afford to integrate technical capability into TPRM—it is whether they can afford not to.

The Long-Term Trajectory

The need for technical expertise in TPRM is not a temporary gap that will close as frameworks mature or tools improve. It is a permanent feature of software supply chain risk management. Software will continue to grow more complex. Dependency chains will continue to deepen. Vulnerabilities will continue to be discovered in widely used components. Attackers will continue to target supply chains because they are effective attack vectors.

TPRM programs that do not build technical capability will fall further behind. They will continue to collect SOC 2 reports that address organizational controls but not the application-layer dependency risks that have emerged as most material. They will continue to send questionnaires that vendors answer perfunctorily. They will continue to learn about critical exposures only after incidents occur and vendors issue notifications.

TPRM programs that integrate technical capability will have competitive and operational advantages. They will identify vendor risks earlier. They will ask better questions during due diligence. They will respond faster when vulnerabilities are disclosed. They will provide more accurate risk assessments to executive leadership and boards.

Over time, regulators will likely formalize expectations that TPRM programs have technical competency for software supply chain risk. DORA already implies this through requirements for assessing "long or complex chains of subcontracting" and evaluating concentration risk. Future regulatory guidance may be more explicit.

Institutions that build the capability now will be ahead of the regulatory curve. Those that wait will be forced to build it under scrutiny and time pressure, which is more expensive and less effective.

The competency gap is not a side issue or a nice-to-have enhancement. It is the central challenge in modernizing TPRM for the software-intensive financial services landscape. No framework can close it. No tool can solve it. Only people with the right skills, embedded in the right organizational structure, can bridge the gap between what TPRM programs were built to see and what software supply chain risk actually looks like.

Theory, however, requires proof. The seven incidents examined in the next chapter provide that proof—demonstrating with painful clarity how dependency risks manifest in practice.

Endnotes - Chapter 6

[1] CISA Cyber Safety Review Board, "Review of the December 2021 Log4j Event," July 2022. The vulnerability received extensive public guidance, proof-of-concept exploits, and vendor advisories within days of disclosure in December 2021. Available at: https://www.cisa.gov/sites/default/files/publications/CSR Report-on-Log4-July-11-2022_508.pdf

[1.1] Apache Software Foundation, "Apache Log4j Security Vulnerabilities," detailing CVE-2021-44228 (Log4Shell) and subsequent patches through Log4j 2.17.1. NIST National Vulnerability Database CVE records. The vulnerability affected Log4j versions 2.0-beta-9 through 2.14.1; version 2.15.0 contained an incomplete fix requiring further patches to 2.16.0 and ultimately 2.17.0 for full remediation.

[1.5] This observation is commonly attributed to software security literature and practitioner experience. Similar findings appear in: Gary McGraw, "Software Security: Building Security In" (Addison-Wesley, 2006); OWASP Foundation, "Secure Coding Practices Quick Reference Guide"; and various academic studies on developer security awareness. The fundamental challenge—that creating secure code requires expertise most developers do not possess—is a consistent finding across software security research.

[2] Endor Labs, "State of Dependency Management 2024," noting data quality issues in vulnerability databases including incomplete CVSS scoring in NVD. See also: NIST National Vulnerability Database quality assessments.

[3] Compensation estimates based on industry salary surveys for hybrid risk/technology roles. Glassdoor, LinkedIn Salary Insights, and industry benchmarking data for comparable roles in financial services.

[3.1] Third-party vendor portfolio sizes vary significantly by institution size and business model. Industry surveys suggest regional banks ($5-50B in assets) typically maintain 100-500 third-party relationships total, with technology vendors (SaaS, cloud, security) representing approximately 30-40% of the portfolio. Larger institutions may have 1,000+ vendor relationships. Estimates derived from OCC examination data, FFIEC guidance on vendor management, and industry TPRM benchmarking surveys.

[4] CISA Cyber Safety Review Board, "Review of the December 2021 Log4j Event," July 2022. Financial services institutions spent weeks identifying exposure across vendor portfolios, with

organizations lacking SBOM capabilities requiring manual investigation that extended response timelines from hours to weeks.

[5] Emsisoft, "Unpacking the MOVEit Breach: Statistics and Analysis," updated December 2023. The CL0P ransomware group's exploitation of Progress Software's MOVEit Transfer platform affected 2,773+ organizations and 95.8 million individuals, with 60+ banks and credit unions confirmed affected. Most financial institutions were compromised through their vendors' use of MOVEit rather than direct implementation.

[6] SBOM generation rates among commercial SaaS vendors remain low. While Executive Order 14028 (May 2021) mandated SBOM provision for federal software suppliers, no equivalent requirement exists for commercial software serving financial services. Industry surveys support limited adoption: Synopsys 2024 OSSRA report found that while 96% of codebases contain open source, SBOM generation remains primarily regulatory-driven; Linux Foundation/OpenSSF "State of SBOM" (2024) reported that only 47% of organizations produce SBOMs for their software products. Practitioner surveys from Shared Assessments and industry TPRM working groups confirm that SBOM requests from financial institutions are frequently met with responses indicating vendors do not generate them.

Part III: Evidence from the Field

Part III: Evidence from the Field

Chapter 7: Seven Incidents, One Pattern

The theory is compelling: traditional TPRM frameworks organize risk around legal entities while real-world technology failures propagate through technical dependencies. But theory requires proof. Between 2020 and 2024, seven major incidents provided that proof—demonstrating with painful clarity that vendor questionnaires and SOC 2 reports were not designed to address the risks that have since emerged as most material, while contract provisions (indemnification clauses, SLAs, notification requirements) manage liability allocation but do not prevent or mitigate operational failures.

This section examines each incident not as isolated failures but as systematic demonstrations of framework inadequacy. The pattern is consistent: attackers and accidents exploit layers of the technology stack that existing frameworks were never designed to see.

Log4j (December 2021): The Transitive Dependency Blind Spot

On December 9, 2021, a vulnerability in Apache Log4j—a logging library used by the vast majority of Java applications globally[1]—achieved CVSS score 10.0 and the designation "one of the most serious vulnerabilities in the history of the internet."[2] Within nine minutes of public disclosure, exploitation began.[3] Within 72 hours, over 800,000 attack attempts were detected.[4] As of late 2024, 12% of Java applications still run vulnerable versions.[5]

What happened technically: CVE-2021-44228 allowed unauthenticated remote code execution through JNDI lookup functionality introduced in 2013.[6] Attackers could execute arbitrary code by including a specially crafted string in any logged message—including HTTP headers, usernames, or error messages. The vulnerability existed for eight years, maintained by 16 unpaid volunteers,[7] before weaponization.

The dependency angle: According to Google Open Source Insights, 17,000+ packages in Maven Central were affected.[8] Critically, 60% of impacted Java projects used Log4j as a *transitive dependency*—meaning they did not directly include Log4j but inherited it through other libraries.[9] Snyk confirmed: "A straightforward search to determine if you're using a vulnerable version of Log4j would not necessarily find all occurrences in your projects."[10]

Organizations without Software Bills of Materials (SBOMs) spent weeks manually examining applications to identify exposure. According to analysis from security researchers and incident response practitioners, "Organizations without SBOM capability often had to engage in time consuming manual searches and risked remaining vulnerable. Organizations with SBOMs were able to report a relatively straightforward and efficient response. Once the vulnerability was discovered,

organizations spent weeks, and in some cases even months, trying to determine where and how they had been exposed. The difficulty could have been avoided had they had a SBOM to provide this information, enabling organizations to identify and mitigate exposure within hours instead of weeks."[11]

Why frameworks did not detect it: SOC 2 audits assess whether vendors *have* vulnerability management processes, not whether they maintain component inventories. Standard vendor questionnaires ask "Do you patch critical vulnerabilities?" but do not require disclosure of transitive dependencies. The vulnerability existed in a library most organizations did not know they used, provided by a volunteer project with no contractual relationship to anyone. Traditional TPRM focuses on commercial vendors with contracts and SLAs—open source dependencies like Log4j are "suppliers" without either.

The U.S. Department of Homeland Security estimates it will take at least a decade to find and fix every vulnerable Log4j instance.[12] This is not a vendor that can be assessed—it is a dependency that can only be inventoried.

SolarWinds (December 2020): Build Process Integrity Failure

Between November 2019 and December 2020, Russian SVR operators (APT29/UNC2452) maintained persistent access to SolarWinds' development environment, ultimately compromising the build servers for Orion network management software. The attack affected 18,000+ organizations including nine federal agencies.[13] Attackers had 14+ months of undetected access.[14]

What happened technically: Attackers deployed SUNSPOT malware on build servers that monitored for Orion compilation processes. During builds, SUNSPOT temporarily replaced legitimate source code (`InventoryManager.cs`) with backdoored versions, waited for compilation to complete, then restored the original file.[15] The resulting malicious DLL (`SolarWinds.Orion.Core.BusinessLayer.dll`) was digitally signed with SolarWinds' legitimate code-signing certificate and distributed through official update channels.[16]

The malicious code created a backdoor named SUNBURST that established command-and-control communication through DNS beaconing to attacker-controlled infrastructure, disguised as normal Orion update checks.[17] Of the 18,000 affected organizations, attackers selectively activated the backdoor for approximately 100-250 high-value targets including the Departments of Treasury, Commerce, Homeland Security, Justice, Energy, and State.[18]

The dependency angle: Organizations using Orion had conducted standard vendor assessments of SolarWinds. Many had SOC 2 Type II reports. SolarWinds had undergone security audits and certifications. None of this prevented or detected the compromise. The attack succeeded because it bypassed all assessment layers by compromising the build environment—a system component that exists upstream of everything traditional TPRM evaluates.

The nation-state reality: It is important to acknowledge that this was a sophisticated intelligence operation by Russia's SVR with essentially unlimited resources and patience. The attackers maintained access for over 14 months while evading detection by security-conscious organizations including the U.S. government. Better auditing or enhanced due diligence almost certainly would not have detected this attack. The lesson is not that TPRM programs failed—it is that software delivery mechanisms represent a dependency category that traditional frameworks do not address, and that well-resourced nation-state adversaries will find and exploit such gaps regardless of how robust individual vendor security programs appear.

The attack exploited what one analysis called a "millisecond window"—the brief period during compilation when source code is transformed into executable binaries.[19] Even if customers had reviewed SolarWinds' source code (they could not—it is proprietary), the malicious code was not in the repository. It existed only transiently during builds.

Why frameworks did not detect it: SOC 2 evaluates controls over IT systems, not build pipeline integrity. Questionnaires ask about change management and code review but not about build environment isolation or artifact verification. Penetration tests assess production systems, not development infrastructure. The "solarwinds123" password incident—where an intern set a weak password on a production FTP server that remained unchanged for 2.5 years and was publicly exposed on GitHub for 17 months[20]—revealed concerning security practices that compliant-looking policies masked.

Digital signatures confirmed the malware was "legitimately" from SolarWinds because attackers used SolarWinds' own signing infrastructure. Code signing provides *authenticity* (this came from SolarWinds) and *integrity in transit* (it was not modified after signing)—but cannot provide *build process integrity* (it matches reviewed source code). As one analysis noted, "Having a valid digital signature does not mean that the data itself is legitimate and was not compromised before the signature was generated and applied."[21]

CrowdStrike (July 2024): Automatic Update Concentration Risk

On July 19, 2024, at 04:09 UTC, CrowdStrike distributed a faulty sensor configuration update through its Falcon endpoint security platform. Within 78 minutes, approximately 8.5 million Windows devices crashed simultaneously,[22] triggering the largest IT outage in history with estimated costs exceeding $10 billion globally.[23]

What happened technically: The Falcon sensor uses two update mechanisms—traditional software updates (Sensor Content) which undergo rigorous testing and customer-controlled deployment, and rapid threat detection updates (Rapid Response Content) which deploy automatically to respond to emerging threats.[24] A Rapid Response Content update included a configuration file (Channel File 291) containing a template with 21 input parameters, but the sensor's Content Interpreter code only provided 20 values.[25] When the interpreter attempted to evaluate the 21st parameter (which used non-wildcard matching criteria), it triggered an out-of-bounds memory read, causing a Windows kernel panic and Blue Screen of Death.[26]

The update affected only systems running Falcon sensor version 7.11+ on Windows that were online during the 78-minute distribution window. Yet this narrow window was sufficient to crash millions of devices across airlines, banks, hospitals, retailers, and critical infrastructure globally.[27]

The dependency angle: Organizations deployed CrowdStrike specifically because it is best-in-class endpoint protection. According to LinkedIn data, approximately 60% of Fortune 500 companies (298 of 500) trust CrowdStrike.[28] This concentration—driven by rational "best of breed" procurement decisions—created systemic risk. When the single best security tool failed, it failed everywhere simultaneously.

The security paradox: the tool meant to protect became the single point of failure. Organizations had granted Falcon administrative privileges and automatic update authority because effective endpoint protection requires both. When that privilege was combined with a faulty update that bypassed testing, there was no defense.

Why frameworks did not detect it: Vendor assessments confirmed CrowdStrike maintains sophisticated security operations, incident response capabilities, and quality assurance processes. The company's operational track record was exemplary. But no framework addressed:

- Pre-deployment testing requirements for security tool configuration updates
- Customer control over rapid response content deployment timing
- Staged rollout mandates for updates affecting kernel-level code
- Concentration risk assessment when a single vendor protects >50% of an industry

The recovery challenge demonstrated another gap: systems crashed before network connectivity, making remote recovery impossible. Organizations needed physical access to each device to boot into Safe Mode and delete the problematic file. For BitLocker-encrypted systems, this required manually entering unique 48-digit recovery keys.[29] The manual touch-every-machine recovery process took days to weeks, with some organizations still recovering systems months later.[30]

MOVEit (May-June 2023): The nth-Party Cascade

Between May 27 and June 5, 2023, the Cl0p ransomware group exploited a zero-day SQL injection vulnerability in Progress Software's MOVEit Transfer managed file transfer (MFT) platform. The campaign ultimately affected 2,773+ organizations and exposed 95.8 million individuals' data,[31] with estimated costs reaching $15.8 billion.[32]

What happened technically: Cl0p had researched the vulnerability since 2021, conducting reconnaissance and testing exploitation techniques over two years before mass deployment.[33] On Memorial Day weekend 2023—chosen strategically when IT security staffing is reduced[34]—they deployed the LEMURLOOT web shell to internet-facing MOVEit Transfer servers. The web shell enabled attackers to bypass authentication, access underlying databases, and exfiltrate sensitive data including names, Social Security numbers, financial information, and health records.[35]

The attack was sophisticated in its staging: deploy LEMURLOOT broadly to gather reconnaissance data, then selectively target high-value victims rather than deploying ransomware universally. Cl0p focused primarily on data theft and extortion rather than encryption, allowing faster operations with less detection risk.[36]

The dependency angle: According to Emsisoft's analysis, "For almost two-thirds of MOVEit victims, breaches occurred because their third-party vendors used MOVEit—or their vendor's vendors used the file-transfer service."[37] The most extensively documented cascade began with Pension Benefit Information (PBI Research Services), a death audit and participant location service provider. When Cl0p compromised PBI's MOVEit environment:

- PBI directly impacted: 1,209,825 individuals[38]
- Milliman (an actuarial firm affected through PBI) exposed records at 200+ additional organizations[39]
- TIAA (affected through PBI) exposed data from 60+ organizations[40]
- PBI's single compromise ultimately affected 354+ additional organizations[41]

The dependency chains looked like: **Bank → Retirement Plan Administrator (TIAA) → Death Audit Service (PBI) → MOVEit Transfer**. Banks had vendor relationships with retirement administrators. Those administrators used PBI. PBI used MOVEit. Banks' data was exposed through a file transfer platform they'd never heard of, provided by a company they had no relationship with.

