

American Journal of Data Science and Analytics

Volume: *7;* ***Issue:*** *10*
Pages: *01-35*

AI-Accelerated Software Supply Chain Compromise: An Emerging Threat to Cyber-Physical Systems and the Case for Zero Trust Architecture

Md Hasinur Rahman[1]; Md Atikur Rahman[2];

[1]. *PhD in information Technology, University of the Cumberland, USA*
M.A. in Professional Communication, University of San Francisco, USA
M.A. in English Language & Literature, Dhaka International University, Bangladesh
B.A. (Honors) in English Literature, Edward College, National University, Bangladesh
Email: rahmanhasinur10@gmail.com

[2]. *Operations Manager, RUVATI USA Inc.*
MS in Management with Data Science emphasis, Faulkner University, USA
MA in Professional Studies, University of North Alabama, USA.
Email: mrahman1@una.edu

Doi: 10.63125/8m7wek47
Received: *09 August 2026;* ***Revised:*** *11 August 2026;* ***Accepted:*** *8 October 2026;* ***Published:*** *10 October 2026*

Abstract

Cyber-physical systems that operate critical infrastructure now depend heavily on open-source software and third-party components, and the software supply chain therefore reaches directly into operational technology (OT). This paper examines AI-accelerated software supply chain compromise as an emerging threat to cyber-physical systems and uses the 2024 XZ Utils backdoor (CVE-2024-3094) as its central warning case. In that incident an adversary spent years posing as a trusted contributor to a widely used compression library and inserted an obfuscated backdoor, hidden in release archives rather than in the source repository, that could have enabled pre-authentication remote code execution through OpenSSH on affected Linux systems before it was discovered by a downstream engineer. Drawing on peer-reviewed supply-chain security research, NIST SP 800-207, SP 800-161r1, SP 800-82 Rev. 3, the NIST Cybersecurity Framework 2.0, CISA guidance and the ISA/IEC 62443 series, the paper characterises the cyber-physical environment, analyses the threat and its actors, constructs a forward-looking OT scenario, assesses technical, organisational and human vulnerabilities, and evaluates impacts and risks across six scenarios. It reaches three findings: the attack surface is moving upstream into maintainers, build pipelines and vendor packages that operators do not control; OT amplifies the risk because of long device lifetimes, constrained patching, safety-critical functions and implicit trust in signed updates; and perimeter and signature-based defences cannot stop malicious code that arrives inside a legitimate update. Evidence from studies of large language models in phishing, code generation and autonomous exploitation indicates that artificial intelligence lowers the cost of the patient trust-building, code manipulation and obfuscation that made XZ possible. The principal recommendation is to extend Zero Trust Architecture beyond users and networks to the software supply chain itself through software bills of materials, provenance attestation, strong cryptographic entropy for signing material, least privilege and segmentation, behavioural monitoring, software quarantine and known-good rollback, and a proposed attestation-based admission control at the OT boundary. A sequenced strategy, response and recovery design, and research agenda are presented.

Keywords

Software Supply Chain, Cyber-Physical Systems, Operational Technology, Zero Trust Architecture, XZ Utils, Provenance Attestation, SBOM, Artificial Intelligence.

I. INTRODUCTION

Critical infrastructure increasingly depends on cyber-physical systems (CPS) in which software directly governs physical processes such as generating and distributing power, treating water, moving trains, compressing and transporting gas, and running manufacturing lines. As operational technology (OT) has converged with information technology (IT), and as the Industrial Internet of Things (IIoT) and edge computing have expanded, the software inside industrial environments has started to resemble the software used in mainstream IT. Gateways, historians, engineering workstations and edge controllers now depend on general-purpose operating systems, open-source libraries and layered third-party dependencies (Makrakis et al., 2021; Sadeghi et al., 2015; Dhirani et al., 2021). This shift improves efficiency, interoperability and manageability, but it also creates a serious and under-examined risk: a trusted software component can be compromised before it ever reaches the organisation, and it can arrive inside an ordinary, validly signed update.

