Supply Chain Security

SL5 Novel Recommendations

NOVEMBER 2025

Lisa Thiergart

Yoav Tzfati

Peter Wagstaff

Luis Cosio

Guy

Philip Reiner

Overview

Executive Summary

The AI supply chain presents a major attack surface for frontier training operations [1]–[3]. Software supply chain attacks are among the cheapest and most scalable attacks for an external individual or nation-state to execute. While hardware attacks are more costly to orchestrate, they are feasible for well-resourced nation-state attackers at **Offensive Capability Level 5 (OC5)** [4], [5].

Whereas SL4 can plausibly be reached incrementally, SL5 can likely only or at least most quickly and cheaply be reached by a **radical reduction in the hardware and software stack that is trusted**, as well as a reduction of the volume of code that interfaces with critical components [6], [7] or is necessary for critical actions (this term for instance includes any processes touching model weights, hardware and software that trust is deferred to, etc.).

Since onshoring the chip supply chain in the coming years is particularly unrealistic, we think this can best be mitigated by investing in compensating controls, such as **investing in a strong, ideally on-chip, hardware root of trust in the next generation of AI chips** and exploring confidential computing and other chip security interventions mentioned in the machine security memo. These could include features such as those developed as part of the OpenTitan [8]–[13] project, though different vendors might have different preferred ways of implementing similar features. Other parts of the hardware supply chain can perhaps be audited and procured from trusted vendors [14], as discussed in the recommendations below.

Achieving a radical reduction in the software stack in the next three years without incurring unrealistic staffing expenses will involve a strong **reliance on untrusted Large Language Models (LLM) labor**[1] (i.e., LLMs we are not confident will not sabotage, self-exfiltrate or malfunction). There is a need to **differentially accelerate both the development of these defensive security capabilities** as well as **developing and deploying a mixed portfolio of oversight techniques [15]–[17] to guard the use of these models**. While the personnel memo discusses monitoring and oversight in particular, this memo calls for the differential acceleration of various defensive-dominant security tooling including how to safely use an SL4 model to accelerate the development of SL5 security infrastructure. We propose 5 actionable recommendations for AI labs and the AI industry to begin bridging the delta between SOTA and what is necessary to reach SL5.

Top 5 Recommendations

1. Minimize attack surface in critical software systems via architectural isolation and progressive access restriction
2. Review, prune, refactor, rewrite, or formally verify all external security-critical software
3. Review, minimize and audit the hardware supply chain for security-critical components
4. Develop and deploy adversarially robust systems to verify data integrity

[1] For SL5 planning, we suggest treating LLM assistance as untrusted-by-default in weight-adjacent workflows. This risk posture will be revisited in further work as the trust criteria of models in coding evolve. See related work [45], [46].

5. Continuously red team all supply chain processes

Minimize attack surface in critical software systems via architectural isolation and progressive access restriction

Explanation of Recommendation

Labs should establish architectural boundaries between systems based on their access to sensitive resources like frontier model weights, and where applicable architect these as rings of hardening [20]. Components with direct access to these resources must run with minimal, highly trusted software dependencies, while less sensitive systems can retain broader functionality. Access to sensitive resources should be provided only through **safe, narrow APIs that perform specific operations** (e.g., fine-tuning, quantization, inference) rather than allowing direct resource manipulation. This enables critical operations to run with a minimal software stack containing only essential, hardened components, while the broader R&D ecosystem—with its necessary but less-trusted dependencies—operates in isolated environments without direct access. Labs should **implement approval pipelines that assess new dependencies based on their proximity to sensitive resources (and ability to perform critical actions), security practices, codebase size, and necessity** [7], [19]. Additionally, **implementing APIs** can aid the implementation of progressive access restriction: **labs should minimize the number of staff, agents, services and other components with access to sensitive environments** (e.g., by establishing strict access quotas and reducing these over time, called "**seat management**" (progressively reducing and time-bounding privileged seats), which has been demonstrated by Cloudflare [21] and by an analogous process used by Google [20].

Research Reasoning

Every external package creates attack vectors through its code and transitive dependencies [18]. Our preliminary mapping identified hundreds of top-level software components across varying access levels, each bringing numerous transitive dependencies. The full ML software stack attack surface is likely infeasible to secure for weight-accessing systems. The solution combines architectural separation with aggressive dependency reduction—weight-accessing processes run minimal, self-sufficient code while R&D, monitoring, and CI/CD operate in separate environments with appropriate isolation.