Colorado State University exemplifies multiple exposure pathways: the university was breached six times by six different vendors (TIAA, National Student Clearinghouse, Corebridge Financial, Genworth Financial, Sunlife, and The Hartford)—each using MOVEit independently.[42] Single organizations experienced multiple distinct exposures because different vendor relationships converged on the same underlying infrastructure.

Why frameworks did not detect it: Banks that assessed TIAA received clean SOC 2 reports and vendor questionnaires. TIAA's relationship with PBI may have been disclosed as a subcontractor. But PBI's use of MOVEit Transfer for file handling was a technical implementation detail, not a vendor relationship requiring disclosure. Even if disclosed, banks had no contractual leverage to assess MOVEit is security posture or require Progress Software to implement specific controls.

The notification disaster compounded the problem: many organizations learned they were affected from news media or Cl0p's leak site rather than from their vendors.[43] Some received forensic details two months after initial exploitation,[44] violating regulatory notification requirements and preventing timely incident response. One breach notification letter documented: unauthorized access May 29-31, company learned July 12 (six weeks later), forensic report provided July 25 (nearly two months later).

Kaseya VSA (July 2021): MSP Business Model as Attack Multiplier

On July 2, 2021—the Friday before July 4th weekend—the REvil ransomware group exploited zero-day vulnerabilities in Kaseya's VSA (Virtual System Administrator) remote monitoring and management platform. The attack affected 50-60 managed service providers and between 800-1,500 downstream businesses across 17 countries,[45] with REvil demanding $70 million for a universal decryption key.[46]

What happened technically: REvil exploited CVE-2021-30116 (authentication bypass) and CVE-2021-30117 (SQL injection) that Dutch researchers had responsibly disclosed to Kaseya on April 6, 2021.[47] Despite 87 days between disclosure and attack, on-premises VSA servers remained unpatched. Attackers extracted agent credentials from installer files, used them to obtain authenticated session tokens, escalated privileges through SQL injection, and deployed malicious "updates" through VSA's legitimate software distribution functionality.[48]

The malicious update disabled Microsoft Defender, dropped a vulnerable Microsoft binary, used DLL side-loading to execute REvil ransomware, and encrypted victim systems with SYSTEM-level privileges.[49] Because the ransomware deployed through the trusted RMM platform, it appeared to victim systems as legitimate IT management activity.

The dependency angle: The MSP business model created an attack multiplier effect. Compromising 50-60 MSPs instantly provided access to their collective 800-1,500 clients.[50] The average force multiplication: 15-25x. Each MSP manages IT infrastructure for dozens to hundreds of small businesses—dental practices, architecture firms, libraries, retailers—that cannot afford dedicated IT staff.

Coop Sweden illustrates the dependency chain: Coop (grocery chain) → Visma EssCom (payment systems supplier) → Kaseya VSA (RMM tool). When VSA was compromised, ransomware deployed to Visma EssCom's systems, which encrypted Coop's point-of-sale infrastructure. Result: 800 grocery stores closed because cash registers could not process payments.[51] Coop had no direct relationship with Kaseya and no visibility into their payment provider's IT management tools.

The "one throat to choke" vendor consolidation strategy that drives MSP adoption also creates the visibility gap. As industry guidance explains: "Most MSPs advertise that they will take full responsibility for the problem and get it solved for you. You do not need to call multiple vendors... you now have 'one throat to choke.' "[52] This arrangement means clients have no contractual visibility into the specific RMM platforms, backup solutions, or security tools their MSP employs.

Why frameworks did not detect it: Businesses using MSPs conduct vendor assessments of the MSP—reviewing their security policies, certifications, and incident response capabilities. But the MSP's choice of RMM platform is an operational decision, not typically disclosed or assessed. Even if disclosed, small businesses lack leverage to require MSPs to use alternative platforms or undergo additional security assessments.

The concentrated RMM market amplifies risk: as of Q4 2024, just three vendors (Kaseya 25.9%, ConnectWise 25.4%, NinjaOne rising to #3) command over 75% market share.[53] This concentration exists because RMM platforms require significant investment in training, scripting, and automation—switching costs are substantial. When a dominant platform is compromised, the impact propagates across thousands of MSPs and tens of thousands of end organizations.

3CX (March 2023): The First Cascading Supply Chain Attack

In March 2023, North Korean state-sponsored actors (Lazarus Group/UNC4736) executed the first documented "supply chain of a supply chain" attack. Attackers compromised Trading Technologies' decommissioned X_TRADER software, which led to compromise of a 3CX employee's personal computer, which enabled access to 3CX's build environment, which resulted in distribution of trojanized VoIP software to hundreds of thousands of organizations with 12 million+ users.[54]

What happened technically: Between November 2021 and July 2022, fewer than 100 people downloaded X_TRADER from Trading Technologies' website. The software had been officially decommissioned in April 2020 but remained available for download.[55] In April 2022, a 3CX employee downloaded the trojanized X_TRADER installer on a personal computer. The malware (VEILEDSIGNAL) stole the employee's 3CX corporate credentials.[56] Two days later, attackers used those credentials via VPN to access 3CX's corporate network.[57]

After months of lateral movement, attackers compromised both Windows and macOS build servers in late 2022. They modified two DLLs in the 3CX DesktopApp: `ffmpeg.dll` (completely replaced) and `d3dcompiler_47.dll` (trojanized with appended encrypted shellcode).[58] On March 3 and March 13, 2023, trojanized builds were signed with legitimate 3CX certificates and distributed to customers.[59]

The malicious payload executed a sophisticated multi-stage attack: 7-day dormancy period to evade sandbox analysis,[60] ICONICSTEALER malware to gather browser history and credentials from all infected systems,[61] and selective deployment of GOPURAM backdoor to fewer than 10 cryptocurrency companies identified as high-value targets through the reconnaissance data.[62]

The dependency angle: The attack chain defies traditional vendor mapping:

- **Level 1**: Your organization uses 3CX VoIP software
- **Level 2**: 3CX (your vendor) had an employee who
- **Level 3**: Downloaded software from Trading Technologies (not a vendor, no business relationship)
- **Level 4**: X_TRADER (decommissioned legacy software, officially unsupported)
- **Level 5**: Compromised by UNC4736/Lazarus Group (nation-state attacker)

3CX was the victim, not a negligent vendor. Trading Technologies had no business relationship with 3CX and no way to know a 3CX employee downloaded their software on a personal device.[63] The employee downloaded unrelated financial trading software on a personal computer—completely outside 3CX's corporate IT visibility.

Why frameworks did not detect it: Organizations that assessed 3CX received SOC 2 reports covering 3CX's operational controls. Those reports could not reveal:

- That a 3CX employee had downloaded compromised trading software on a personal device
- That Trading Technologies' decommissioned software was serving malware
- That 3CX's build environment had been compromised for months
- That distributed updates contained sophisticated backdoors signed with legitimate certificates

The "week that mattered" compounded the failure: customers began reporting EDR alerts on March 22. 3CX initially dismissed these as false positives, with support staff recommending customers whitelist the application.[64] CrowdStrike publicly disclosed the compromise on March 29. 3CX officially acknowledged it March 30—eight days during which organizations actively disabled the security controls that were correctly detecting the threat.[65]

The fundamental challenge: even perfect vendor assessment cannot protect against threats originating from your vendor's vendor's decommissioned legacy software downloaded by an employee on a personal device. As Coalition Inc.'s retrospective noted, this demonstrates "complete supply chain visibility is practically impossible."[66]

xz Utils (March 2024): Patient Adversary Infiltration

In early 2024, the xz Utils backdoor (CVE-2024-3094) was uncovered—what security researchers immediately recognized as the most sophisticated open source supply chain attack ever documented.[67] Unlike Log4j (accidental vulnerability), SolarWinds (external compromise of build systems), or MOVEit (exploitation of vendor software), the xz Utils incident demonstrated a fundamentally different attack pattern: **deliberate infiltration through patient social engineering to gain insider access**.

This incident deserves prominence because it proves that even perfect vendor due diligence cannot detect nation-state-level adversaries willing to invest years building trust before attacking. The OpenSSF and OpenJS Foundation issued warnings in April 2024 that this attack pattern "may not be an isolated incident" and similar infiltrations may already exist undetected in other critical open source projects.[68]

What happened technically: October 2021: An individual using the handle "Jia Tan" began contributing to xz Utils, a widely-used data compression library found in virtually every Linux distribution. The initial contributions were legitimate—bug fixes, performance improvements, documentation updates. Over the following months, Jia Tan established a pattern of helpful, technically competent participation.[69]

2022-2023: Multiple sockpuppet accounts—likely controlled by the same actor or coordinated team—began pressuring the original maintainer, Lasse Collin, to accept more help. These accounts referenced Collin's public statements about experiencing burnout and struggling to keep up with maintenance demands. The pressure campaign was methodical and sophisticated, designed to appear as organic community concern about project sustainability.[70]

November 2022: Jia Tan was granted commit access to the xz Utils repository. The promotion

appeared reasonable—Jia Tan had demonstrated technical competence through months of legitimate contributions and multiple community members had vouched for bringing on additional maintainers.[71]

January 2024: Jia Tan gained control of the project's website and infrastructure. By this point, Jia Tan was trusted as a core maintainer with broad responsibilities across the project.[72]

February-March 2024: Sophisticated malicious code was inserted into xz Utils versions 5.6.0 and 5.6.1. The backdoor was designed to compromise OpenSSH sshd authentication, enabling remote code execution on systems running the affected versions. The implementation was technically remarkable—the malicious code was hidden in binary test files and obfuscated build scripts, making it nearly impossible to detect through standard code review.[73]

The backdoor targeted specifically the SSH daemon (sshd) used for remote system access. If undetected, it would have enabled attackers to bypass authentication and execute arbitrary code with root privileges on millions of Linux systems globally. Every major distribution—Debian, Ubuntu, Fedora, Red Hat, Arch—was either already affected or would have been within weeks as the compromised version propagated through their update channels.[74]

March 29, 2024: Andres Freund, a Microsoft engineer and PostgreSQL developer, noticed unusual CPU usage and behavioral anomalies during routine performance testing. The symptoms were subtle—slight delays in SSH connections, marginally increased CPU consumption during authentication. Freund's investigation traced the anomaly to the recently updated xz library.[75]

This discovery was entirely accidental. The backdoor was not detected by code review of pull requests (the malicious code was hidden in binary test files), automated vulnerability scanning (no known vulnerability signatures existed), security audits (xz Utils had no formal security audit program), or supply chain security tools (the maintainer credentials were legitimate).[76]

If not for Freund's curiosity about minor performance degradation—the kind of anomaly most engineers would ignore or attribute to normal system variance—the backdoor would likely have achieved widespread deployment. The timing suggests attackers intended the compromised version to be included in the upcoming Debian stable release (scheduled for mid-2024), which would have embedded the backdoor in one of the most widely-deployed Linux distributions for years.[77]

The dependency angle: The xz Utils attack represents a qualitatively different threat model than other supply chain incidents. Key distinctions include insider access vs. external compromise: SolarWinds attackers broke into build servers. Jia Tan was granted legitimate maintainer access through social engineering. From a security perspective, Jia Tan was an insider, not an intruder. All commits were properly authenticated. All code signing used legitimate credentials. All project governance processes were followed.[78]

Years of investment: Most cyberattacks operate on timescales of days to months. The xz Utils infiltration required 2.5 years of patient work—building reputation, earning trust, waiting for the right moment. This demonstrates nation-state-level commitment and sophistication. As security researcher Dan Lorenc noted, "This might be the best executed supply chain attack we've seen described in the open, and it is a nightmare scenario."[79]

Targeting volunteer-maintained infrastructure: xz Utils is maintained by unpaid volunteers. Lasse Collin, the original maintainer, was managing the project alone while dealing with burnout—a common situation in open source. The attackers exploited this vulnerability not through technical means but through social manipulation. They offered help to an overworked maintainer and used community pressure to make refusing that help appear unreasonable.[80]

The dependency is ubiquitous but invisible: According to census of open source usage, xz Utils is present in every major Linux distribution (Debian, Ubuntu, Fedora, Red Hat Enterprise Linux, Arch, openSUSE), container base images (affecting millions of containerized applications), cloud VM images from AWS, Azure, Google Cloud, embedded systems and network appliances, and virtually any system that handles compressed archives.[81] Yet no TPRM program inventories xz Utils as a dependency. It's not a vendor. It has no SOC 2 report. It has no commercial entity behind it. It's maintained by volunteers whom millions of organizations depend on but have no relationship with.

nth-party depth exceeds visibility: The dependency chain for a typical financial institution might look like: **Bank → SaaS Vendor → Cloud Provider (AWS/Azure) → Base Linux Distribution (Ubuntu/RHEL) → xz Utils (volunteer-maintained library)**. The bank assesses the SaaS vendor (third-party). The vendor discloses AWS in their SOC 2 (fourth-party). The cloud provider's Linux distribution is not disclosed (fifth-party). xz Utils is several layers beyond that (sixth-party+). Traditional TPRM sees three layers: vendor, cloud infrastructure, maybe the OS. The actual dependency chain extends far deeper.

Why frameworks did not detect it: The xz Utils attack exposed framework limitations that are architectural, not procedural. SOC 2 cannot audit social engineering over 2.5 years: Vendor security assessments evaluate controls at a point in time. They verify that change management processes exist, that commit access is controlled, that code review is performed. All of these controls existed for xz Utils. Jia Tan followed every process correctly because Jia Tan had legitimate access. The compromise was the granting of that access through patient manipulation, not a control failure after access was granted.[82]

Vendor due diligence cannot assess volunteer maintainer burnout: TPRM questionnaires ask commercial vendors about employee training, access controls, separation of duties, and incident response capabilities. These questions are meaningless for volunteer-maintained open source projects. Lasse Collin was not an employee—he was an unpaid maintainer managing a critical infrastructure component in his spare time. The attack exploited the structural vulnerability of open source sustainability, not a technical security gap.[83]

No contractual relationship = no assessment leverage: Financial institutions cannot audit Apache, Linux kernel developers, or xz Utils maintainers because there is no contract. You cannot send a vendor questionnaire to a volunteer. You cannot require SOC 2 certification from a GitHub repository. You cannot demand incident response SLAs from someone maintaining software for free. Yet your entire technology stack depends on these dependencies.[84]

Binary transparency gaps: The malicious code was hidden in binary test files (`.xz` compressed data) included in the repository. Code review focused on source code changes missed the backdoor because reviewers do not manually examine binary test data—they trust that test files are benign. This is a reasonable assumption that attackers exploited.[85] The build process further obfuscated detection: the backdoor activated only when specific build conditions were met (GNU/Linux, x86-64 architecture, glibc, systemd, GCC toolchain). In other environments, the malicious code remained dormant. This meant testing in development environments (often macOS or different Linux configurations) would not trigger suspicious behavior.[86]

Detection required forensic-level investigation: Andres Freund's discovery process involved noticing 500ms SSH connection delay (most engineers would ignore this), profiling to identify xz library as the cause, reverse-engineering obfuscated backdoor code hidden in binary files, and recognizing the pattern as deliberately malicious rather than a performance bug. This level of investigation is far beyond standard vulnerability scanning or security monitoring. It required deep technical expertise,

specialized tools, significant time investment, and—critically—noticing an anomaly subtle enough that 99% of engineers would dismiss it as normal system variance.[87]

What makes this uniquely concerning is that the attack vector is reusable: Jia Tan's approach—build credibility over years, exploit maintainer burnout, use social engineering to gain access—is applicable to thousands of open source projects. According to analysis from Tidelift and the Linux Foundation, 60% of critical open source infrastructure is maintained by 1-2 individuals, many experiencing burnout.[88] Each represents a potential target for this attack pattern.