The 2024 XZ Utils backdoor is one of the clearest warnings of that risk. It was not a quick intrusion. Public analyses describe a patient, multi-year effort by a pseudonymous contributor to gain trust and eventually co-maintainer authority in a compression project that ships in most Linux distributions, followed by the insertion of an obfuscated payload that was present in release archives but not in the source repository and that hooked the OpenSSH daemon on affected builds (Tafani-Dereeper & Knowles, 2024; Lins et al., 2024; SentinelOne, 2024). The backdoor was caught before it reached most stable production systems, by a downstream engineer who noticed anomalous CPU consumption and latency rather than by any signature-based control (Tafani-Dereeper & Knowles, 2024; Cybersecurity and Infrastructure Security Agency, 2024). The same type of attack aimed at Linux-based gateways, controllers and edge devices in OT would be much harder to detect and far more serious, because such devices are re-verified rarely, patched slowly and connected to physical processes.

Supply chain attacks are also increasing. The European Union Agency for Cybersecurity analysed two dozen supply chain attacks reported between early 2020 and mid-2021, found that most of them targeted the supplier's code, attributed roughly half to advanced persistent threat groups, and projected a fourfold increase in such attacks over the preceding year, concluding that even a well-defended organisation can be compromised through a trusted supplier (European Union Agency for Cybersecurity, 2021). The SolarWinds Orion compromise of 2020, in which a vendor's build system was subverted so that signed updates carried malicious code to thousands of customers, showed the same mechanism at the level of a commercial product (Peisert et al., 2021). Studies of public package registries have catalogued hundreds of malicious packages and shown how small the attacker's cost can be relative to the reach that a single dependency confers (Ohm et al., 2020; Zimmermann et al., 2019; Duan et al., 2021).

Artificial intelligence (AI) increases this concern because it can help attackers scale the same kind of social engineering, code-review manipulation and package analysis that made the XZ case so dangerous. Controlled studies have shown that large language models can produce spear-phishing messages whose effectiveness approaches that of human experts at a fraction of the cost (Heiding et al., 2024; Hazell, 2023), that AI agents can exploit known vulnerabilities with limited human direction (Fang et al., 2024), and that code-generation models produce a substantial share of insecure code that a reviewer may accept (Pearce et al., 2022). None of these studies concerns the XZ incident itself, and there is no public evidence that AI was used in it; the argument of this paper is forward-looking. It is that the preconditions for XZ-style attacks are growing in OT at the same time as the effort required to execute them is falling.

This paper analyses an emerging cybersecurity challenge that is expected to affect CPS and critical infrastructure in the coming years. Its goals are to explain why software supply chain compromise creates new risks for OT environments, to show how AI can accelerate those attacks, and to evaluate how Zero Trust Architecture (ZTA) can reduce the risk. The analysis is supported by peer-reviewed scholarship and by recognised frameworks, including NIST SP 800-207 on zero trust (Rose et al., 2020), NIST SP 800-161r1 on cybersecurity supply chain risk management (Boyens et al., 2022), NIST SP 800-82 Rev. 3 on OT security (Stouffer et al., 2023), the NIST Cybersecurity Framework 2.0 (National Institute of Standards and Technology, 2024a), CISA guidance (Cybersecurity and Infrastructure Security Agency, 2023a; Cybersecurity and Infrastructure Security Agency, 2023b) and the ISA/IEC

62443 series (International Society of Automation, 2024). The XZ backdoor is treated not as an isolated incident but as a warning about how future attacks could reach industrial systems through trusted software.

The remainder of the paper reviews the related literature (Section II), characterises the cyber-physical environment (Section III), analyses the threat and the XZ case (Section IV), constructs a future OT scenario (Section V), assesses vulnerabilities, impacts and risks (Sections VI to VIII), extends zero trust to the software supply chain (Section IX), develops response, recovery and strategy (Sections X and XI), and closes with future research and final recommendations (Sections XII and XIII).

II. RELATED WORK

Software Supply Chain Security

The insight that trust in software cannot be established by inspecting source code alone is old. Thompson's demonstration that a compiler can be made to insert a backdoor into the programs it compiles, including future versions of itself, showed that the trustworthiness of an artifact depends on the entire chain of tools that produced it (Thompson, 1984). The modern software supply chain, with its package registries, continuous-integration services, container images and automated dependency resolution, has multiplied the links in that chain. Zimmermann et al. showed that a typical npm package transitively depends on dozens of other packages maintained by many unrelated people, so that a single maintainer account can affect a large share of the ecosystem (Zimmermann et al., 2019). Ohm et al. assembled a dataset of malicious packages observed in npm, PyPI and RubyGems and derived an attack tree covering typosquatting, account compromise and injection into legitimate packages (Ohm et al., 2020), and Duan et al. built a measurement pipeline that found hundreds of previously unknown malicious packages across interpreted-language registries (Duan et al., 2021).