Labs should implement multiple layers of separation—software-level API boundaries, physical host isolation, and network-level controls (as detailed in our Network Security recommendations)—to create defense-in-depth. Each additional layer reduces the probability that a compromised dependency could cause critical damage, allowing proportional reduction of untrusted dependencies based on each subsystem's risk level.

Labs should begin this transformation now by architecting systems that separate components with weight access from those without, enabling different dependency requirements for each subsystem. In parallel, they should eliminate existing dependencies in critical subsystems. Key areas requiring attention include operating systems, ML frameworks, and containerization/orchestration tools [22], [23]. The earlier labs establish these boundaries and reduce dependencies, the smoother their SL5 transition will be—early investment will compound to enable faster adaptation when needed.

Cost-Benefit

- **Costs:** Engineering effort for both architectural separation and dependency reduction, plus operational overhead from maintaining separate subsystems with different dependency sets. Engineers on critical components lose access to many external packages, requiring custom implementations.
- **Benefits:** Enables risk-proportionate attack surface reduction by allowing weight-accessing systems to run with minimal dependencies while less critical systems retain necessary functionality. Dramatically reduces vulnerability exposure to the most sensitive components without completely sacrificing the productivity benefits of external packages in lower-risk areas.
- **Alternatives:** Accepting current dependency chains (massive attack surface); attempting reduction without architectural separation (limited effectiveness); or deferring until SL5 transition (more disruptive, might be too late).

Review, prune, refactor, rewrite, or formally verify all external security-critical software

Explanation of Recommendation

We recommend that AI labs apply risk-proportionate hardening techniques to all external software with access to model weights, outputs, or critical infrastructure. This includes both non-AI and AI-assisted techniques such as **vulnerability scanning, feature pruning, security refactoring, language rewriting (e.g., C to Rust), and formal verification**, based on the software's criticality and AI capabilities available at implementation time. Security-critical software includes components with access to model weights, model outputs (which could enable distillation) [24], [25], CI/CD tools that affect these systems, and network access to weight storage. Labs should classify external dependencies by their security properties—trust level, codebase size, test coverage, language safety characteristics, and production criticality—then apply appropriate hardening based on this classification.

Research Reasoning

This tiered approach leverages expected advances in AI-assisted software hardening capabilities while accounting for uncertainties. We anticipate decreasing software volumes at higher security levels in data centers, making stricter interventions feasible for the most critical components. Labs will need to carefully architect their systems to support clean security boundaries between tiers and analyze the security properties of each import.

The effectiveness depends on **two key unknowns: the maturity of AI capabilities for tasks like secure language rewriting and formal verification assistance** when needed, and the actual **scope of software requiring these interventions**. Labs should start investing in and differentially accelerating these neglected cyber-defense capabilities. While some benchmarks exist for certain capabilities, there appear to be **limited well-established evaluations specifically for secure-language rewriting or formal-verification assistance**; recent system/model cards and surveys suggest these capabilities may **not yet be actively optimized** [26]–[31].

Labs that invest now in developing these capabilities will be able to apply stricter processes to more software. Since LLM outputs may not be trustworthy [15], [16], additional controls may be necessary when applying these procedures. See our personnel security document for such controls.

Cost-Benefit

- **Costs:** Engineering time for hardening processes, compute resources for AI-assisted analysis, and the risk of introducing bugs during rewrites that cause system malfunctions.
- **Benefits:** Optimizes the security-functionality trade-off by applying resource-intensive interventions like formal verification only to the most critical components while using lighter-weight processes like pruning for less sensitive software. Enables use of external dependencies while mitigating sophisticated backdoors and vulnerabilities that would otherwise remain undetected.
- **Alternatives:** Applying only automated vulnerability scanning and manual review (leaves sophisticated backdoors undetected); completely avoiding external dependencies (impractical given the scope of required software); relying solely on sandboxing and isolation (provides defense-in-depth but does not eliminate vulnerabilities); or accepting full codebases as-is (unacceptable risk for nation-state threats).