It may not be isolated: The OpenSSF (Open Source Security Foundation) and OpenJS Foundation issued joint guidance warning that "the xz Utils backdoor may not be an isolated incident" and that similar social engineering campaigns may already be underway targeting other critical projects.[89] Security researchers have begun re-examining commit histories of other infrastructure libraries, looking for similar patterns of new maintainers gaining access during original maintainer burnout, sockpuppet accounts applying pressure to accept help, long periods of legitimate contributions before any malicious activity, and sophisticated technical knowledge suggesting state-sponsored resources.

Defenders must trust but verify—with no good verification method: Open source depends on trust. Maintainers must trust contributors. Distributions must trust upstream maintainers. Organizations must trust distributions. The xz Utils attack shows patient adversaries can build trust systematically to exploit these chains. The industry has no good answer for how to verify trust at scale across thousands of volunteer-maintained dependencies.[90]

Nation-state adversaries have validated the approach: Intelligence agencies monitor security research carefully. The xz Utils attack demonstrated that patient social engineering can compromise even security-conscious projects, volunteer-maintained infrastructure is vulnerable to burnout-based manipulation, binary obfuscation can evade code review for months, and the attack was within hours of succeeding globally before accidental discovery. This proof-of-concept will inform future attacks. As security researcher Filippo Valsorda noted, "This is the sort of attack that intelligence agencies have been capable of for years but we've never seen documented in the open. Now that it is public, expect variations."[91]

TPRM implications: Dependency mapping must extend beyond vendor relationships: Financial institutions need capability to answer "Are we exposed to xz Utils?" as quickly as "Which vendors do we use?" This requires Software Bill of Materials (SBOM) for all critical applications, recursive dependency mapping (not just direct imports), continuous monitoring of vulnerability disclosures in transitive dependencies, and automated tooling to identify affected systems when vulnerabilities are disclosed.[92]

Open source dependencies are critical infrastructure: The industry treats commercial vendors as manageable risks (contracts, assessments, leverage) while treating open source as "free" and therefore not requiring risk management. The xz Utils incident proves this is backwards. The most critical dependencies—the ones everything else builds on—are often volunteer-maintained open source projects with no security budget, no formal governance, and no accountability structure.[93]

Vendor assessments should include software supply chain maturity: When assessing SaaS vendors, TPRM programs should evaluate whether the vendor maintains SBOMs for their products, how they monitor for vulnerabilities in dependencies, what is their process for responding to incidents like Log4j or xz Utils, whether they have capability to identify affected systems within hours vs. weeks, and how deep is their dependency visibility (direct vs. transitive). These questions address risk management capability, not just compliance checkbox completion.[94]

Industry coordination is required: Individual organizations cannot assess every open source maintainer. The solution requires industry funding for critical open source infrastructure (Google, Microsoft, Amazon have begun this through OpenSSF), shared intelligence on suspicious contribution patterns, coordinated security audits of high-impact libraries, standards for secure open source development practices, and support systems for burned-out maintainers (so accepting help from strangers is not the only option).[95]

After the xz Utils disclosure, security teams at financial institutions asked their vendors: "Are you affected by CVE-2024-3094?" Many vendors responded: "We're investigating." The correct answer required knowing which systems and containers use Linux distributions, identifying which distribution versions include xz Utils 5.6.0/5.6.1, determining whether those systems run sshd, assessing whether the vulnerable code path is reachable, and understanding build environment configurations (the backdoor activated conditionally). Organizations with mature software supply chain visibility answered within hours. Organizations without visibility spent days or weeks investigating. Some still do not know definitively.

This is the same pattern as Log4j in 2021. Three years later, the fundamental problem remains unsolved: TPRM programs built to assess vendors cannot answer questions about dependencies. The xz Utils attack adds a new dimension: those dependencies may be compromised not through accidental vulnerabilities or external attacks, but through patient adversaries who spend years earning the trust required to attack from within. No vendor questionnaire can detect that. No SOC 2 audit can prevent it. No contract can indemnify against it.

The question is whether the industry will treat the xz Utils near-miss as a warning that requires fundamental change, or as an anomaly that can be managed through existing frameworks. The early signs suggest the latter. That choice may define the next decade of supply chain security risk.

These seven incidents provide systematic evidence that current frameworks cannot see the layers where risk propagates. The next chapter extracts the patterns from this evidence—mapping precisely where framework scope ends and new risks begin.

Endnotes - Chapter 7

[1] Impact estimates vary by methodology. CISA Cyber Safety Review Board, "Review of the December 2021 Log4j Event," July 2022, documented that 35,000+ Java packages (8% of Maven Central) were affected. Industry analyses from Contrast Security and others estimated 60-64% of Java applications contained the vulnerability. The exact percentage depends on methodology (analyzing packages vs. deployed applications vs. vulnerable code paths), but all sources agree the impact was extraordinarily widespread. Available at: https://www.cisa.gov/sites/default/files/publications/CSRB-Report-on-Log4-July-11-2022_508.pdf

[2] Jen Easterly, CISA Director, quoted in "Apache Log4j Vulnerability Guidance," CISA News and Events, December 10, 2021.

[3] Matthew Prince, Cloudflare CEO, "Exploitation of CVE-2021-44228 Before Public Disclosure and Evolution of WAF Evasion Patterns," *Cloudflare Blog*, December 2021.

[4] Check Point Software, "Log4j Downloads Show Supply Chain Wake-Up Call Ignored," *DeveloperTech*, December 2021.

[5] "Log4j Downloads Show Supply Chain Wake-Up Call Ignored," *Developer-Tech*, December 2024.

[6] Apache Software Foundation, LOG4J2-313, July 18, 2013; "Inside the Log4j2 Vulnerability (CVE-2021-44228)," *Cloudflare Blog*, December 2021.

[7] Volkan Yazici, Apache Log4j maintainer, quoted in "Inside the Breach That Broke the Internet: The Untold Story of Log4Shell," *GitHub Blog*, Open Source, 2021; "Log4j: A New Chapter with STF Funding," *Apache Logging Blog*, December 14, 2023.

[8] Google Open Source Insights, "Dependencies of Dependencies: The Critical Challenge of Managing Software Supply Chain Risk," *ISMS.online*, 2021.

[9] "Dependencies of Dependencies: The Critical Challenge of Managing Software Supply Chain Risk," *ISMS.online*, Information Security, 2021.

[10] Snyk, quoted in "Navigating Cybersecurity with SBOMs: Lessons from Log4j and MOVEit," *LinkedIn Pulse*, Jack Lilley, 2023.

[11] Analysis compiled from: Jack Lilley (security researcher), "Navigating Cybersecurity with SBOMs: Lessons from Log4j and MOVEit," *LinkedIn Pulse*, 2023; ReversingLabs (software supply chain security firm), "Log4j: Why Your Organization Needs SBOM," *ReversingLabs Blog*, December 2021; CISA, "Review of the December 2021 Log4j Event," CSRB Report, July 2022. These sources document the substantial difference in incident response timelines between organizations with and without SBOM capability during the Log4j crisis.

[12] U.S. Department of Homeland Security, estimate documented in "Log4j Vulnerability Timeline," *Automox Blog*, 2021; "The Apache Log4j Vulnerabilities: A Timeline," *CSO Online*, December 2021.

[13] U.S. Senate Republican Policy Committee, "The SolarWinds Cyberattack," Policy Papers, 2021; U.S. Government Accountability Office, "SolarWinds Cyberattack Demands Significant Federal and Private-Sector Response," *GAO Blog*, Infographic, 2021.

[14] Palo Alto Networks Unit 42, "SolarStorm Timeline: Details of the Software Supply-Chain Attack," Unit 42 Blog, 2021; SecurityWeek, "SolarWinds Shares More Information on Cyberattack Impact, Initial Access Vector," December 2020.

[15] CrowdStrike, "SUNSPOT Malware: A Technical Analysis," CrowdStrike Blog, 2021; Bleeping Computer, "New Sunspot Malware Found While Investigating SolarWinds Hack," February 2021.

[16] Mandiant, "SUNBURST Additional Technical Details," Mandiant Resources Blog, December 2020.

[17] Security Boulevard, "SolarWinds/SUNBURST: DGA or DNS Tunneling?" December 2020; Varonis, "SolarWinds SUNBURST Backdoor: Inside the Stealthy APT Campaign," December 2020.

[18] U.S. Government Accountability Office, "SolarWinds Cyberattack Demands Significant Federal and Private-Sector Response," Infographic, 2021.

[19] ReversingLabs, "The Attack on SolarWinds: Next-Level Stealth Was Key," *ReversingLabs Blog*, December 2020.

[20] Specops Software, "SolarWinds Hack: Weak Password 'solarwinds123' Cause," *Specops Blog*, 2021; CNN, "Former SolarWinds CEO Blames Intern for 'solarwinds123' Password Leak," February 26, 2021.

[21] DigiCert, "SolarWinds Attack: Is Code Signing to Blame?" *DigiCert Blog*, December 2020; Cryptomathic, "Post SolarWinds Attack: Code-Signing Best Practices," *Cryptomathic Blog*, 2021.

[22] Wikipedia, "2024 CrowdStrike-Related IT Outages," accessed December 2024; Bitsight, "CrowdStrike Outage Timeline and Analysis," *Bitsight Blog*, July 2024.

[23] Fortune, "CrowdStrike Outage Will Cost Fortune 500 Companies $5.4 Billion in Damages," August 3, 2024; Guy Carpenter, "Unveiling the Global Impact of CrowdStrike Event," *Insights*, July 2024.

[24] CrowdStrike, "Falcon Content Update Preliminary Post Incident Report," *CrowdStrike Blog*, July 2024; Computer Weekly, "CrowdStrike Blames Outage on Content Configuration Update," July 2024.

[25] CrowdStrike, "External Technical Root Cause Analysis — Channel File 291," August 6, 2024.

[26] *Ibid.*

[27] Bitsight, "CrowdStrike Outage Timeline and Analysis," *Bitsight Blog*, July 2024.

[28] LinkedIn Data, "Over Half of Fortune 500 Companies Trust CrowdStrike," LinkedIn Corporate Post, 2024; NotebookCheck, "How and Why CrowdStrike Has a Massive Market Share," July 2024.

[29] TechTarget, "BitLocker Workaround May Offer Aid for CrowdStrike Customers," SearchSecurity News, July 2024; Wikipedia, "2024 CrowdStrike-Related IT Outages," accessed December 2024.

[30] Wikipedia, "2024 CrowdStrike-Related IT Outages," accessed December 2024; US Cloud, "Case Study: How US Cloud Led Clients Through the CrowdStrike Outage," Evidence/Client Case Studies, 2024.

[31] Emsisoft, "Unpacking the MOVEit Breach: Statistics and Analysis," *Emsisoft Blog*, 2023.

[32] *Ibid.*

[33] Kroll, "Clop Ransomware MOVEit Transfer Vulnerability Analysis," *Kroll Publications*, Cyber, 2023; SecurityWeek, "Evidence Suggests Ransomware Group Knew About MOVEit Zero-Day Since 2021," June 2023.

[34] CISA and FBI, "#StopRansomware: CL0P Ransomware Gang Exploits CVE-2023-34362," Cybersecurity Advisories AA23-158A, 2023; Kroll, "Clop Ransomware MOVEit Transfer Vulnerability Analysis," 2023.

[35] CISA, "#StopRansomware: CL0P Ransomware Gang Exploits CVE-2023-34362," July 8, 2023; Picus Security, "CVE-2023-34362: Cl0p Ransomware Exploits MOVEit Transfer SQLi Vulnerability," *Picus Security Blog*, 2023.

[36] Kolide, "MOVEit Hack: The Ransomware Attacks Explained," *Kolide Blog*, 2023; Wing Security, "Understanding the MOVEit Ransom Attack," *SaaS Security*, 2023.

[37] Emsisoft, "Unpacking the MOVEit Breach: Statistics and Analysis," *Emsisoft Blog*, 2023.

[38] HIPAA Journal, "Pension Benefit Information 371,359 Hack," 2023; TechTarget, "MOVEit Transfer Cyberattack Impacts 1.2M at Pension Benefit Information," HealthTech Security News, 2023.

[39] ORX News, "MOVEit Transfer Data Breaches Deep Dive," ORX Resource, 2023.

[40] *Ibid.*

[41] *Ibid.*

[42] Colorado State University, "MOVEit Software Cyberattack Notification," Source, July 12, 2023; Cybersecurity Dive, "Progress Software's MOVEit Meltdown: Uncovering the Fallout," News, 2023.

[43] Cybersecurity Dive, "MOVEit Mass Exploit Timeline," News, 2023.

[44] California Attorney General, "MOVEit Individual Notice Letter," Version 1, September 19, 2023.

[45] Kaseya, CEO Fred Voccola statement, quoted in "The Kaseya Ransomware Attack: A Timeline," *CSO Online*, July 2021; Varonis, "REvil MSP Supply Chain Attack," *Varonis Blog*, 2021.

[46] Wikipedia, "Kaseya VSA Ransomware Attack," accessed December 2024.

[47] Dutch Institute for Vulnerability Disclosure (DIVD), Case File DIVD-2021-00011, April 6, 2021.

[48] Tenable, "CVE-2021-30116: Multiple Zero-Day Vulnerabilities in Kaseya VSA Exploited to Distribute Ransomware," *Tenable Blog*, July 2021; Truesec, "How the Kaseya VSA Zero-Day Exploit Worked," *Truesec Hub*, Blog, 2021.

[49] Morphisec, "Real-Time Prevention of the Kaseya VSA Supply Chain REvil Ransomware Attack," *Morphisec Blog*, July 2021; ThreatLocker, "Why ThreatLocker: Kaseya VSA Use Cases," 2021.

[50] Kaseya CEO Fred Voccola statement, quoted in "The Kaseya Ransomware Attack: A Timeline," *CSO Online*, July 2021.

[51] The Record, "Supermarket Chain Coop Closes 800 Stores Following Kaseya Ransomware Attack," July 2021; Truesec, "Coop Back in Business After the Largest Ransomware Attack of All Time," Case Study, 2021.

[52] CharTec, "Have Only One Throat to Choke with Vendor Management," *CharTec Blog*, 2021.

[53] MSP Success, "Kaseya Surpasses ConnectWise in RMM PSA Market Shift," December 2024; Canalys, "RMM Vendors Must Reckon with the New Economic Narrative," Resources, 2024.

[54] Mandiant/Google Cloud, "3CX Software Supply Chain Compromise Initiated by a Prior Software Supply Chain Compromise," *Google Cloud Blog*, Topics/Threat Intelligence, April 2023.

[55] *Ibid.*; BleepingComputer, "3CX Hack Caused by Trading Software Supply Chain Attack," Security, April 2023.

[56] Mandiant/Google Cloud, "3CX Software Supply Chain Compromise," April 2023.

[57] *Ibid.*

[58] ReversingLabs, "The 3CX Attack Gets Wilder: Marks First Cascading Supply Chain Compromise," *ReversingLabs Blog*, April 2023; Fortinet FortiGuard Labs, "3CX Desktop App Compromised (CVE-2023-29059)," *Threat Research*, March 2023.

[59] Huntress, "3CX VoIP Software Compromise & Supply Chain Threats," *Huntress Blog*, March 2023; BleepingComputer, "3CX Confirms North Korean Hackers Behind Supply Chain Attack," April 2023.

[60] Kaspersky, "Gopuram Backdoor Deployed Through 3CX Supply Chain Attack," *Securelist*, April 2023; Volexity, "3CX Supply Chain Compromise Leads to ICONIC Incident," *Volexity Blog*, March 30, 2023.