Ladisa et al. consolidated this work into a systematic taxonomy of attacks on open-source software supply chains, organised around the attacker's goal of injecting malicious code and distinguishing the creation of a distinct malicious package, the subversion of a legitimate package through its maintainers, source, build or distribution, and the compromise of the consuming environment; the taxonomy was validated with practitioners and mapped to safeguards (Ladisa et al., 2023). Okafor et al. framed supply chain security around three properties, transparency, validity and separation, and used them to assess existing frameworks (Okafor et al., 2022). Enck and Williams reported the challenges practitioners face, including updating vulnerable dependencies, leveraging software bills of materials (SBOMs), choosing trusted dependencies and securing the build process (Enck & Williams, 2022; Hammi & Zeadally, 2023). The SolarWinds incident prompted a set of expert perspectives that emphasised build-system integrity and the limits of signature-based trust (Peisert et al., 2021), while the Log4Shell vulnerability demonstrated how rapidly a flaw in a ubiquitous library is scanned for and exploited once it becomes public (Hiesgen et al., 2022).

The defensive literature has produced several building blocks. Cappos et al. showed that package managers and mirrors could be attacked through replay, freeze and metadata manipulation (Cappos et al., 2008), which led to The Update Framework for survivable key compromise in update systems (Samuel et al., 2010) and to Uptane for automotive over-the-air updates, a setting whose long-lived, safety-relevant devices resemble OT (Kuppusamy et al., 2018). In-toto provides a layout-based mechanism for attesting every step of a software pipeline from source to delivery (Torres-Arias et al., 2019); Sigstore offers keyless signing with short-lived certificates and a transparency log (Newman et al., 2022); reproducible builds allow independent parties to verify that a binary corresponds to its source (Lamb & Zacchiroli, 2022); and the Supply-chain Levels for Software Artifacts (SLSA) specification defines graded build-provenance requirements (Open Source Security Foundation, 2023). SBOMs have been defined by the U.S. National Telecommunications and Information Administration (National Telecommunications and Information Administration, 2021) and examined empirically by Xia et al. and Stalnaker et al., both of whom found that adoption is uneven and that consumers struggle to act on SBOMs they receive (Xia et al., 2023; Stalnaker et al., 2024); Policy has moved in the same direction through Executive Order 14028 (Executive Office of the President, 2021), NIST SP 800-161r1 (Boyens et al., 2022), the Secure by Design principles (Cybersecurity and Infrastructure Security Agency et al., 2023) and the European Union's Cyber Resilience Act, which imposes vulnerability-handling and update obligations on manufacturers of products with digital elements (European Parliament and

Council, 2024).

Open-Source Sustainability and Maintainer Risk

The XZ incident turned on the fragility of volunteer maintenance. Eghbal documented the unseen labour behind digital infrastructure and the mismatch between the importance of open-source components and the resources available to maintain them (Eghbal, 2016). Avelino et al. estimated truck factors for popular GitHub projects and found that a majority depended on one or two developers (Avelino et al., 2016), Coelho and Valente found that loss of maintainer time and interest is a leading reason for project failure (Coelho & Valente, 2017), and Champion and Hill proposed underproduction, the mismatch between a package's importance and the labour it receives, as a measurable risk indicator (Champion & Hill, 2021). Zahan et al. identified weak-link signals in npm such as inactive or single maintainers and expired maintainer e-mail domains (Zahan et al., 2022), and later evaluated the OpenSSF Scorecard as a way to assess project security practices (Zahan et al., 2023). These studies supply the measurable preconditions for the maintainer-targeting attack described in Section IV.