Review and audit the hardware supply chain for security-critical components using a trusted, diverse supplier base

Explanation of Recommendation

AI labs should review and regularly audit their hardware suppliers for security-critical components, while maintaining a **diverse supply base of trusted sources for those components to eliminate single points of failure**. The intent is not broad vendor diversity. It is controlled by diversity within a high-assurance supplier program. For SL5, labs should (1) define which components are security-critical, (2) set a minimum diversity requirement per critical component category (for example, at least two independent trusted sources or supply paths), and (3) limit procurement to a small set of pre-qualified suppliers that can support deep vetting, high transparency, and recurring audits. Chosen suppliers must regularly pass security audits, and in some cases should be assessed against relevant SL5 personnel security controls.

Research Reasoning

Most current training and inference stacks rely on components sourced from an extensive list of vendors, some of which have hard-to-verify provenance or known vulnerabilities (including Intel, AMD, etc.) [32]-[34]. A very broad vendor base increases exposure because it is difficult to maintain deep, continuous visibility into every supplier.

At the same time, over-consolidating on a single supplier creates single points of failure and increases the impact of a disruption or compromise. On balance, controlled diversity improves security because it reduces correlated failures and complicates adversary targeting, but only if the diverse sources are deeply trusted and continuously monitored. Unlike other areas of AI security which are predominantly defense advantaged, supply chain security is arguably offense advantaged, making it a field which differentially benefits from utilizing the strategy of **"security by obscurity"** (ie, sacrificing the security posture's simplicity for the benefit of obfuscating the posture to the attacker) [49].

DoDI 5200.44 [47] explicitly emphasizes minimizing supply chain risk to ensure uncompromised systems, and it relies on all-source intelligence analysis of suppliers. That depth of vetting is operationally infeasible across a broad, diverse vendor base. Therefore, for SL5 we recommend a **consolidated, high-assurance supply chain model that still preserves diversity for critical components by using a small number of trusted suppliers per component category**, similar in spirit to a "Trusted Supplier" approach. As procurement decisions are made for SL5 data centers and endpoints, prefer components where at least two trusted sources can be sustained (such as switches, NICs, cables, CPUs, and other replaceable infrastructure). Where possible, establish **agreements that improve visibility and auditability**, including **access to key design artifacts** needed to verify delivered hardware. Combine this with recurring and **randomized audits designed to detect sophisticated compromise**, including targeted deep inspections and sampled destructive evaluation when warranted.

This approach is consistent with SP 800-161 Rev. 1 guidance [48] on supply base diversification, which emphasizes diversification as a means to eliminate single points of failure for critical components. SP 800-161 explicitly couples diversification with strong supplier due diligence, upstream dependency analysis, and continuous monitoring, recognizing that unmanaged diversity can increase risk by overwhelming an organization's ability to perform deep vetting

Cost-Benefit

- **Costs:** Higher qualification and audit overhead, smaller pool of eligible suppliers, potential price increases, added engineering work to support two or more approved parts, and trusted suppliers may become more attractive targets.
- **Benefits:** Raises adversary cost by requiring compromise across multiple trusted supply paths instead of one, minimizes supply chain risk while avoiding single points of failure, increases resilience to disruption, improves detection rates through continuous monitoring and audits.
- **Alternatives:** Broad, open diversification across many vendors (makes all-source vetting and continuous assurance infeasible); strict single-supplier consolidation (creates single points of failure and increases impact, though would also simplify the implementation and enforcement of rigorous security controls); rely solely on post-delivery inspection without deep supplier vetting (misses sophisticated embedded compromises); or attempt complete onshoring of all components (infeasible in the near term given current manufacturing capabilities).

Develop and deploy adversarially robust systems to verify data integrity

Explanation of Recommendation

We recommend that AI labs develop and then deploy adversarially robust detection systems to **screen all textual content entering the data center**—including training data, inference inputs, and code undergoing LLM-based hardening—for **poisoning attempts, jailbreaks, and other malicious content**. This control requires screening of all text that could influence SL5 models or data center operations. Detection systems must identify backdoor attacks in training corpora, jailbreak attempts in inference data, and malicious patterns in code being reviewed or rewritten by LLMs. For code specifically, attackers could craft inputs that jailbreak the reviewing LLM to ignore vulnerabilities, insert backdoors, or introduce subtle sabotage. The **detection systems themselves must be hardened against adversarial attacks**, as sophisticated actors may craft inputs designed to bypass screening while compromising target systems.