[61] CISA, quoted in Volexity, "3CX Supply Chain Compromise Leads to ICONIC Incident," March 2023; Kaspersky, "Cryptocurrency Companies Targeted via Gopuram Malware Through the 3CX Attack," *Press Release*, April 2023.

[62] Kaspersky, "Gopuram Backdoor Deployed Through 3CX Supply Chain Attack," *Securelist*, April 2023.

[63] Trading Technologies statement, quoted in TechCrunch, "3CX's Supply Chain Attack Was Caused by Another Supply Chain Attack," April 20, 2023.

[64] The Register, "3CX Decided Supply Chain Attack Indicator Was False Positive," April 3, 2023.

[65] CrowdStrike, "CrowdStrike Detects and Prevents Active Intrusion Campaign Targeting 3CXDesktopApp," *CrowdStrike Blog*, March 29, 2023; 3CX CEO Nick Galea, "3CX DesktopApp Security Alert," *3CX Blog*, March 30, 2023.

[66] Coalition Inc., "Security Incident Retrospective: The 3CX Supply Chain Attack," *Coalition Blog*, 2023.

[67] Wikipedia, "XZ Utils backdoor," https://en.wikipedia.org/wiki/XZ_Utils_backdoor; Dan Lorenc (Chainguard CEO), quoted in The Verge, "The XZ Utils Backdoor: Everything You Need to Know," March 2024; Ars Technica, "What We Know About the XZ Utils Backdoor That Almost Infected the World," April 2024.

[68] Nextgov, "Linux Backdoor Was a Long Con, Possibly with Nation-State Support, Experts Say," April 2024, https://www.nextgov.com/cybersecurity/2024/04/linux-backdoor-was-long-con-possibly-nation-state-support-experts-say/395511/; OpenSSF and OpenJS Foundation, "Joint Statement on xz Utils Social Engineering Attack," April 2024; The Record, "Open Source Foundations Warn of More Attacks Like XZ Utils Backdoor," April 2024.

[69] Securelist by Kaspersky, "Social Engineering Aspect of the XZ Incident," https://securelist.com/xz-backdoor-story-part-2-social-engineering/112476/; Ars Technica, "How One Volunteer Stopped a Backdoor from Exposing Linux Systems Worldwide," April 2024.

[70] Dark Reading, "Attacker Social-Engineered Backdoor Code Into XZ Utils," https://www.darkreading.com/applic security/attacker-social-engineered-backdoor-code-into-xz-utils; Securelist, "Social Engineering Aspect of the XZ Incident," 2024; Ars Technica, "What We Know About the XZ Utils Backdoor," April 2024.

[71] Ars Technica, "What We Know About the XZ Utils Backdoor That Almost Infected the World," April 2024; GitHub xz repository commit history analysis, March-April 2024.

[72] Timeline documented in: Ars Technica, "What We Know About the XZ Utils Backdoor," April 2024; Openwall mailing list analysis, March-April 2024.

[73] CVE-2024-3094, NIST National Vulnerability Database, March 29, 2024; RedHat Security Advisory RHSA-2024:1590, March 2024; Ars Technica technical analysis, "The XZ Backdoor: A Technical Deep Dive," April 2024.

[74] RedHat Security Advisory RHSA-2024:1590; Debian Security Advisory DSA-5649-1, March 2024; Ubuntu Security Notice USN-6698-1, March 2024; Arch Linux security advisory, March 2024.

[75] Andres Freund, original disclosure post to openwall oss-security mailing list, March 29, 2024; The Verge, "How a Microsoft Engineer Helped Prevent a Massive Cyberattack," April 2024; Ars Technica, "How One Volunteer Stopped a Backdoor," April 2024.

[76] Analysis compiled from multiple sources documenting detection limitations: Ars Technica, "What We Know About the XZ Utils Backdoor," April 2024; RedHat Security Blog, "Analysis of CVE-2024-3094," March 2024.

[77] Debian release schedule, https://wiki.debian.org/DebianBookworm; analysis from The Register, "XZ Backdoor Was Weeks Away from Debian Stable," April 2024.

[78] Security analysis documenting insider access vs. external compromise: Ars Technica technical analysis, April 2024; OpenSSF commentary on the incident, April 2024.

[79] Dan Lorenc (Chainguard CEO), quoted in The Verge, "The XZ Utils Backdoor: Everything You Need to Know," March 2024.

[80] Lasse Collin statements documented in: Ars Technica, "What We Know About the XZ Utils Backdoor," April 2024; Securelist, "Social Engineering Aspect of the XZ Incident," 2024; analysis of public communications in xz-devel mailing list archives, 2021-2024.

[81] Census II of Free and Open Source Software, Linux Foundation & Harvard Lab for Innovation Science, 2020; Synopsys OSSRA Report 2024; analysis from Chainguard, "The Ubiquity of XZ Utils," April 2024.

[82] SOC 2 scope and limitations analysis from: AICPA, "SOC 2 Reporting on Controls at a Service Organization," 2023; Vanta, "What SOC 2 Does and Doesn't Cover," 2024.

[83] Nadia Eghbal (now Asparouhova), "Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure," Ford Foundation, 2016; Tidelift, "The 2023 State of the Open Source Maintainer Report," 2023.

[84] Open Source Security Foundation (OpenSSF), "The State of Software Bill of Materials (SBOM) and Cybersecurity Readiness," 2023; Linux Foundation, "A Summary of Census II: Free and Open Source Software," 2020.

[85] Technical analysis from: Ars Technica, "The XZ Backdoor: A Technical Deep Dive," April 2024; RedHat Security Blog, "Analysis of CVE-2024-3094 in xz/liblzma," March 2024; Gynvael Coldwind (Google Project Zero), xz backdoor technical analysis, April 2024.

[86] RedHat Security Blog, "Analysis of CVE-2024-3094 in xz/liblzma," March 2024; Technical analysis from Andres Freund's original disclosure, openwall oss-security mailing list, March 29, 2024.

[87] Andres Freund interview in The Verge, "How a Microsoft Engineer Helped Prevent a Massive Cyberattack," April 2024; detailed timeline from Ars Technica, "How One Volunteer Stopped a Backdoor from Exposing Linux Systems Worldwide," April 2024.

[88] Tidelift, "The 2023 State of the Open Source Maintainer Report," 2023; Linux Foundation & Harvard Innovation Lab, "Census II of Free and Open Source Software," 2020; analysis showing 41.8% of FOSS packages are maintained by single developers.

[89] OpenSSF and OpenJS Foundation, "Joint Statement on xz Utils Social Engineering Attack," April 2024; Nextgov, "Linux Backdoor Was a Long Con, Possibly with Nation-State Support, Experts Say," April 2024; The Record, "Open Source Foundations Warn of More Attacks Like XZ Utils Backdoor," April 2024.

[90] Filippo Valsorda (cryptographer, former Go security lead), commentary on xz backdoor, Twitter/X thread, March-April 2024; Bruce Schneier, "The XZ Backdoor and Trust in Open Source," Schneier on Security blog, April 2024.

[91] Filippo Valsorda, commentary on xz backdoor implications, Twitter/X and personal blog, April 2024.

[92] CISA, "Software Bill of Materials (SBOM)," https://www.cisa.gov/sbom; NTIA, "The Minimum Elements for a Software Bill of Materials," 2021; analysis from "Navigating Cybersecurity with SBOMs: Lessons from Log4j and MOVEit," LinkedIn Pulse, Jack Lilley, 2023.

[93] Nadia Eghbal, "Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure," Ford Foundation, 2016; Veracode, "State of Software Security: Open Source Edition," 2023.

[94] Shared Assessments, "SIG Questionnaire: Software Supply Chain Security," 2024; NIST Secure Software Development Framework (SSDF), SP 800-218, 2022.

[95] OpenSSF, "Securing Open Source Software: A Plan of Action," 2023; GitHub, "The State of Open Source Security 2024," 2024; analysis of industry funding commitments to critical open source infrastructure from Amazon, Google, Microsoft, and others through OpenSSF, 2021-2024.

Chapter 8: Where Framework Scope Ends and New Risks Begin

The seven incidents reveal a systematic pattern: every major TPRM framework was designed for different risk categories than those that have since emerged as most material. Organizations followed the playbooks—this is not execution failure. It's a fundamental disconnect between what frameworks were designed to assess and where actual threats now manifest.

Incident	Attack Vector	DORA's Scope Limitations	SOC 2's Design Boundaries	SBOM/CISA Scope Gaps
SolarWinds (2020)	Build environment compromise enabling code injection during compilation	DORA's ICT risk management does not specifically address build pipeline integrity or artifact verification[1]	SOC 2 assesses change management policies but not build environment isolation, reproducible builds, or signing key protection[2]	SBOMs show what components are declared but cannot verify binaries match source code or detect build-time injection
Log4j (2021)	Transitive dependency 5-10 levels deep in open source library	No mandate for transitive dependency inventory or vulnerability correlation	No requirement to maintain component inventories or assess open source supply chain	SBOM mandate came *after* Log4j; financial services lacks enforcement mechanism for SBOM requirements
Kaseya VSA (2021)	RMM platform compromise propagating through MSP customer base	DORA addresses concentration risk at organizational level but not technology platform concentration (multiple vendors using same RMM)[5]	SOC 2 of MSP assesses their security practices but not the specific RMM platform security posture they've chosen	SBOM of MSP's service would not capture the RMM tool's internal components or vulnerabilities
3CX (2023)	Cascading supply chain: compromised legacy software → employee device → build environment → customer distribution	DORA requires exit strategies for dependencies but was not designed to address vendor employee's personal software choices	SOC 2 was not scoped to cover employee personal device security or vendor's upstream dependencies unrelated to service delivery	SBOM from 3CX would show DesktopApp components after compromise but could not prevent the attack originating from employee's personal device
MOVEit (2023)	Zero-day in widely-used MFT platform affecting customers through vendor dependencies	DORA requires mapping ICT dependencies but provides no mechanism to discover fourth-party tool choices like MFT platforms[4]	SOC 2 Type II of TIAA would not reveal that PBI (their subcontractor) uses MOVEit Transfer for file handling—not because SOC 2 failed, but because operational tools were not part of its original scope	SBOM from TIAA would not include PBI's operational infrastructure choices; MOVEit is infrastructure not a software component
xz Utils (2024)	Backdoor inserted by compromised open source maintainer targeting SSH authentication	DORA provides no framework for monitoring open source maintainer activity or project health	SOC 2 was not designed to assess the security of individual open source dependencies or their maintenance practices	SBOM would list xz Utils as a component but could not detect malicious code inserted by trusted maintainer with commit access
CrowdStrike (2024)	Automatic security tool update bypassing testing and customer controls	DORA requires resilience testing but no framework mandates staged deployment for security tool updates[3]	SOC 2 was not built to assess update deployment practices for products vendors sell (only vendor's own IT systems)	SBOM irrelevant—the faulty component was a configuration file, not a software library; problem was deployment process not component vulnerability

The pattern is stark: **none of these attacks occurred through vectors that existing frameworks were designed to detect.**

Build Integrity Blind Spot: SolarWinds and 3CX both exploited build environment access. SOC 2's CC8.1 (Change Management) assesses whether organizations have code review and testing processes but was not designed to verify build environment isolation, signing key protection, or reproducible build capabilities[6]. SLSA Level 3+ would require hardened, isolated builds with non-falsifiable provenance—but no regulatory framework mandates SLSA compliance, and no Big 4 firm offers SLSA attestation services[7]. (See Chapter 9 for comprehensive analysis of how SOC 2 could evolve to address these emerging risks.)

Transitive Dependency Invisibility: Log4j demonstrated that 60% of vulnerabilities exist in dependencies pulled transitively—components organizations do not directly choose and often do not know they use.[8.1] SOC 2's CC9.2 addresses vendor relationships (organizational entities) but not software component dependencies (technical artifacts)[8]. DORA requires mapping ICT dependencies but provides no technical specification for depth or granularity[9]. SBOMs can theoretically capture transitive dependencies, but only 51% of organizations validate received SBOMs, and tools frequently miss components with incomplete metadata[10].

Automatic Update Authority: CrowdStrike revealed that security tools require automatic update authority to function effectively—but this creates concentration risk when updates bypass testing.[13] No framework addresses pre-deployment testing mandates for security tool updates, staged rollout requirements, or customer control over deployment timing for non-emergency updates. The tension between security responsiveness and operational stability exists outside current compliance structures.

Fourth-Party Tool Opacity: MOVEit and Kaseya demonstrate that the tools vendors use to deliver services (MFT platforms, RMM software) are invisible to traditional vendor management. SOC 2 may carve out subservice organizations like AWS, but operational tools like MOVEit Transfer are not considered "subservice organizations"—they're infrastructure choices[11]. Contracts typically do not require vendors to disclose what file transfer mechanisms or remote management platforms they employ.

Concentration Risk at the Technology Layer: Multiple incidents (CrowdStrike affecting 50% of Fortune 500, Kaseya affecting 75% of MSP market, MOVEit cascading through retirement administrators) demonstrate concentration risk exists at the technology platform level, not just the vendor level. Organizations using different vendors for different services were still simultaneously affected because those vendors converged on common underlying platforms. No framework systematically assesses or addresses this convergent concentration risk.

The Framework-Reality Disconnect: The parties-versus-dependencies mismatch described in Chapter 4 manifests consistently across all seven incidents. When your data is exposed through your retirement administrator's audit service provider's managed file transfer platform, mapping legal entity relationships provides no useful risk insight.

As one analysis concluded: "Third-party risk now includes your fourth and fifth parties, too. Many victims of the breach did not even know MOVEit was in use, because it was used by a vendor's vendor."[12] Until frameworks address this architectural mismatch, the structural gap will persist.

The gaps are now mapped. The question is whether the tens of billions of dollars in incident costs will drive the ecosystem to close them—or whether the industry will wait for the next crisis to force change.

Endnotes - Chapter 8

[1] European Union, "Regulation (EU) 2022/2554 of the European Parliament and of the Council on digital operational resilience for the financial sector (DORA)," Official Journal of the European Union, December 14, 2022. DORA's ICT Risk Management Framework (Article 6) addresses governance, risk identification, protection measures, detection, response, and recovery, but does not specifically mandate build pipeline integrity controls, artifact verification, or reproducible builds.

[2] American Institute of CPAs (AICPA), "Trust Services Criteria," 2017 (with 2022 revised points of focus). SOC 2 Common Criterion CC8.1 (Change Management) requires organizations to "authorize, design, develop or acquire, configure, document, test, approve, and implement changes to infrastructure, data, software, and procedures," but does not require verification of build environment isolation, hermetic builds, or signing key protection practices.

[3] European Union, DORA Article 25 (Digital operational resilience testing). DORA requires financial entities to conduct vulnerability assessments, network security testing, and threat-led penetration testing (TLPT) for systemically important institutions, but does not mandate staged deployment, canary releases, or pre-deployment testing requirements for security tool configuration updates.

[4] European Union, DORA Article 28 (Register of information in relation to contractual arrangements on the use of ICT services). DORA requires financial entities to maintain registers identifying ICT third-party service providers and mapping ICT dependencies, but the Register of Information focuses on legal entities providing ICT services rather than the specific tools (e.g., MFT platforms, RMM software) that vendors use operationally.

[5] European Union, DORA Article 28(3)(g) requires assessment of "concentration of ICT third-party service providers at entity or group level." However, this addresses concentration at the organizational vendor level, not at the technology platform level where multiple independent vendors converge on the same underlying tools (e.g., multiple MSPs all using Kaseya VSA).