Offensive Uses of Artificial Intelligence

Brundage et al. anticipated that AI would expand existing threats, introduce new ones and change their character by lowering the cost of labour-intensive attacks and enabling more targeted operations (Brundage et al., 2018), and Kaloudi and Li surveyed AI-based cyber threats across the attack life cycle (Kaloudi & Li, 2020). Mirsky et al. ranked offensive AI capabilities by their expected impact, with impersonation and social engineering among the most concerning (Mirsky et al., 2023). Empirical work has since supplied evidence. Hazell showed that large language models can generate personalised spear-phishing at negligible marginal cost (Hazell, 2023), and Heiding et al. found in a controlled study that AI-generated phishing achieved click-through rates comparable to messages written by human experts (Heiding et al., 2024). Fang et al. reported that an LLM agent could autonomously exploit a high proportion of a set of real one-day vulnerabilities when given their descriptions (Fang et al., 2024). On the code side, Pearce et al. found that roughly two-fifths of programs produced by a code-completion model in security-relevant scenarios contained vulnerabilities (Pearce et al., 2022), and Schuster et al. and Gu et al. showed that the models themselves can be poisoned to suggest insecure code or to carry hidden triggers (Schuster et al., 2021; Gu et al., 2019). Together these results support the claim that the human-intensive stages of a supply chain attack, persona management, communication, code review and obfuscation, are becoming cheaper.

Zero Trust and Cyber-Physical Systems

NIST SP 800-207 defines zero trust as a set of principles in which no implicit trust is granted on the basis of network location or asset ownership and every access decision is made dynamically by a policy engine that evaluates identity, device posture and context (Rose et al., 2020). The approach was popularised by Google's BeyondCorp (Ward & Beyer, 2014), surveyed by He et al. and Syed et al. (He et al., 2022; Syed et al., 2022), and organised into a maturity model by CISA (Cybersecurity and Infrastructure Security Agency, 2023b). Buck et al. reviewed the academic and practitioner literature and noted that empirical evidence of zero trust outcomes remains thin (Buck et al., 2021). Its application to industrial systems is more recent: Feng and Hu proposed a cyber-physical zero trust architecture for industrial CPS that incorporates physical-state awareness into access decisions (Feng & Hu, 2023), Ojo examined adoption in critical infrastructure protection (Ojo, 2025), and Qudus characterised CPS vulnerability as systemic and argued for resilient design (Qudus, 2025). Earlier CPS security surveys established the inverted priority of availability and integrity, the long life of installed equipment and the coupling of safety and security that make OT different from enterprise IT (Humayed et al., 2017; McLaughlin et al., 2016; Giraldo et al., 2018; Knowles et al., 2015), and NIST SP 800-82 Rev. 3 now addresses zero trust explicitly in an OT context (Stouffer et al., 2023).

Attestation, Firmware Integrity and Cryptographic Foundations

Verifying that a device runs the software it is supposed to run is the subject of remote attestation. Sailer et al. designed a measurement architecture based on the Trusted Platform Module that records every loaded component (Sailer et al., 2004; Trusted Computing Group, 2019), Coker et al. set out the principles of remote attestation (Coker et al., 2011), Abera et al. extended attestation to control flow for embedded devices (Abera et al., 2016), and Ambrosin et al. surveyed collective attestation for large device populations (Ambrosin et al., 2020). Large-scale analyses of embedded firmware have shown

how often vulnerable or outdated components are shipped (Costin et al., 2014), and Cui et al. demonstrated that firmware updates can be weaponised when their integrity is not verified (Cui et al., 2013). Signing and attestation depend on cryptographic keys, and Heninger et al. showed that poor entropy at key generation produced factorable or shared keys on a measurable fraction of internet-facing devices (Heninger et al., 2012); NIST's recommendations on deterministic random bit generators, entropy sources and key management set out the engineering response (Barker & Kelsey, 2015; Turan et al., 2018; Barker, 2020).