Research Reasoning

Adversarial robustness remains an unsolved problem [17]. While production systems like Anthropic's Constitutional Classifiers [40] and OpenAI's safety systems for ChatGPT Agent [41] demonstrate initial capabilities, comprehensive detection for data poisoning does not yet exist [35], [36], [42], [43]. Nation-state actors could embed sophisticated backdoors that activate under specific conditions or craft inputs that appear benign while triggering unintended behaviors. **The scale challenge is acute**—SL5 training will require vast text corpora where **manual review is infeasible** [44]. Automated systems must maintain high detection rates while managing false positives. More research is needed to have trust in datasets that will be used for SL5. **Labs investing now in detection research will be better positioned when SL5 capabilities emerge**.

Cost-Benefit

- **Costs:** Computational overhead for screening (depending on implementation, potentially approaching a meaningful fraction of training costs), delays from false positives, and substantial R&D investment. Detection research timelines as well as feasibility of reaching a high accuracy remain uncertain.
- **Benefits:** Prevents sophisticated data poisoning attacks that could compromise model behavior, protects against jailbreak attempts that could bypass safety measures, and provides defense against malicious code injections during LLM-assisted hardening processes.
- **Alternatives:** Accept poisoning risk while relying on other defenses (leaves models vulnerable to sophisticated backdoors); radically reduce external data (severely limits training data diversity and model capabilities); manual review (infeasible at the scale required for SL5 training); or limit to old data from before ChatGPT era (poisoning attacks likely weren't happening yet, but significantly restricts available training data).

Continuously red team all supply chain processes

Explanation of Recommendation

AI labs should implement **automated systems that continuously inject benign adversarial actions into supply chain processes**. This includes occasionally passing intentionally vulnerable or backdoored code through the import process, poisoned data through screening systems, and software dependencies that should not be approved under the minimization paradigm [37]–[39]. Labs should use LLMs to generate diverse attack patterns and test vectors that simulate both basic security failures and sophisticated attacks. These automated systems should be designed in consultation with nation-state-level whitehat attackers to ensure appropriate levels of attack vectors are reflected in this process.

Research Reasoning

Unlike traditional red teaming which provides point-in-time assessments, continuous adversarial injection ensures constant readiness whilst generating metrics on detection effectiveness between formal red team exercises. This approach complements traditional red teaming by serving dual purposes: testing processes against basic attacks that could slip through due to human error or oversight, while also simulating sophisticated attacks that could compromise the supply chain. Critical to implementation is maintaining clear distinction between planned injections and real attacks to prevent adversaries from hiding actual compromise among simulated events.

Cost-Benefit

- **Costs:** Engineering effort to build automated injection systems and operational overhead from investigating benign alerts, with risk of alert fatigue if poorly calibrated.
- **Benefits:** Provides continuous validation of security processes between traditional red team exercises, generates metrics on detection effectiveness, and maintains constant readiness against both simple oversights and sophisticated attacks.
- **Alternatives:** Relying solely on periodic red team exercises (leaves long untested periods); purely reactive monitoring (first failure could be catastrophic); or using only external red teams without continuous testing (expensive and infrequent).

Bibliography

[1] Cybersecurity and Infrastructure Security Agency (CISA), "A Practical Guide to Supply Chain Compromises," 2025. https://www.cisa.gov/resources-tools/resources/practical-guide-supply-chain-compromises

[2] MITRE, "CVE-2024-3094," CVE Record, 2024. https://www.cve.org/CVERecord?id=CVE-2024-3094

[3] Red Hat, "Xz Backdoor Explained & Analysis (CVE-2024-3094)," 2024. https://www.redhat.com/en/blog/xz-backdoor-explained-and-analysis-cve-2024-3094

[4] Electronic Frontier Foundation, "20131230-Appelbaum—NSA ANT Catalog," 2013. https://www.eff.org/document/20131230-appelbaum-nsa-ant-catalog

[5] Kaspersky, "A virus in HDD firmware is real, what's next?," Feb. 2015. https://www.kaspersky.com/blog/equation-hdd-malware/7623/

[6] NIST, SP 800-160, Vol. 1 Rev. 1: Engineering Trustworthy Secure Systems, 2022. https://csrc.nist.gov/pubs/sp/800/160/v1/r1/final