[6] AICPA Trust Services Criteria, CC8.1 (Change Management). The criterion addresses whether organizations have processes for code review, testing, and deployment approval, but does not require specific technical controls such as: build environment isolation from development networks, ephemeral build systems, non-falsifiable provenance generation, or reproducible build verification.

[7] Google, "Supply-chain Levels for Software Artifacts (SLSA) Framework," v1.1 specification, April 2025, https://slsa.dev/. SLSA Level 3 requires hardened builds with isolated/ephemeral build environments and non-falsifiable provenance. As of December 2025, no Big 4 audit firm (Deloitte, PwC, EY, KPMG) offers formal SLSA attestation services; compliance remains self-assessed with specialized security vendors offering consulting.

[8] AICPA Trust Services Criteria, CC9.2 (Vendor and Business Partner Risk Management). The 2022 revision added language about "identifying and evaluating vulnerabilities arising from vendor and business partner relationships," but this addresses organizational relationships (collecting vendor SOC reports, reviewing contracts) rather than software component dependencies like open-source libraries or transitive dependencies.

[8.1] Snyk analysis found that 60% of affected Java applications used Log4j as a transitive dependency rather than a direct dependency. This means organizations inherited Log4j exposure through other

libraries that themselves depended on Log4j, creating invisible vulnerability exposure. Source: Snyk, "Log4j Vulnerability Explained," December 2021.

[9] European Union, DORA Article 28 (ICT Third-Party Risk Management). While DORA requires mapping ICT dependencies and maintaining a Register of Information on ICT service providers, the regulation provides no technical specification for dependency inventory depth (direct vs. transitive), granularity (library-level vs. service-level), or format (e.g., SBOM standards).

[10] Linux Foundation/OpenSSF, "Toward a Comprehensive Software Supply Chain Assurance Framework for Financial Institutions," 2024 industry analysis. Research indicates only 51% of organizations validate received SBOMs, and SBOM generation tools frequently miss components with incomplete metadata, components installed dynamically at runtime, or components embedded as binaries rather than declared as dependencies.

[11] AICPA, "SOC 2 Reporting on an Examination of Controls at a Service Organization Relevant to Security, Availability, Processing Integrity, Confidentiality or Privacy," SSAE 18. SOC 2 allows service organizations to "carve out" subservice organizations (e.g., AWS for infrastructure) using Complementary Subservice Organization Controls (CSOCs). However, operational tools like MOVEit Transfer (MFT platforms) or Kaseya VSA (RMM platforms) are typically not considered "subservice organizations" requiring carve-out disclosure—they are treated as internal technical implementation choices.

[12] Emsisoft, "Unpacking the MOVEit Breach: Statistics and Analysis," 2023, https://www.emsisoft.com/en/blog/4
the-moveit-breach-statistics-and-analysis/. Emsisoft's analysis found that "for almost two-thirds of MOVEit victims, breaches occurred because their third-party vendors used MOVEit—or their vendor's vendors used the file-transfer service."

[13] CrowdStrike, "Falcon Content Update Preliminary Post Incident Report," CrowdStrike Blog, July 2024; Microsoft Security Blog, "Windows security best practices for integrating and managing security tools," July 2024. Endpoint security tools require kernel-level access and rapid update capabilities to detect and respond to emerging threats. The rapid response content mechanism enabled CrowdStrike to deploy threat detection updates within hours of identifying new attack patterns, but this same mechanism propagated the faulty configuration update that caused the July 2024 outage.

Chapter 9: The Cost of Misalignment

Between 2021 and 2024, seven major incidents stemming from technical dependencies—not traditional vendor relationship failures—caused an estimated **$30 billion to $200 billion in combined global economic damage** (see methodology in endnotes)[1]. This staggering cost provides the most compelling evidence for the framework shift outlined in this paper.

These were not hypothetical risks or theoretical vulnerabilities. These were actual operational failures and security breaches where the dependency chains described throughout this analysis became attack vectors or failure cascades. The technical details of each incident are documented in Chapter 7; the framework gaps they exposed are mapped in Chapter 8. This chapter quantifies the aggregate cost, extracts the systematic patterns explaining why traditional TPRM frameworks failed, and examines the regulatory response lag that leaves institutions navigating these risks without clear guidance.

The Aggregate Cost: Tens of Billions and Counting

The seven incidents examined in this paper represent an estimated $30 billion to $200 billion in combined economic damage over a four-year period (2021-2024)[2]. Note that incident cost estimates vary widely depending on methodology; the figures below represent mid-range estimates compiled from multiple industry sources:

Incident	Estimated Cost	Primary Impact Category	Estimate Quality
SolarWinds (2020)	~$100 billion	Software delivery compromise	Upper-bound; includes indirect intelligence damage
Kaseya VSA (2021)	$1-2 billion	MSP/RMM platform exploit	Moderate confidence; based on ransom + recovery
Log4j (2021)	$15-100+ billion	Embedded component vulnerability	Wide range; methodological challenges
3CX (2023)	$2+ billion	Software delivery compromise	Lower confidence; limited public data
MOVEit (2023)	$15.8 billion	MFT platform breach	High confidence; documented breach costs
xz Utils (2024)	$0 (prevented)	Open source maintainer compromise	Incident prevented; potential impact incalculable
CrowdStrike (2024)	$10+ billion	Operational failure at scale	High confidence; documented insured losses

These figures represent:

- **Direct costs**: IT response, forensics, legal fees, regulatory fines, customer compensation
- **Business disruption**: Lost revenue, operational downtime, delayed transactions
- **Recovery expenses**: System restoration, enhanced monitoring, control implementation
- **Remediation investments**: Software updates, infrastructure changes, process improvements

What the aggregate cost demonstrates is that **technical dependency risk is now a material financial risk category on par with credit risk, market risk, and operational risk as traditionally defined**. Organizations cannot treat vendor management as a compliance checklist exercise when the actual financial exposure from dependency failures reaches tens of billions of dollars annually.

What Traditional TPRM Missed: A Pattern Analysis

Examining these seven incidents reveals five systematic gaps in how traditional third-party risk management programs assess vendor relationships:

Gap 1: Entity-Centric vs. Dependency-Centric

Traditional TPRM organizes risk around legal entities: Who is our vendor? What is their security posture? What are our contractual protections? This approach assumes risk flows through organizational relationships.

The incidents reveal risk flows through technical dependencies that cross organizational boundaries:

- **Log4j**: The Apache Software Foundation is not a vendor. Organizations do not contract with Apache. Yet Apache's logging library created exposure across thousands of vendor products.
- **MOVEit**: Organizations were breached by Progress Software despite never having a contractual relationship with Progress because their vendor's vendor used MOVEit.
- **Kaseya**: Businesses were ransomed by an attack on Kaseya despite never being Kaseya customers—their MSP chose Kaseya, and the organization inherited that dependency invisibly.

Gap 2: Policy Assessment vs. Architecture Assessment

Traditional vendor risk assessments focus on documented policies, certifications, and controls: Does the vendor have an information security policy? Do they conduct penetration testing? Are they SOC 2 certified?

The incidents demonstrate that policy compliance is independent of architectural risk:

- **CrowdStrike** had excellent security policies and achieved top scores in industry evaluations. The architectural risk came from kernel-mode execution, global simultaneous deployment, and industry-wide concentration—factors unrelated to policy assessment.
- **SolarWinds** maintained certifications and documented security practices. The build environment compromise exploited implementation gaps that policy reviews do not reveal.
- **3CX** was a reputable vendor with standard security certifications. The supply chain compromise occurred in infrastructure that vendor risk questionnaires do not examine.

Gap 3: Organizational Concentration vs. Industry Concentration

Traditional concentration risk assessments ask: "Are we too dependent on a single vendor?" The focus is on organizational resilience—could we survive if this vendor failed?

The incidents reveal industry-level concentration creates systemic risk:

- **CrowdStrike**: Individual organizations rationally selected the best endpoint security platform. Collectively, 60% of Fortune 500 companies created a single point of failure at industry level.
- **MOVEit**: MFT platforms achieve market concentration because regulated industries need compliant file transfer solutions. This concentration made a single vulnerability into a sector-wide event.
- **Kaseya**: MSP industry economics drive platform consolidation. When three vendors control 75% of the RMM market, a compromise affects a large fraction of MSP clients simultaneously.

Gap 4: Direct Vendor Risk vs. Nth-Party Dependencies

Traditional TPRM assesses direct vendors and sometimes attempts to evaluate "critical fourth parties." The framework assumes risk exposure correlates with contractual distance—direct vendors are higher priority than their subcontractors.

The incidents demonstrate nth-party dependencies create exposure regardless of contractual relationships:

- **MOVEit**: Organizations were exposed through chains like Bank → Retirement Plan Administrator → Death Audit Service → MOVEit Transfer. The organization had visibility into the retirement plan administrator but none into PBI Research Services or Progress Software.
- **Kaseya**: Small businesses contracted with MSPs and inherited exposure to Kaseya VSA without knowledge that RMM software was part of the technology stack.
- **Log4j**: Software contains transitive dependencies—libraries that depend on other libraries. Organizations running vendor software inherited vulnerabilities from components nested four or five layers deep in dependency trees.

Gap 5: Vendor Failure vs. Vendor Success as Risk

Traditional risk management assumes vendor failure (bankruptcy, service outage, security breach) creates risk. The framework evaluates vendor financial stability, business continuity planning, and incident response capabilities.

The incidents reveal vendor success creates concentration risk:

- **CrowdStrike** did not fail as a company—it succeeded in becoming the industry-leading endpoint security platform. That success created systemic risk when an operational error affected millions of systems simultaneously.
- **MOVEit** succeeded in becoming the compliant MFT solution for regulated industries. That market penetration meant a vulnerability affected organizations across healthcare, finance, insurance, and government simultaneously.

The paradox: the better a vendor's product, the more widely it is adopted; the more widely it is adopted, the greater the systemic risk if that vendor experiences operational or security failure.

The Regulatory Response Timeline: An 18-36 Month Pattern

A consistent pattern emerges across these incidents: the gap between major technology disruptions and meaningful regulatory guidance addressing the specific risk category revealed by those disruptions spans 18-36 months. This timeline—which reflects the deliberate processes of notice-and-comment rulemaking, interagency coordination, and stakeholder consultation—creates a challenging period where financial institutions understand a new risk category exists but lack regulatory clarity on expectations, minimum standards, or safe harbor practices[3].

Post-SolarWinds Response Timeline

December 2020: SolarWinds breach disclosed

May 2021: President Biden issues Executive Order 14028 on cybersecurity, establishing high-level principles but limited specific requirements[4]

February 2022: NIST releases initial secure software development framework guidance

September 2022: OMB releases implementation guidance for federal agencies on software supply chain security

July 2023: CISA begins publishing guidance on Software Bills of Materials (SBOMs)[5]

The timeline reveals that organizations faced the SolarWinds risk category (compromised software delivery mechanisms) for approximately three years before comprehensive regulatory frameworks emerged defining minimum standards and expectations.

Post-Log4j Response Timeline

December 2021: Log4Shell vulnerability disclosed

March 2022: CISA adds Log4j to Known Exploited Vulnerabilities catalog but provides only reactive guidance

May 2023: Executive Order on improving software supply chain security emphasizes SBOMs, but implementation remains voluntary for most sectors

Ongoing: Financial services regulators have issued no specific examination procedures for embedded open-source component risk management[6]

The gap means financial institutions still lack clear regulatory expectations for how to identify, track, and manage vulnerabilities in open-source components embedded in vendor software—despite Log4j demonstrating this is a critical risk category.

Post-CrowdStrike Response Timeline

July 2024: CrowdStrike incident occurs

September 2024: House Homeland Security Committee holds hearing with CrowdStrike; initial discussions of "systemically important technology providers"

Ongoing: Potential NIST standards for software update testing and deployment; possible Congressional legislation on critical infrastructure technology provider oversight remain under development[7]

Financial institutions continue to face vendor concentration risk and kernel-mode software delivery risk without clear regulatory frameworks defining reasonable expectations—a gap likely to persist until comprehensive guidance emerges.

Why This Timeline Matters

The 18-36 month regulatory response timeline creates three problems for financial institutions:

Compliance Uncertainty: Organizations do not know what standard they'll be held to when examiners eventually develop specific guidance. This makes it difficult to justify investments in new controls or to prioritize remediation activities.

Peer Practice Divergence: Without regulatory clarity, different institutions adopt different approaches to newly identified risks. This divergence means "industry best practice" does not crystallize, making it harder for organizations to benchmark their maturity.

Retroactive Expectations: When regulatory guidance does eventually emerge, it often includes expectations that institutions "should have" implemented controls earlier. Examiners cite major incidents as evidence that risks were "known," even if specific regulatory requirements did not exist.

The solution is not necessarily faster regulatory response—hastily written guidance can be counterproductive. The problem is that our TPRM frameworks remain anchored to legal entity relationships

(vendor contracts, due diligence, questionnaires) while these incidents demonstrate that technical dependencies (software components, delivery mechanisms, infrastructure platforms) propagate risk regardless of contractual boundaries.

The Question These Incidents Force

The multi-billion dollar cost documented in this chapter forces financial institutions to confront an uncomfortable question: **Can effective third-party risk management coexist with vendor concentration driven by genuinely superior products?**

Every organization that selected CrowdStrike, deployed MOVEit, or embedded Log4j-dependent software made rational decisions optimizing for security effectiveness, compliance requirements, or operational efficiency. The concentration emerged not from vendor lock-in or anti-competitive practices but from market dynamics rewarding technical excellence.

The tragedy is that individually rational choices created collectively fragile systems.

Traditional TPRM frameworks assume risk can be managed through better vendor selection, enhanced due diligence, and stronger contractual protections. These incidents demonstrate that when 60% of an industry adopts the same technology platform—even for excellent reasons—the industry becomes systemically vulnerable to single points of failure.

Understanding the cost and pattern of failures is necessary but insufficient. Part IV addresses how to restructure third-party risk management to account for technical dependencies, industry-level concentration, and nth-party exposure chains that traditional frameworks cannot capture.

Endnotes - Chapter 9

[1] Aggregate total calculated from individual incident cost estimates documented throughout this paper. Cost estimates represent economic damage including direct costs, business disruption, recovery expenses, and remediation investments across all affected organizations globally. See Chapter 7 for individual incident documentation.

[2] Cost range reflects methodological challenges in estimating supply chain incident damages. Lower bound (˜$30B) uses conservative direct-cost estimates: SolarWinds ($0.1B documented), Kaseya ($1.5B), Log4j ($15B lower estimate), MOVEit ($15.8B), 3CX ($2B), CrowdStrike ($10B). Upper bound (˜$200B) includes indirect impacts, opportunity costs, and upper-range estimates for Log4j ($100B+) and SolarWinds ($100B including intelligence damage). The wide range reflects genuine uncertainty in incident cost attribution and methodology.

[3] Regulatory response timeline analysis compiled from multiple sources documenting guidance issuance following major incidents.

[4] Executive Order 14028, "Improving the Nation's Cybersecurity," May 12, 2021, https://www.whitehouse.gov/briefi room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[5] "Secure Software Development Framework," NIST SP 800-218, February 2022; "OMB Memorandum M-22-18: Enhancing the Security of the Software Supply Chain through Secure Software Development Practices," September 2022; "CISA SBOM Guidance," July 2023, https://www.cisa.gov/sbom

[6] Log4j regulatory response timeline compiled from CISA advisories, NIST guidance updates, and financial services regulatory issuances (OCC, Federal Reserve, FDIC) December 2021-present. As of publication, no financial services-specific examination procedures for open-source dependency management have been issued.