Research Gap

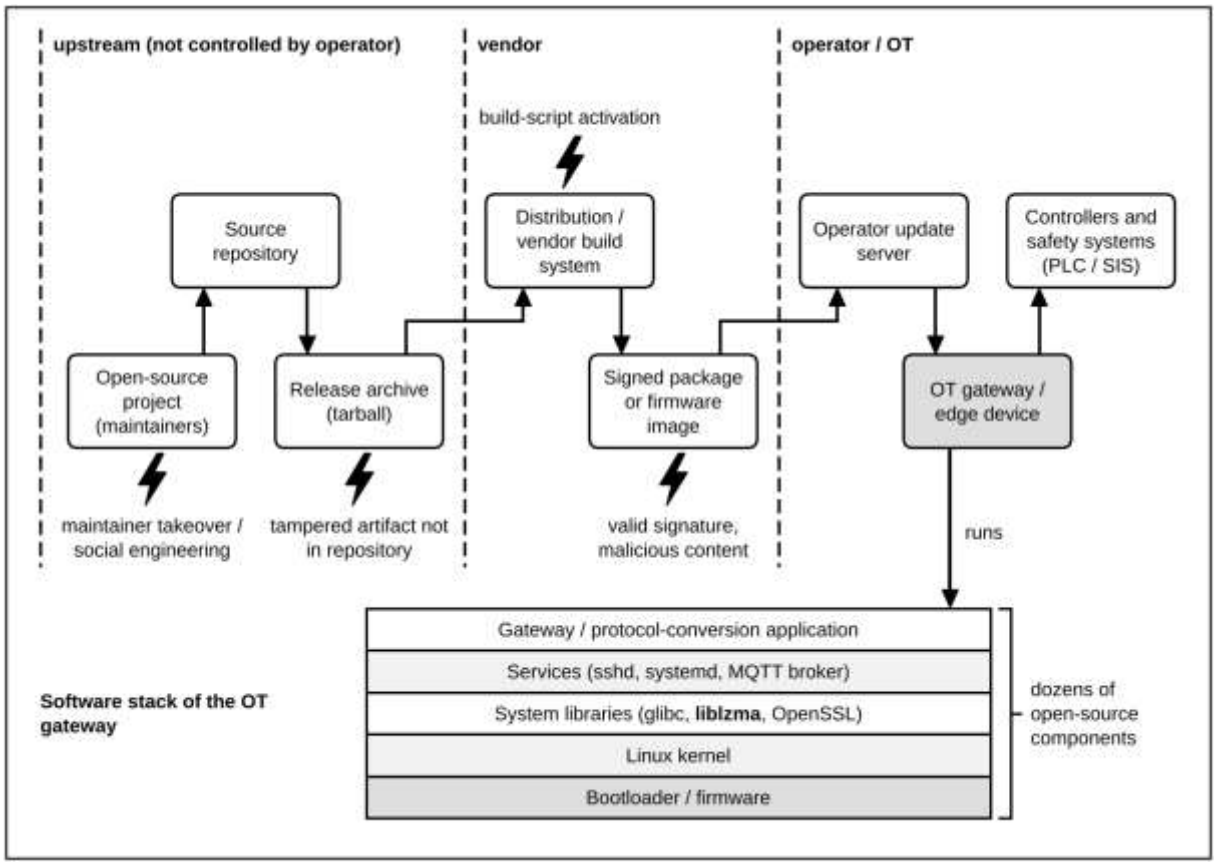
Each strand above is mature on its own, but their intersection is thin. The supply chain literature focuses on registries, build systems and enterprise software; the CPS literature focuses on network segmentation, protocol monitoring and physical-process anomalies; the offensive-AI literature focuses on phishing and exploitation; and the zero trust literature focuses on users and sessions. Few works ask what happens when a socially engineered, AI-assisted upstream compromise is delivered through a signed update to a long-lived industrial device, or how zero trust should be extended to the software that runs on such devices. This paper addresses that gap by connecting the strands in a single analysis of threat, vulnerability, impact, risk and architecture.

III. CYBER-PHYSICAL SYSTEM ENVIRONMENT

A cyber-physical system couples computation and communication with physical processes. In critical infrastructure, sensors measure physical conditions, programmable logic controllers (PLCs) and remote terminal units execute control logic, supervisory control and data acquisition (SCADA) systems aggregate signals, human–machine interfaces (HMIs) display information to operators, historians record the process, and communication networks connect these layers (Galloway & Hancke, 2013; Yadav & Paul, 2021). Historically, these environments were designed for availability, safety and deterministic real-time behaviour, and they were often assumed to be protected by physical isolation. Their security objectives are inverted relative to enterprise IT: the integrity of control actions and the availability of the process come before confidentiality, because the consequence of a successful attack is physical (Cárdenas et al., 2011; Humayed et al., 2017; Stouffer et al., 2023).