[7] NIST Special Publication (SP) 800-53 Rev. 5.2.0, National Institute of Standards and Technology, Gaithersburg, MD, Nov. 2023. [Online]. Available: https://csrc.nist.gov/projects/cprt/catalog#/cprt/framework/version/SP_800_53_5_2_0/home

[8] lowRISC & partners, "OpenTitan®: Open source silicon root of trust," 2025. https://opentitan.org/

[9] NIST, SP 800-193: Platform Firmware Resiliency Guidelines, 2018. https://csrc.nist.gov/pubs/sp/800/193/final

[10] Confidential Computing Consortium, A Technical Analysis of Confidential Computing, v1.3, 2023. Available: https://confidentialcomputing.io/wp-content/uploads/sites/10/2023/03/CCC-A-Technical-Analysis-of-Confidential-Computing-v1.3_unlocked.pdf

[11] NVIDIA, "Confidential Compute on NVIDIA Hopper H100," Whitepaper, 2023. Available: https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/HCC-Whitepaper-v1.0.pdf

[12] AMD, "SEV-SNP: Strengthening VM Isolation with Integrity Protection and More," Whitepaper, 2020. https://www.amd.com/en/resources/amd-secure-encrypted-virtualization.html

[13] Intel, "Intel® Trust Domain Extensions (Intel® TDX): Architectural Overview," Whitepaper, 2024. https://www.intel.com/content/www/us/en/content-details/773875/intel-trust-domain-extensions-intel-tdx.html

[14] NIST, SP 800-161 Rev. 1 (with updates through 2024): Cybersecurity Supply Chain Risk Management Practices, 2022. https://csrc.nist.gov/pubs/sp/800/161/r1/upd1/final

[15] OWASP Foundation, "LLM01:2025 Prompt Injection," OWASP GenAI Security Project, 2025. https://genai.owasp.org/llmrisk/llm012025-prompt-injection/

[16] E. Hubinger et al., "Sleeper Agents: Training Deceptive LLMs that Persist Through Safety Training," arXiv:2401.05566, 2024. https://arxiv.org/abs/2401.05566

[17] A. Vassilev et al., "Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations," NIST AI 100-2e2025, Mar. 2025, doi:10.6028/NIST.AI.100-2e2025. https://csrc.nist.gov/pubs/ai/100/2/e2025/final

[18] OWASP Foundation, "LLM03:2025 Supply Chain," OWASP GenAI Security Project, 2025. https://genai.owasp.org/llmrisk/llm032025-supply-chain/

[19] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," NIST SP 800-207, Aug. 2020. https://csrc.nist.gov/pubs/sp/800/207/final

[20] Google Cloud, "How Google protects its production services," *Google Cloud*, Jun. 2024. [Online]. Available: https://cloud.google.com/docs/security/production-services-protection

[21] Cloudflare, "Seat management," *Cloudflare Zero Trust docs*. [Online]. Available: https://developers.cloudflare.com/cloudflare-one/identity/users/seat-management/

[22] M. Souppaya, K. Scarfone, and D. Dodson, "Secure Software Development Framework (SSDF) Version 1.1," NIST SP 800-218, Feb. 2022. https://csrc.nist.gov/pubs/sp/800/218/final

[23] OpenSSF, "Supply-chain Levels for Software Artifacts (SLSA) v1.0," 2024. https://slsa.dev/spec/v1.0

[24] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," arXiv:1503.02531, Mar. 2015. https://arxiv.org/abs/1503.02531

[25] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing Machine Learning Models via Prediction APIs," in Proc. USENIX Security, 2016. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer

[26] OpenAI, "GPT-5 System Card," Aug. 2025. https://openai.com/index/gpt-5-system-card/

[27] Anthropic, "Claude 4 System Card," 2025. Available: https://www-cdn.anthropic.com/4263b940cabb546aa0e3283f35b686f4f3b2ff47.pdf

[28] Google DeepMind, "Gemini 2.5: Technical Report," 2025. Available: https://storage.googleapis.com/deepmind-media/gemini/gemini_v2_5_report.pdf

[29] xAI, "Grok-4 Model Card," Aug. 2025. Available: https://data.x.ai/2025-08-20-grok-4-model-card.pdf