[7] "House Committee Examines CrowdStrike Processes in Congressional Hearing," House Homeland Security Committee, September 26, 2024, https://homeland.house.gov/2024/09/26/icymi-committee-examines-crowdstrike-processes-in-first-congressional-hearing-on-the-disastrous-july-global-it-outage/; "CrowdStrike IT Outage: Impacts to Public Safety Systems," Congressional Research Service, https://www.congress.gov/crs-product/IF12717

Part IV: The Path Forward

Part IV: The Path Forward

Chapter 10: Multi-Stakeholder Problem, Multi-Stakeholder Solution

The challenges revealed in the preceding chapters cannot be solved by financial institutions acting alone. The fundamental mismatch between current TPRM frameworks and the realities of modern software dependencies reflects systemic gaps in the regulatory, assurance, and vendor ecosystems. A multi-stakeholder problem demands a multi-stakeholder solution.

Financial institutions cannot solve this alone by "doing better" at vendor management. It requires coordinated evolution across four distinct but interconnected domains: regulatory guidance, assurance frameworks, vendor compliance capabilities, and institutional technical maturity. Each stakeholder has a role to play; progress requires movement across all four dimensions.

10.1 Regulatory Evolution: Creating the Demand Signal

Regulatory pressure is the single most powerful force for driving compliance behavior in financial services. History demonstrates this consistently: vendors resist new requirements until regulations make them table stakes, at which point market forces accelerate adoption. SOC 2 was not widely adopted because vendors recognized its inherent value—it became standard because customers demanded it, and customers demanded it because regulators expected evidence of third-party security controls.

The same demand-creation mechanism must extend to software supply chain transparency.

The Current Regulatory Gap

Current U.S. financial services guidance addresses third-party risk at the entity level but remains largely silent on code-level dependency risks:

FFIEC IT Examination Handbook (revised August 2024) references secure software development practices and cites NIST SP 800-218 (Secure Software Development Framework), but stops short of requiring Software Bill of Materials (SBOM) collection or dependency transparency from vendors.[1] Examination procedures address "supply chain communication" and "development designs that include security validation," yet lack specific expectations around application-layer dependency disclosure.

OCC/Federal Reserve/FDIC Interagency Guidance on Third-Party Relationships (June 2023) establishes comprehensive lifecycle management requirements but focuses on vendor entities rather than technical dependencies.[2] The guidance requires risk-based due diligence and ongoing monitoring, but does not explicitly address open source libraries, CI/CD pipeline security, or software composition analysis.

NYDFS Cybersecurity Regulation (23 NYCRR 500) requires written policies setting minimum cybersecurity standards for third-party service providers and mandates CISO reporting on third-party risks.[3] Yet like its federal counterparts, it does not require dependency-level visibility or SBOM provisions.

The result: financial institutions have regulatory obligations to manage third-party risk, but lack clear examination expectations around the dependency-level risks that drove Log4j, SolarWinds, and similar incidents.

The European Model: DORA's Entity-Level Approach

The European Union's Digital Operational Resilience Act (DORA), which took effect January 17, 2025, represents the most comprehensive operational risk regulation ever implemented in financial services.[4] DORA requires financial entities to maintain detailed Registers of Information cataloging all ICT third-party contracts, mandates assessment of "long or complex chains of subcontracting," and requires identification of "material sub-contractors" that "effectively underpin" critical services.[5]

DORA is a significant regulatory advancement—but it remains fundamentally **entity-focused**. The framework addresses "who are your subcontractors?" but not "what's in your code?" Code libraries, which lack Legal Entity Identifiers (LEIs) and exist outside traditional contractual relationships, do not fit cleanly into DORA's structure. Notably, the European Commission rejected Article 5 of the July 2024 draft Regulatory Technical Standards that would have required ongoing monitoring of subcontractor chains, suggesting even European regulators recognize the practical limits of entity-level oversight[5.1].

DORA demonstrates regulatory willingness to tackle supply chain complexity, but even this ambitious framework has not bridged the gap to code-level dependencies.

What Regulatory Evolution Would Look Like

Effective regulatory guidance on software supply chain risk would establish clear examination expectations:

SBOM as a Standard Due Diligence Element: Guidance should position Software Bill of Materials collection as an expected component of vendor due diligence for critical software providers, comparable to SOC 2 reports or business continuity plans. This does not require universal SBOM collection from all vendors—a risk-tiered approach would focus requirements on vendors supporting critical functions or processing sensitive data.

Dependency Transparency in Vendor Assessments: Examination procedures should include evaluator questions about how institutions assess application-layer dependencies, concentration risks in shared code libraries, and mechanisms for responding to widespread vulnerabilities in third-party components. The goal is not to prescribe specific methodologies but to establish that dependency-level risk is within the scope of institutional responsibility.

Software Delivery Mechanism Evaluation: Guidance should acknowledge that how software is updated—the CI/CD pipeline, deployment controls, testing procedures—constitutes a material risk dimension. The CrowdStrike incident demonstrated that automatic update mechanisms can propagate failures as efficiently as they deploy fixes. Vendor assessments should evaluate software delivery risk alongside traditional security and operational risk categories.

Integration of SBOM Standards: Regulatory guidance should reference existing technical standards—CycloneDX, SPDX, SLSA provenance, NIST SSDF—to provide vendors with clear implementation pathways.[6] This avoids reinventing technical specifications while signaling regulatory expectations.

The objective is not to create a new compliance burden divorced from actual risk reduction. Rather, it is to align regulatory expectations with the demonstrated failure modes of modern technology dependencies. When regulators signal that dependency visibility matters, financial institutions gain the leverage to demand it contractually, and vendors face market pressure to provide it.

The Demand Creation Chain

Fintech willingness to provide SBOMs is currently a moot point. Most fintech vendors do not generate SBOMs because no customer is requiring them contractually, no audit framework is testing for them, and no regulator is examining for them. The rational business decision is to allocate engineering resources elsewhere.

Regulatory guidance changes this calculus by setting off a predictable chain reaction:

Regulatory Pressure → FI Pressure → Fintech Compliance

This is precisely how every significant compliance requirement has propagated through the industry:

Data Processing Addendums (DPAs) became table stakes for SaaS vendors following GDPR requirements in Europe (effective May 2018). Initially, vendors resisted, arguing that DPAs imposed operational constraints and legal liability. Within 18 months, market pressure—combined with subsequent U.S. state privacy laws like CCPA (2020)—made DPAs non-negotiable for enterprise contracts. Today, refusing to execute a DPA disqualifies a vendor from consideration at most financial institutions.[7.1]

SOC 2 Type II reports followed a similar trajectory. In the early 2010s, SOC 2 was considered optional or "nice to have."[7] As financial institutions began requiring SOC 2 reports in vendor contracts—driven by regulatory expectations around third-party risk management—vendors recognized that lacking a SOC 2 report eliminated them from procurement processes. By the late 2010s, SOC 2 Type II became the baseline expectation for any SaaS vendor serving regulated entities.

The same mechanism will drive SBOM adoption—once regulatory guidance establishes examination expectations, financial institutions will incorporate SBOM requirements into contracts, and vendors will comply because market access depends on it.

Addressing the "Unfunded Mandate" Critique

Critics will argue that SBOM requirements impose costs on vendors without corresponding security benefits, particularly for smaller fintech companies operating on limited budgets. This critique deserves attention.

The reality is more nuanced. For organizations with modern CI/CD infrastructure—automated build pipelines, containerized deployments, explicit dependency management—SBOM generation is technically straightforward. GitHub and GitLab provide native SBOM generation capabilities. Open-source tools like Syft and Trivy can produce CycloneDX or SPDX-formatted SBOMs in minutes, integrated directly into CI/CD pipelines[7.3]. For organizations already using automated build processes, SBOM generation adds minimal engineering overhead—often less than a day of initial

configuration. However, this describes a best-case scenario. Many fintechs—particularly those with legacy codebases, monolithic applications, or years of accumulated technical debt—face substantially greater challenges. Applications built before modern dependency management may have implicit dependencies, vendored code, or components installed through non-standard mechanisms that automated tools miss. For these organizations, achieving accurate SBOM generation may require significant remediation work before transparency is even possible.

The solution is not to exempt legacy vendors but to establish **tiered expectations with reasonable timelines**. A vendor with a modern stack might demonstrate SBOM capability within months. A vendor undertaking modernization might receive a longer runway with milestone-based progress requirements. The goal is transparency about the path forward, not exemption from the destination.

The actual cost is not in generation but in **organizational processes around SBOMs**: maintaining accuracy as dependencies change, establishing workflows for vulnerability notifications, training customer-facing teams to discuss SBOMs intelligently, and potentially facing customer scrutiny over dependency choices. These are real costs, but they are also reasonable expectations for organizations providing critical software to regulated entities.

Moreover, the alternative—financial institutions operating with zero visibility into what code their vendors are running—is not a sustainable risk posture. Dependency transparency does impose costs, but those costs pale against the risks they mitigate. Software supply chain attacks cost an estimated $46 billion in 2023 and are projected to reach $138 billion by 2031.[7.2] The case for transparency is compelling.

Regulatory guidance that establishes clear but proportionate expectations—tiered by vendor criticality, with implementation timelines that allow for capability building—can drive necessary evolution without imposing unreasonable burdens on smaller vendors.

10.2 Assurance Framework Updates: Closing the Attestation Gap

Regulatory guidance creates demand; assurance frameworks provide the mechanism for verifiable compliance. Current frameworks leave systematic gaps that undermine their utility for software supply chain risk.

SOC 2's Design Scope and Emerging Gaps

SOC 2 Type II remains the baseline attestation expectation for SaaS vendors serving financial institutions. The framework was designed to address organizational security controls in an era when software supply chain risks were not yet widely recognized as material threats. The 2017 Trust Services Criteria, which remain substantively unchanged despite "revised points of focus" in 2022,[8] were not designed to address:

- Software Bill of Materials (SBOM) generation or maintenance
- Open source dependency management or vulnerability tracking
- CI/CD pipeline security or build integrity
- Software delivery mechanism controls
- Application-layer dependency disclosure
- Security tooling transparency (SIEM, MSSP, EDR providers)

The "carve-out" methodology compounds these gaps. When a SaaS vendor carves out Amazon Web Services, the auditor tests the vendor's monitoring controls over AWS—not AWS's actual controls.

When AWS carves out their subprocessors, another layer of abstraction emerges. No one is mapping end-to-end assurance coverage across realistic dependency chains; instead, carve-outs create the appearance of comprehensive oversight while leaving substantial blind spots.

Subservice organization disclosures in SOC 2 reports typically include only infrastructure providers—AWS, Microsoft Azure, Google Cloud Platform, co-location facilities. Application-layer dependencies, security service providers, and software delivery tools remain undisclosed. Auditors are not failing here—they're following scoping decisions permitted under current standards.

The result: an institution can review a vendor's SOC 2 Type II report and conclude the vendor maintains adequate controls, while remaining completely unaware of critical dependencies like managed file transfer systems (MOVEit), logging libraries (Log4j), or endpoint management platforms—the precise attack vectors exploited in recent major incidents.

The SOC for Supply Chain Missed Opportunity

The AICPA introduced **SOC for Supply Chain** examinations in March 2020 explicitly in response to software supply chain attacks.[9] This should have been the mechanism for comprehensive software dependency attestation. Instead, it applies the same Trust Services Criteria used in SOC 2, and its documentation focuses primarily on physical product manufacturing and distribution—"producers, manufacturers, and distribution companies."

While AICPA materials note that SOC for Supply Chain covers "software, applications, and electronic products," the framework lacks specific criteria for:

- Build provenance and integrity
- Transitive dependency management
- Software composition analysis
- Source code supply chain security
- Automated deployment pipeline controls

The opportunity to create a software-specific attestation framework was available in 2020. The industry chose to extend existing criteria rather than develop new ones tailored to software supply chain risks. Five years later, the gap remains.

Pathways for Framework Evolution

Two pathways exist for closing the attestation gap:

Option A: SOC 2 Scoping Expansion

AICPA could revise the Trust Services Criteria to explicitly include application-layer dependencies and software supply chain controls. This would require adding control objectives around:

- **SBOM Maintenance**: Criteria requiring generation, accuracy validation, and periodic updates of machine-readable SBOMs in standardized formats (CycloneDX, SPDX)
- **Dependency Curation**: Controls for vetting third-party components before incorporation, tracking vulnerability disclosures, and enforcing policies against known-vulnerable libraries
- **Build Integrity**: Requirements for CI/CD pipeline security, build environment isolation, and provenance generation for released artifacts
- **Software Delivery Controls**: Criteria for staged deployments, pre-release testing, customer control over update timing, and rollback capabilities

- **Subservice Organization Expansion**: Mandatory disclosure of application-layer dependencies and security service providers, not just infrastructure

The challenge: AICPA standards evolve slowly, driven by practitioner consensus and technical committee processes that prioritize broad applicability across industries.[10] Software supply chain controls are highly technical and may not apply cleanly to non-software service organizations. Achieving consensus on meaningful criteria could take years.

That said, AICPA has the institutional infrastructure, the Big 4 relationships, and the technical hiring capacity to lead this evolution—if market demand and regulatory pressure create sufficient incentive. The same organization that created SOC 2 in response to SAS 70 misuse could create the next framework evolution in response to supply chain risk. The question is whether the incentive structure will drive that outcome.

This paper positions the AICPA not as the source of the problem but as the institution best positioned to lead its resolution. SOC 2's existing market penetration, institutional trust, and auditor ecosystem give it advantages that no greenfield framework could replicate. An evolved SOC 2 that addresses software supply chain risk would leverage decades of established practice rather than requiring the industry to adopt an unfamiliar alternative.

Option B: Parallel Attestation Framework

Alternatively, the industry could develop a parallel, software-specific attestation standard that complements rather than replaces SOC 2. This would allow:

- **SOC 2** to retain its role for general security, availability, and processing integrity controls
- **Software Supply Chain Attestation** (call it "SOC 2.5" or a distinct framework) to provide specialized coverage of build integrity, dependency management, and software delivery risks

This approach has precedent: SOC for Cybersecurity exists alongside SOC 2, providing specialized coverage for cybersecurity risk management.[10.1] A similar specialized framework for software supply chain could integrate existing technical standards—SLSA provenance levels, NIST SSDF practice groups, S2C2F dependency management practices—into auditable control objectives.

The advantage: faster implementation without requiring AICPA to overhaul SOC 2. The disadvantage: vendors face dual attestation costs, and institutions must evaluate multiple reports per vendor.

The Auditor Capability Challenge

Regardless of which pathway the industry pursues, a significant barrier remains: **auditor expertise**. Current SOC 2 auditors are skilled at evaluating change management policies, deployment logs, access controls, and disaster recovery plans. Far fewer possess deep expertise in:

- Container security and Kubernetes configurations
- SBOM validation and accuracy testing
- Build pipeline integrity assessment
- Provenance verification and signing infrastructure
- Software composition analysis tools and vulnerability correlation

Training programs are emerging—Practical DevSecOps offers a Certified Software Supply Chain Security Expert (CSSE) credential; SANS provides a Software Supply Chain Security Graduate Certificate[11.1]—but no AICPA-recognized certification for software supply chain attestation

auditing exists. Building this capability across major audit firms will require substantial time, assuming market demand accelerates investment.

The implication: even if frameworks are updated today, the auditor capacity to execute comprehensive software supply chain attestations will trail framework updates by at least a year.