That assumption of isolation no longer holds. OT and IT are now connected through enterprise networks, remote support, cloud services and analytics platforms. Modern IIoT gateways, edge-computing nodes, protocol converters, SCADA servers and HMIs often run general-purpose operating systems and depend on the same open-source libraries, package managers and update mechanisms used in mainstream IT (Boyens et al., 2022; Sadeghi et al., 2015; Dhirani et al., 2021). A water-utility historian server, a substation gateway or a manufacturing edge controller may rely on dozens of open-source components maintained by people and organisations the operator has never directly evaluated. Large-scale firmware analyses have shown that embedded devices routinely ship with outdated versions of common libraries, and that the same component can be found across products from many vendors (Costin et al., 2014). The reference architectures that organise OT security, the Purdue model and the zones-and-conduits model of ISA/IEC 62443, were designed to govern communication between levels; they say little about where the software at each level came from (Stouffer et al., 2023; International Society of Automation, 2024).

Figure 1: Software Supply Chain Reach into a Cyber-Physical System and the Software Stack of an OT Gateway



This is the pivotal feature of the contemporary CPS environment for the purposes of this paper: the software supply chain now reaches directly into OT. Figure 1 traces the path from an open-source project through source repository, release archive, distribution or vendor build system, signed package or firmware image and operator update server to the OT gateway and the controllers behind it, and marks where in that path the XZ campaign intervened. The compression library at the centre of the XZ incident, liblzma, is the kind of low-level dependency that is embedded inside larger software stacks, including software running on industrial Linux devices, because compression is used by package managers, logging, archive handling and, through the systemd notification library on some distributions, by the OpenSSH daemon itself (Tafani-Dereeper & Knowles, 2024; Lins et al., 2024). When such a component is compromised upstream, the malicious code can enter an OT environment through an ordinary firmware or software update. It may arrive with a valid signature and may not trigger the perimeter alarms that OT security programmes rely on.

Table 1 summarises the software base, representative open-source dependencies, update channels and supply-chain exposure of the main CPS asset classes. Three observations follow. First, the asset classes with the broadest open-source exposure, gateways, edge nodes, SCADA servers and historians, are also those that sit at the boundary between the control network and everything else, so that a compromise there is well placed for lateral movement. Second, the asset classes with the narrowest exposure, controllers and safety systems, have the highest consequence and the least capacity for on-site verification, because their firmware is opaque and vendor-controlled (Cui et al., 2013; Costin et al., 2014). Third, every class depends on an update channel that the operator trusts but does not control. Studies of update systems have long warned that the update channel is itself an attack surface, through replay, freeze, mix-and-match and key-compromise attacks against repositories and mirrors (Cappos et al., 2008; Samuel et al., 2010).

Table 1: Cyber-Physical System Asset Classes, Software Bases, Open-Source Dependencies, Update Channels and Supply-Chain Exposure

CPS Asset Class	Typical Software Base	Representative Open-Source Dependencies	Update Channel	Supply-Chain Exposure
IIoT / edge gateways and protocol converters	Embedded Linux (Yocto, Debian derivatives), containerised services	Kernel, glibc, OpenSSL, liblzma / xz, systemd, OpenSSH, MQTT brokers, protocol libraries	Vendor firmware images; package feeds; OTA services	Very high: full general-purpose stack, often internet-adjacent, long service life
SCADA servers, HMIs and engineering workstations	Windows or Linux servers with vendor applications	Runtime libraries, OPC UA stacks, web frameworks, database engines, compression and TLS libraries	Vendor installers; OS package updates	High: large third-party dependency trees; privileged position in the control hierarchy
Historians and analytics nodes	Enterprise operating systems, time-series databases	Database engines, data-science libraries, web servers	OS and application package managers	High: bridge between OT and enterprise or cloud
PLCs, RTUs and DCS controllers	Vendor real-time operating system or embedded Linux	Real-time kernels, TCP/IP stacks, cryptographic and compression libraries inside firmware	Vendor firmware via engineering tools	Medium to high: opaque firmware; infrequent updates; verification rarely possible on site
Safety instrumented systems	Certified safety controller firmware	Limited, but vendor-supplied libraries are still embedded	Vendor-controlled firmware	Lower frequency but highest consequence; recertification needed after change
Cloud and remote-support tooling	Vendor SaaS, remote-access agents, VPN clients	Web and agent frameworks, SDKs, container base images	Continuous vendor deployment	High: fast-moving dependencies with direct reach into OT sessions