[30] M. Mohammadi et al., "Evaluation and Benchmarking of LLM Agents: A Survey," arXiv:2507.21504, 2025. https://arxiv.org/abs/2507.21504

[31] S. M. Xie et al., "CLEVER: A Curated Benchmark for Formally Verified Code Generation," arXiv:2505.13938, 2025. https://arxiv.org/abs/2505.13938

[32] Microsoft, "KB5029778: How to manage the vulnerability associated with CVE-2022-40982," 2023. https://support.microsoft.com/en-us/topic/kb5029778-how-to-manage-the-vulnerability-associated-with-cve-2022-40982-d461157c-0411-4a91-9fc5-9b29e0fe2782

[33] MITRE, "CVE-2022-40982," CVE Record, 2023. https://www.cve.org/CVERecord?id=CVE-2022-40982

[34] The Hacker News, "New Research Reveals Spectre Vulnerabilities Resurface," Oct. 2024. https://thehackernews.com/2024/10/new-research-reveals-spectre.html

[35] X. Zhang et al., "PoisonBench: Assessing Large Language Model Vulnerability to Data Poisoning," arXiv:2410.08811, 2024. https://arxiv.org/abs/2410.08811

[36] H. Li et al., "Data Poisoning in Deep Learning: A Survey," arXiv:2503.22759, 2025. https://arxiv.org/abs/2503.22759

[37] K. Shortridge and A. Rinehart, Security Chaos Engineering: Sustaining Resilience in Software and Systems, O'Reilly, 2023. https://www.oreilly.com/library/view/security-chaos-engineering/9781098113810/

[38] MITRE, "Caldera™: A Scalable, Automated Adversary Emulation Platform," 2025. https://caldera.mitre.org/

[39] Netflix, "Chaos Monkey," GitHub repository, 2012–2025. https://github.com/Netflix/chaosmonkey

[40] Anthropic, "Constitutional Classifiers," Feb. 2025. https://www.anthropic.com/news/constitutional-classifiers

[41] OpenAI, "Introducing ChatGPT Agent," Jul. 17, 2025. https://openai.com/index/introducing-chatgpt-agent/

[42] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, "Poisoning Web-Scale Training Datasets is Practical," in Proc. 2024 IEEE Symposium on Security and Privacy (SP), 2024, pp. 407–425. doi: 10.1109/SP54263.2024.00179. https://doi.org/10.1109/SP54263.2024.00179

[43] D. A. Alber et al., "Medical large language models are vulnerable to data-poisoning attacks," Nature Medicine, vol. 31, pp. 618–626, 2025. doi: 10.1038/s41591-024-03445-1. https://doi.org/10.1038/s41591-024-03445-1

[44] P. Villalobos et al., "Will we run out of data? Limits of LLM scaling based on human-generated data," arXiv:2211.04325v2, 2024. https://arxiv.org/html/2211.04325v2

[45] J. Betley, D. Tan, N. Warncke, A. Sztyber-Betley, X. Bao, M. Soto, N. Labenz, and O. Evans, "Emergent Misalignment: Narrow finetuning can produce broadly misaligned LLMs," *arXiv preprint* arXiv:2502.17424, ver. 6, May 12, 2025. [Online]. Available: https://arxiv.org/abs/2502.17424

[46] Palisade Research, "Misalignment Bounty Submissions Dataset," *Hugging Face*, 2025. [Online]. Available: https://huggingface.co/datasets/palisaderesearch/Misalignment-Bounty-Submissions

[47] U.S. Department of Defense, "Protection of Mission Critical Functions to Achieve Trusted Systems and Networks," DoD Instruction 5200.44, Feb. 16, 2024. [Online]. Available: https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/520044p.pdf

[48] J. Boyens, A. Smith, N. Bartol, K. Winkler, A. Holbrook, and M. Fallon, "Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations," NIST Special Publication 800-161 Revision 1 Update 1 (includes updates as of Nov. 1, 2024), National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, May 2022. doi: 10.6028/NIST.SP.800-161r1-upd1. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-161r1-upd1.pdf

[49] J. C. Smith, Effective Security by Obscurity, arXiv:2205.01547 [cs.CR], Apr. 30, 2022. Available: https://arxiv.org/abs/2205.01547