Cost Implications and Tiered Approaches

Comprehensive software supply chain attestation would substantially increase vendor compliance costs. The components involved—SBOM generation tooling, management platforms, provenance infrastructure, build pipeline hardening, additional audit scope, and specialized consulting—can collectively represent a significant investment, potentially doubling or tripling current compliance expenditures for vendors already maintaining SOC 2 Type II attestations.[11] Industry cost benchmarks indicate SOC 2 Type II audits range from $20,000-$80,000 annually depending on organization size and complexity; adding comprehensive software supply chain attestation with SLSA Level 2+ provenance, continuous dependency monitoring, and expanded audit scope could increase total compliance costs to $50,000-$200,000+ for critical vendors.[11a]

This creates legitimate concerns about market accessibility for smaller fintechs operating on constrained budgets. The solution is tiered requirements:

Basic Attestation (for lower-risk vendors): SBOM generation and availability, basic vulnerability scanning, documented dependency management process. Lower audit costs, achievable with limited tooling investment.

Enhanced Attestation (for critical vendors): Full SLSA Level 2+ provenance, continuous dependency monitoring, comprehensive build integrity controls, automated policy enforcement. Higher costs justified by vendor criticality.

Tiering allows the ecosystem to evolve without creating insurmountable barriers for smaller vendors while ensuring that institutions supporting critical functions meet enhanced standards.

10.3 Fintech and Technology Vendor Participation

The challenges outlined in this analysis create gaps that technology vendors—including fintechs, TPRM platform providers, and risk management solution developers—are uniquely positioned to address. The market gaps are real: financial institutions need dependency visibility tools, SBOM analysis capabilities, and technical translation services that bridge software engineering and risk management domains.

Fintech participation in the multi-stakeholder solution is essential. As both providers of software to financial institutions and consumers of third-party dependencies themselves, fintechs occupy a critical position in the dependency chain. Their willingness to adopt transparency practices—generating SBOMs, disclosing dependency chains, implementing software supply chain security controls—directly determines whether financial institutions can achieve dependency-level visibility.

Technology vendors capable of building SBOM aggregation platforms, dependency risk scoring services, and technical analysis tools can provide commercial solutions that financial institutions need but cannot efficiently build in-house. The specific products and approaches will vary based on market demand, technical feasibility, and competitive positioning. What matters is recognizing that fintech and technology vendors are not passive recipients of regulatory requirements—they are active stakeholders whose innovation and compliance capabilities shape the ecosystem's evolution.

10.4 Financial Institution Program Maturity: Building Internal Capability

Regulatory evolution, framework updates, and fintech innovation are necessary but insufficient. Financial institutions must build internal technical capability to consume and act on dependency-level information.

The most practical, near-term step: **integrate IT expertise into TPRM functions**.

The IT-TPRM Integration Imperative

Third-Party Risk Management has historically been staffed with professionals whose expertise lies in contracts, compliance, operational risk, and vendor relationship management. These competencies remain essential—but they are no longer sufficient.

The highest-risk vendors that TPRM programs oversee are **technical organizations**: SaaS platforms, cloud providers, managed security service providers, payment processors, core banking systems, cybersecurity tools. Evaluating these vendors requires technical fluency that traditional risk professionals do not possess.

Consider the questions TPRM analysts must now answer:

- "This vendor uses Log4j 2.15. Is that the vulnerable version or the patched version?"
- "The vendor's SBOM lists 847 dependencies. Which ones are high-risk?"
- "What does 'container orchestration via Kubernetes' mean for our data security?"
- "The vendor carved out their CI/CD pipeline from SOC 2 scope. Should we care?"
- "This vendor uses an MSSP we've never heard of. How do we assess that relationship?"

These are technical questions that require technical expertise to answer competently. No amount of risk management training will enable a professional with a compliance or legal background to evaluate Kubernetes security configurations, interpret SBOM component listings, or assess CI/CD pipeline integrity.

The solution is not to replace TPRM teams with technologists. Rather, it is to **integrate technical subject matter expertise into TPRM workflows**, creating cross-functional capability that combines risk management discipline with technical fluency.

Implementation Approaches

Financial institutions address this capability gap through various organizational models. Some create dedicated roles—a "Third-Party Technology Risk Analyst" or equivalent—embedded within the TPRM organization. Such positions bridge vendor management and technical assessment, translating SBOM findings into risk language, evaluating CI/CD security practices, and monitoring vulnerability disclosures affecting the vendor portfolio.

Other institutions reallocate existing IT resources, assigning engineers or architects to support TPRM on a fractional or rotational basis. This approach leverages institutional knowledge without requiring new headcount, though divided attention and competing priorities can limit effectiveness.

Still others establish cross-functional collaboration frameworks: standing IT liaisons, joint review processes for critical vendor assessments, and escalation pathways for complex technical evaluations. This model minimizes organizational disruption but may lack dedicated ownership and consistent application.

The appropriate approach depends on institutional size, portfolio complexity, and organizational structure. Large institutions with hundreds of SaaS relationships often require dedicated resources; smaller institutions may achieve adequate coverage through structured IT collaboration. The specific model matters less than the underlying principle: **TPRM can no longer operate in isolation from technical expertise**. The nature of third-party risk has become too technical for traditional risk management skills alone to address adequately.

The Economics: Technical Capability vs. Incident Cost

Executives concerned about the cost of adding technical capability to TPRM should consider the alternative: the cost of failing to identify critical vendor dependencies before they result in operational disruptions.

The investment required—whether for dedicated headcount, fractional IT allocation, or formalized collaboration—is modest relative to the costs these capabilities help avoid. Consider the scale of recent incidents:

- **MOVEit breach**: Estimated $15.8 billion in total damages across 2,700+ affected organizations; individual institutions faced breach notification costs, regulatory scrutiny, litigation, and reputational damage[12]
- **CrowdStrike outage**: $10+ billion globally; affected organizations experienced operational downtime, customer service disruptions, manual workaround costs, and potential regulatory examination[13]
- **Log4j response**: Emergency weekend response teams, vendor portfolio scans, patch coordination—institutions spent tens to hundreds of thousands on emergency response for an incident many did not know they were exposed to until the vulnerability was publicly disclosed[14]

A single major incident can exceed years of investment in technical capability. Institutions cannot afford to skip building technical capability within TPRM—the real risk is what happens without it.

Regulatory guidance, framework evolution, and vendor transparency will take years to materialize fully. But financial institutions cannot wait for the ecosystem to catch up. The next chapter addresses the one action every institution can take immediately: integrating technical capability into TPRM functions.

Endnotes - Chapter 10

[1] Federal Financial Institutions Examination Council (FFIEC), "IT Examination Handbook - Outsourcing Technology Services," revised August 2024, https://ithandbook.ffiec.gov/it-booklets/outsourcing-technology-services

[2] Office of the Comptroller of the Currency (OCC), Federal Reserve, Federal Deposit Insurance Corporation (FDIC), "Interagency Guidance on Third-Party Relationships: Risk Management," OCC Bulletin 2023-17 / Federal Reserve SR 23-4 / FDIC FIL-23-2023, June 6, 2023, https://www.occ.gov/news-issuances/bulletins/2023/bulletin-2023-17.html

[3] New York State Department of Financial Services, "Cybersecurity Requirements for Financial Services Companies," 23 NYCRR Part 500, effective March 1, 2017

[4] European Union, "Regulation (EU) 2022/2554 of the European Parliament and of the Council

on digital operational resilience for the financial sector (DORA)," Official Journal of the European Union, December 27, 2022, L 333, pp. 1–79; application date January 17, 2025

[5] European Insurance and Occupational Pensions Authority (EIOPA), "Digital Operational Resilience Act (DORA)," https://www.eiopa.europa.eu/digital-operational-resilience-act-dora_en

[5.1] European Commission rejection of DORA RTS Article 5 (ongoing subcontractor chain monitoring) documented in: Commission Delegated Regulation supplementing Regulation (EU) 2022/2554, final text (December 2024) compared with July 2024 draft RTS. The rejection was finalized prior to DORA's January 2025 application date.

[6] National Institute of Standards and Technology (NIST), "Secure Software Development Framework (SSDF)," NIST SP 800-218, February 2022, https://csrc.nist.gov/publications/detail/sp/800-218/final

[7] Secureframe, "The History of SOC 2," https://secureframe.com/hub/soc-2/history

[7.1] GDPR Article 28 (effective May 25, 2018) created binding requirements for data processing agreements between controllers and processors. Enterprise adoption of DPAs in financial services was primarily driven by GDPR compliance requirements for organizations processing EU citizen data, predating CCPA (effective January 1, 2020) by approximately 20 months.

[7.2] Software supply chain attack cost projections: Juniper Research, "Software Supply Chain Attacks to Cost the World $46 billion in 2023" (2023); Cybersecurity Ventures, "Software Supply Chain Attacks to Cost Global Economy $138 Billion by 2031" (2023). These projections include direct remediation costs, incident response, business disruption, and reputational damage.

[7.3] GitHub, "About the dependency graph," GitHub Docs; GitLab, "Dependency Scanning," GitLab Docs. Open-source tools: Anchore Syft (https://github.com/anchore/syft) and Aqua Security Trivy (https://github.com/aquasecurity/trivy) documentation.

[8] American Institute of Certified Public Accountants (AICPA), "2017 Trust Services Criteria (With Revised Points of Focus – 2022)," September 2022, https://www.aicpa-cima.com/resources/download/2017-trust-services-criteria-with-revised-points-of-focus-2022

[9] American Institute of Certified Public Accountants (AICPA), "SOC for Supply Chain," introduced March 2020, https://www.aicpa-cima.com/topic/audit-assurance/audit-and-assurance-greater-than-soc-for-supply-chain

[10] Johanson Group, LLP, "The History of SOC 2 Compliance," https://www.johansonllp.com/blog/the-history-of-soc-compliance (describing AICPA's consensus-driven standards development process)

[10.1] American Institute of Certified Public Accountants (AICPA), "SOC for Cybersecurity," https://www.aicpa-cima.com/topic/audit-assurance/audit-and-assurance-greater-than-soc-for-cybersecurity. SOC for Cybersecurity examinations, introduced in 2017, provide a framework for organizations to communicate information about their cybersecurity risk management program to stakeholders, distinct from but complementary to SOC 2 attestations.

[11] Schellman, "Which Big 4 Auditing Firm Should Perform Your SOC Audit?" https://www.schellman.com/blog/so examinations/which-audit-big-4-should-perform-your-soc-audit (providing industry cost benchmarks for SOC 2 audits)

[11a] Cost analysis based on: Schellman SOC 2 benchmarks ($20K-$80K); Chainguard, "The True Cost of SLSA Compliance" (2024) estimating $30K-$120K for SLSA Level 2+ implementation;

Sonatype, "State of the Software Supply Chain Report" (2024) on dependency monitoring platform costs ($15K-$50K annually); and PwC/Deloitte consulting rate cards for specialized supply chain security assessments ($200-$400/hour, 100-300 hours for comprehensive attestation).

[11.1] Practical DevSecOps, "Certified Software Supply Chain Security Expert (CSSE)" program documentation; SANS Institute, "Software Supply Chain Security" curriculum offerings.

[12] Compliance Week, "Shades of SolarWinds in Lessons from MOVEit Hack," June 2023, https://www.complianceweek.com/cybersecurity/shades-of-solarwinds-in-lessons-from-moveit-hack/33211.article

[13] Wikipedia, "2024 CrowdStrike-related IT outages," https://en.wikipedia.org/wiki/2024_CrowdStrike-related_IT_outages

[14] Cybersecurity and Infrastructure Security Agency (CISA), "Review of the December 2021 Log4j Event," CSRB Report, July 2022, https://www.cisa.gov/sites/default/files/publications/CSRB-Report-on-Log4-July-11-2022_508.pdf

Chapter 11: Closing the Capability Gap

The multi-stakeholder solution described in Chapter 10 requires years to fully materialize. Regulatory guidance takes time to develop and promulgate. Assurance frameworks evolve through consensus processes measured in standards-body cycles. Fintech innovation responds to market demand signals that may take quarters or years to become clear.

Financial institutions cannot wait for the ecosystem to evolve. The risks are present today; the incidents are occurring now. There is one action every institution can take immediately, regardless of external stakeholder progress: **integrate technical expertise into third-party risk management functions**.

The Technical Capability Gap

The demand for technical understanding in third-party risk management has never been greater—and it is not going away. The highest-risk items that TPRM teams oversee are not office supply vendors or facilities management contracts; they are:

- **SaaS platforms** running core business functions
- **Cloud infrastructure providers** hosting applications and data
- **Managed security service providers** operating security operations and endpoint detection
- **Cybersecurity tools** with deep system access
- **Payment processors and fintech integrations** with direct access to customer financial data
- **Core banking platforms and middleware** supporting critical operations

Every one of these vendor categories requires technical evaluation that exceeds the capabilities of traditional risk management training. A professional with expertise in contract review, operational risk assessment, and SOC 2 control evaluation is not equipped to interpret a software bill of materials, evaluate CI/CD pipeline integrity controls, or assess the implications of a vendor's infrastructure architecture choices.

These are not edge cases. They are routine questions that arise during vendor assessments, incident response, and ongoing monitoring. Without technical capability, TPRM teams either ignore these questions (leaving risk unaddressed), rely on vendor representations without independent validation (creating false assurance), or escalate every technical question to overloaded IT or InfoSec teams (creating bottlenecks and friction).

The Economic Case

The cost of technical capability is modest compared to the cost of incidents it helps prevent:

- A **single major vendor breach** can result in remediation costs, regulatory examination, reputational damage, and potential enforcement actions totaling millions of dollars
- **Operational disruption** from unidentified vendor dependencies (like CrowdStrike) can halt business operations for hours or days, with estimated global financial damage exceeding $10 billion[1]
- **Emergency response** to widespread vulnerabilities (like Log4j) requires expensive weekend mobilization of technical teams across the institution

Technical capability for TPRM costs approximately the same as mid-tier vendor management software subscriptions that many institutions already pay.[4] The return on investment is avoiding

blind spots that lead to incidents, and improving the quality of vendor risk assessments to provide actual assurance rather than false comfort.

Furthermore, technical capability enables TPRM to reduce reliance on external consultants for vendor technical assessments, often achieving positive ROI within the first year through reduced consulting spend.[5]

Why This Exceeds Current Regulatory Requirements—And Why That Matters

Regulators have not explicitly mandated that TPRM teams include technical subject matter experts. But regulatory compliance is a floor, not a ceiling. The goal of TPRM is not to check boxes for examiners; it is to identify and mitigate third-party risks that could disrupt operations, compromise data, or harm customers.

Recent incidents demonstrate that traditional TPRM approaches are insufficient:

- **Log4j**: Institutions with fully compliant TPRM programs were blindsided by a vulnerability in a code library embedded throughout their vendor portfolios—because dependency-level risks were never assessed.[2]
- **SolarWinds**: Vendors with passing SOC 2 audits suffered build pipeline compromises that traditional controls did not detect—because software delivery mechanisms were out of scope.
- **CrowdStrike**: Institutions using a widely trusted cybersecurity vendor experienced widespread outages from a routine update—because no one evaluated update deployment controls as a risk dimension.[1]

Regulatory requirements reflect past incidents and established best practices. They necessarily address known risk categories rather than emerging ones—this is inherent to how deliberate regulatory processes work. Institutions that wait for explicit regulatory mandates before building necessary capabilities are perpetually reactive, responding to incidents rather than preventing them.

Moreover, regulatory expectations are evolving. DORA in Europe explicitly addresses ICT third-party risk with technical depth. FFIEC guidance now references NIST secure software development frameworks.[3] The trajectory is clear: regulators increasingly expect technical sophistication in third-party risk management. Building capability now positions institutions ahead of the curve rather than scrambling to catch up after the next examination finding.

Implementation Approaches

How institutions integrate technical capability into TPRM will vary based on size, complexity, and risk exposure—the same "commensurate with" principle that governs all risk management investment decisions. Options range from dedicated technical analyst roles within TPRM, to formalized partnerships with Information Security or IT Architecture teams, to structured escalation frameworks that route technical questions appropriately.