In today's CPS environments, security can no longer stop at the network boundary. Organisations must also consider the software suppliers, open-source projects, build systems and third-party components that support critical operations, which is precisely the scope that cybersecurity supply chain risk management (C-SCRM) is meant to cover (Boyens et al., 2022; Boyes, 2015; Ghadge et al., 2020).

IV. THREAT ANALYSIS

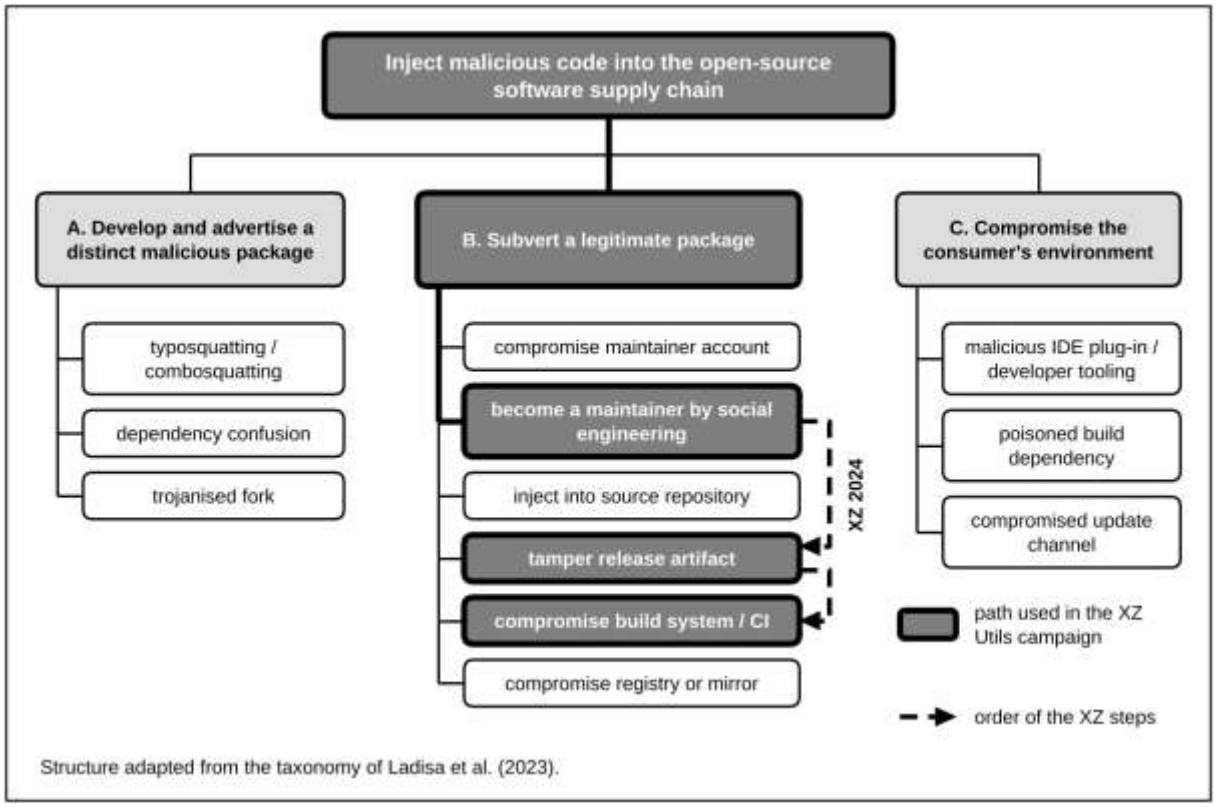
Software Supply Chain Compromise as a Threat Class