The specific model matters less than the outcome: ensuring that someone with genuine technical expertise is involved in evaluating the vendors that present technology-driven risks. Each institution must determine the approach appropriate to its circumstances.

What Technical TPRM Capability Looks Like in Practice

For institutions considering dedicated technical roles within TPRM, here are concrete elements to consider:

Role Profile: Third-Party Technology Risk Analyst

Core responsibilities:

- Review vendor SBOMs and correlate against vulnerability databases
- Evaluate CI/CD pipeline security practices during critical vendor assessments
- Translate technical findings into risk language for TPRM reports
- Monitor CVE disclosures affecting the vendor portfolio
- Conduct technical deep-dives during incident response

Background characteristics (not all required):

- Software development or DevOps experience
- Cloud architecture familiarity (AWS, Azure, GCP)
- Understanding of container and Kubernetes environments
- Security certifications (CISSP, cloud security credentials) helpful but not essential
- Ability to read and interpret technical documentation

Screening Questions for Candidates

Technical depth:

- "Walk me through how you would assess whether a vendor is vulnerable to a Log4j-type incident."
- "What would you look for in a vendor's SOC 2 report to understand their software delivery controls?"
- "If a vendor told you they 'use containers,' what follow-up questions would you ask?"

Risk translation:

- "How would you explain CI/CD pipeline risk to a board member?"
- "A vendor says they 'practice DevSecOps.' What does that actually tell you about their security posture?"

Essential Tools and Skills

- SBOM analysis: Familiarity with CycloneDX/SPDX formats, tools like Dependency-Track or Snyk
- Vulnerability correlation: NVD, vendor security advisories, CVE databases
- Cloud architecture: Understanding shared responsibility models, IAM, network segmentation
- Container security: Basic understanding of image scanning, runtime protection, orchestration risks
- SOC 2 interpretation: Ability to read Section III scoping and evaluate carve-outs critically

What is not optional is recognizing the capability gap and taking deliberate action to close it. The incidents of recent years have demonstrated that traditional TPRM competencies—contract review, financial analysis, SOC 2 interpretation, business continuity assessment—are necessary but insufficient for managing the risks posed by modern technology vendors.

Building capability is an ongoing process. The final chapter provides diagnostic questions to assess current maturity and identify priority gaps.

Endnotes - Chapter 11

[1] CrowdStrike incident analysis: The outage on July 19, 2024 caused approximately 8.5 million Windows devices to crash, with worldwide financial damage estimated at $10 billion or more. Fortune 500 companies faced $5.4 billion in direct losses according to Parametrix study. Sources: Wikipedia, "2024 CrowdStrike-related IT outages"; American Banker, "Tech issues afflict banks, Microsoft after critical CrowdStrike glitch"; Fortune, "CrowdStrike outage will cost Fortune 500 companies $5.4 billion."

[2] Log4j incident analysis: The vulnerability disclosed in December 2021 affected hundreds of millions of systems through transitive dependencies organizations did not know they had. 60% of affected Java applications used Log4j as a transitive dependency invisible to standard security assessments. Organizations with Software Bills of Materials reduced remediation time from weeks to hours. Sources: Apache Foundation disclosure; CISA Emergency Directive 22-02; NY DFS Industry Letter, December 17, 2021.

[3] FFIEC IT Examination Handbook, "Development and Acquisition" booklet (August 2024) references NIST Secure Software Development Framework (SSDF) as guidance for software development security practices.

[4] Cost comparison based on: Mid-tier TPRM platform subscriptions (ProcessUnity, Prevalent, OneTrust) range from $50,000-$150,000 annually for 100-500 vendor portfolio according to G2 and Gartner Peer Insights reviews (2023-2024). A dedicated Technical Risk Analyst role (salary $85K-$130K depending on market and experience per Glassdoor/Bureau of Labor Statistics) or fractional IT allocation (0.25-0.5 FTE, $40K-$70K cost) falls within comparable range to software subscriptions already in TPRM budgets.

[5] External consultant cost avoidance analysis: Specialized vendor technical assessments from Big 4 or boutique security firms cost $15,000-$50,000 per engagement (Deloitte, PwC, Coalfire rate cards). Financial institutions conducting 5-10 deep technical vendor assessments annually through external consultants spend $75K-$500K. Internal technical capability reduces this to $10K-$100K in residual consulting for highly specialized assessments, generating $65K-$400K in annual savings that offset dedicated headcount costs within 12-18 months.

Chapter 12: What Mature Dependency Risk Management Looks Like

The preceding chapters have diagnosed the framework gap: the mismatch between entity-focused oversight and dependency-level risk propagation. But diagnosis without direction is incomplete. What does mature dependency risk management actually look like in practice?

Rather than prescribing specific action plans—which vary by institution size, risk appetite, and organizational structure—this chapter frames the question differently: **What should a mature TPRM program be able to answer?**

The questions below map to the three risk dimensions established throughout this analysis. An institution that can answer these questions has achieved meaningful visibility into dependency-level risk. An institution that cannot answer them has work to do—regardless of how well its traditional vendor management processes perform.

This chapter does not prescribe specific implementation steps—institutions will build these capabilities differently based on their size, complexity, and risk appetite. Instead, it offers diagnostic questions that reveal where current capabilities provide genuine visibility and where blind spots remain. Within each section, questions are ordered from those answerable with existing resources to those requiring new capabilities or vendor cooperation.

Supply Chain Risk: What's in the Code

These questions address the embedded components, libraries, and dependencies within vendor software—the layer where Log4j-type vulnerabilities propagate invisibly through transitive dependencies.

If a critical vendor's subcontractor experienced a breach affecting your data, how would you expect to learn about it—and do your contracts support timely notification?

The MOVEit incident revealed that many organizations learned of their exposure from news media or the Cl0p ransomware gang's leak site rather than from their vendors. Some received forensic details two months after initial exploitation. Contractual notification requirements typically address direct vendor breaches but may not cascade to nth-party incidents where your vendor is also a victim. Mature programs have mapped notification pathways for critical dependency chains and have contractual language requiring vendors to notify them of upstream breaches affecting their data—not just breaches of the vendor's own systems. This assessment requires only reviewing existing contracts—no new vendor cooperation needed.

Can you identify which open source components are embedded across your critical vendor portfolio?

This is the SBOM question. Mature programs can produce a consolidated view of what code libraries their critical vendors use, either through vendor-provided SBOMs or through systematic inquiry. Less developed programs know only that vendors exist—not what's inside them. Answering this question requires vendor cooperation in providing component inventories.

If a Log4j-scale vulnerability emerged tomorrow, how quickly could you determine vendor exposure?

When the Log4j vulnerability (CVE-2021-44228, detailed in Chapter 7) was disclosed, institutions with dependency visibility identified exposure within hours. Institutions without it spent weeks surveying vendors manually—and some never achieved complete visibility.[1] Another Log4j will happen—the question is when. Response time is a function of preparation—specifically, whether you have collected and can query component inventories before an incident occurs.

For vendors that provide SBOMs, do you have the capability to ingest and analyze them?

Collecting SBOMs is meaningless without the ability to interpret them. Mature programs can correlate vendor SBOMs against vulnerability databases, identify concentration on specific components, and surface emerging risks. Traditional programs collect documents they cannot use. This capability requires tooling investment beyond vendor cooperation.

Do you track transitive dependencies, or only direct vendor relationships?

As detailed in Chapter 7, the majority of Log4j exposure existed through transitive dependencies—meaning applications inherited the library through other components rather than declaring it directly.[2] A program that tracks only direct vendor relationships misses the majority of supply chain risk. Understanding transitive dependencies requires sophisticated SBOM analysis capabilities.

For your critical vendors, do you understand whether their high-risk dependencies are direct or transitive—and how that affects remediation timelines?

When Log4j was disclosed, organizations with direct Log4j dependencies could patch immediately. Organizations whose vendors used Log4j transitively—through other libraries that themselves depended on Log4j—faced longer remediation timelines because patches required upstream library maintainers to update first, then vendors to incorporate those updates, then customers to deploy. Transitive dependencies create remediation chains that extend timelines from days to weeks or months. Understanding dependency depth for critical components informs realistic expectations for vendor response. This represents the most sophisticated level of supply chain visibility.

Software Delivery Risk: How Updates Reach You

These questions address the mechanisms through which vendor software is built, signed, and deployed—the layer where SolarWinds and 3CX-type compromises occur.

Can you distinguish between vendors' SOC 2 infrastructure coverage and application-layer coverage?

Many SOC 2 reports cover "corporate IT systems" or "infrastructure" without explicitly including the customer-facing application. A mature program scrutinizes Section III of every SOC 2 report to determine what's actually in scope. A less sophisticated program treats "has SOC 2" as binary assurance.[4] This assessment requires only reading reports you already have—no vendor cooperation needed.

Do you know which vendors push automatic updates versus customer-controlled deployment?

The CrowdStrike incident affected organizations precisely because updates deployed automatically without customer testing windows.[3] Understanding whether you control update timing—or whether

the vendor does—is fundamental to assessing delivery risk. This information can be gathered through standard vendor inquiry.

For critical vendors, do you understand their software release and testing processes?

CrowdStrike's "Rapid Response Content" followed different testing protocols than standard software releases.[3] Understanding a vendor's release process—including what gets expedited testing and what does not—reveals risk that questionnaire-based assessments miss. This requires asking vendors questions that go beyond standard questionnaires.

Do you assess vendor CI/CD pipeline security, or only production environment controls?

SolarWinds had production security controls. The compromise occurred in the build environment—upstream of everything traditional assessments examine. Mature programs ask about build environment isolation, artifact signing, and deployment controls. Early-stage programs assess only where code runs, not how it gets there. Assessing CI/CD security requires technical questions that most vendors are not accustomed to answering.

For vendors with privileged or kernel-level system access, do you control when updates deploy to your environment?

The CrowdStrike incident affected organizations precisely because they followed security best practices—deploying trusted updates automatically from a trusted vendor. The 78-minute window between defective update release and rollback was sufficient to crash millions of systems globally. Organizations that maintained staging environments or delayed deployment windows for even critical security tools had time to observe failures before production impact. The question is not whether you trust the vendor, but whether that trust includes ceding control over deployment timing for software that can render systems inoperable. Implementing deployment controls for privileged software requires infrastructure and process changes within your own organization.

Infrastructure Concentration Risk: Where It All Runs

These questions address the underlying platforms, providers, and services that vendors depend on—the layer where concentration creates correlated failure risk across apparently diverse vendor portfolios.

For your critical vendors, do you know which have multi-region or multi-cloud resilience versus single points of failure?

Vendor business continuity assessments typically confirm that disaster recovery plans exist, and many SOC 2 reports disclose whether a vendor operates across multiple regions or cloud providers. This information is often available in existing attestations—the challenge is systematically extracting and aggregating it across the portfolio to identify which vendors have genuine geographic and platform diversity versus those whose "business continuity" depends on a single infrastructure remaining available. This assessment can begin with reviewing attestations you already have.

Do you distinguish between vendors and platforms in your risk framework?

AWS is both a vendor (a legal entity you may contract with) and a platform (infrastructure that many of your other vendors share). This distinction matters for concentration analysis. Mature programs track both dimensions. Less evolved programs treat everything as a vendor relationship.

This is a conceptual framework adjustment that requires no external data—only internal alignment on how to categorize and track dependencies.

What percentage of your critical vendor portfolio runs on AWS versus Azure versus GCP?

An institution with 100 "diverse" vendors that all run on AWS has significant concentration risk on a single infrastructure provider—one with whom the institution has no direct contractual relationship. Mature programs map this concentration. Traditional programs see only the vendor layer. This information can be gathered through vendor inquiry during due diligence or renewal.

How many of your vendors share the same EDR platform, identity provider, or CDN?

CrowdStrike is deployed across 298 of the Fortune 500.[5] Okta provides identity services for over 19,400 customers including major enterprises.[5.1] Cloudflare handles approximately 20% of all web traffic.[5.2] Concentration on these platforms creates correlated risk that vendor-by-vendor assessment does not reveal. Like cloud provider mapping, this requires asking vendors about their technology stack.

If a major cloud region experienced an extended outage, how many of your critical vendors would be simultaneously affected?

Geographic concentration compounds platform concentration. A mature program can model the blast radius of infrastructure failures. A reactive program discovers concentration only during incidents. This requires aggregating the cloud provider and region data collected from individual vendors into a portfolio-level view.

Do you assess vendor market penetration across financial services—not just "are we dependent on this vendor" but "are we and our peers dependent"?

Traditional concentration risk asks whether your institution is over-reliant on a single vendor. But the collective fragility paradox operates at industry level: when 60% of the Fortune 500 uses the same endpoint security platform, or the majority of financial services runs on three cloud providers, individual institution diversification is insufficient. A vendor's operational failure becomes a sector-wide systemic event. Mature programs track not only their own vendor dependencies but the industry's collective convergence on critical platforms—because correlated failure across peers creates systemic risk that individual assessments cannot detect. This requires access to industry-level data beyond what individual vendor assessments provide.

The Maturity Progression

Institutions will find themselves at different points along the maturity spectrum for each risk dimension. Few organizations can answer all these questions comprehensively today—the capability gap is real, and closing it takes time.

The value of this framework is not as a pass/fail test, but as a diagnostic tool. It identifies where existing capabilities provide genuine visibility and where blind spots remain. It frames dependency risk management not as a compliance checklist but as an ongoing capability-building exercise.

For institutions beginning this journey, the honest answer to many of these questions may be "we do not know." That recognition is the first step. The incidents documented in this paper demonstrate

the cost of not knowing. Will institutions build the capability to answer these questions before the next incident—or after?

Endnotes - Chapter 12

[1] Log4j response timelines documented in multiple industry analyses (see Chapter 7 for detailed statistics). Organizations with SBOM capability identified exposure within hours; organizations without SBOM capability spent weeks on manual vendor surveys. See: CISA, "Apache Log4j Vulnerability Guidance," December 2021.

[2] For detailed Log4j transitive dependency statistics, see Chapter 7. Snyk analysis: "A straightforward search to determine if you're using a vulnerable version of Log4j would not necessarily find all occurrences in your projects" due to transitive dependencies. Source: Snyk, "Log4j Vulnerability Explained," December 2021.

[3] CrowdStrike post-incident analysis: Rapid Response Content updates deploy automatically to address emerging threats, following different testing protocols than standard Sensor Content updates. The July 19, 2024 incident resulted from a configuration file update that bypassed full testing procedures. Source: CrowdStrike, "Preliminary Post Incident Report," July 2024.

[4] SOC 2 scoping analysis: Service organizations determine which systems fall within audit scope based on "service commitments and system requirements." A vendor can legitimately scope SOC 2 to cover internal IT operations while excluding customer-facing applications—and remain compliant with AICPA standards. Source: AICPA, "SOC 2 Reporting on an Examination of Controls at a Service Organization."

[5] CrowdStrike market penetration: "Over half of Fortune 500 companies trust CrowdStrike" (298 of 500). Source: CrowdStrike corporate communications, June 2024; NotebookCheck, "How and why CrowdStrike has a massive market share," 2024.

[5.1] Okta, Inc. SEC Form 10-K filing (FY 2024) reports over 19,400 customers. Okta's enterprise identity platform is used by organizations including JetBlue, Nordstrom, Siemens, Slack, T-Mobile, and Zoom. See: Okta, "Customers," https://www.okta.com/customers/

[5.2] Cloudflare, Inc. states that its network handles approximately 20% of all web traffic. See: Cloudflare, "About Cloudflare," https://www.cloudflare.com/about-overview/; W3Techs, "Usage statistics of Cloudflare," December 2024.

This paper represents independent analysis and is not an official position of any organization.