The primary emerging threat examined here is software supply chain compromise: the subversion of a component, dependency, build pipeline or update mechanism so that malicious functionality is delivered to victims through trusted channels. Ladisa et al. (2023) provide a peer-reviewed taxonomy of attacks on open-source software supply chains, distinguishing, among other vectors, the development of a distinct malicious package through typosquatting or dependency confusion (Vu et al., 2020; Ohm et al., 2020), the subversion of a legitimate package through compromise of maintainer accounts, injection into the source repository, tampering with release artifacts or compromise of the build system, and the compromise of the consumer's own environment. Table 2 arranges these vector families with the empirical evidence for each and records which the XZ campaign used. The XZ backdoor combined several of them: a long-term social-engineering campaign to obtain maintainer authority, followed by injection of an obfuscated payload hidden in release artifacts and activated during the build process (SentinelOne, 2024; Lins et al., 2024). Figure 2 shows the resulting taxonomy as a tree with the XZ path highlighted.

Table 2: Open-Source Supply Chain Attack Vectors, Evidence, and Their Use in the XZ Utils Campaign

Vector Family	Representative Techniques	Research and Incident Evidence	Used in XZ Utils?
Distinct malicious package	Typosquatting, combosquatting, dependency confusion, trojanised forks	Registry studies of npm, PyPI and RubyGems (Ohm et al., 2020; Vu et al., 2020; Duan et al., 2021)	No
Subvert a legitimate package: maintainer	Account takeover; social engineering to obtain maintainer rights; abandoned-project adoption	Weak-link analysis of npm maintainers (Zahan et al., 2022); XZ pressure campaign (Lins et al., 2024)	Yes: co-maintainer authority obtained through a multi-year persona
Subvert a legitimate package: source and artifacts	Commits of obfuscated code; malicious binary test data; release archive differs from repository	Taxonomy of Ladisa et al. (2023); XZ release tarball analysis (Tafani-Dereeper & Knowles, 2024)	Yes: payload hidden in test files and an archive-only build macro
Subvert build and distribution infrastructure	Tampered build scripts; compromised CI; package-manager and mirror attacks	Package-manager attacks (Cappos et al., 2008); SolarWinds build compromise (Peisert et al., 2021)	Yes: activation through modified build scripts under targeted conditions
Consumer-side compromise	Malicious developer tooling, IDE plug-ins, poisoned build dependencies, compromised update channels	Update-system attacks (Samuel et al., 2010; Kuppusamy et al., 2018)	No, but the intended downstream effect was a compromised update channel

Figure 2: Taxonomy of Open-Source Supply Chain Attack Vectors with the XZ Utils Path Highlighted



What distinguishes supply chain compromise from the attack classes that dominate the OT incident record is its position in the kill chain. The intrusion models that OT defenders use, the Lockheed Martin kill chain and its ICS-specific extension, describe an adversary who must gain initial access, establish a foothold, move laterally and develop the capability to affect the process (Hutchins et al., 2011; Assante & Lee, 2015). Stuxnet, Industroyer and Triton all followed that pattern, each requiring weeks or months

of intrusion inside the victim's network before an effect could be produced (Falliere et al., 2011; Cherepanov, 2017; Johnson et al., 2017). A supply chain compromise collapses the early stages: the adversary's code is delivered by the victim's own update process, with the victim's own privileges, to every site that installs it. In the MITRE ATT&CK for ICS knowledge base the equivalent initial-access technique, supply chain compromise, is listed alongside exploitation of public-facing applications and spear phishing, but it differs from them in requiring no interaction with the victim at the moment of entry (Alexander et al., 2020; Strom et al., 2018). The Stuxnet authors already exploited a related form of misplaced trust by signing their drivers with stolen vendor certificates, so that the operating system accepted them as legitimate (Falliere et al., 2011; Langner, 2011); the XZ technique goes one step further by producing a genuinely signed artifact whose contents are malicious.

Threat Actors

The likely threat actors span the spectrum that Ojo (2025) describes for critical infrastructure, from opportunistic criminals to sophisticated nation-state groups, but the emerging supply chain threat is dominated by patient and well-resourced actors. ENISA found that roughly half of the supply chain attacks it analysed were attributed to advanced persistent threat groups and characterised suppliers as a fast-emerging weakest link (European Union Agency for Cybersecurity, 2021). The XZ operation fits this profile. Analysts assess that its multi-year timeline, operational security and tradecraft point to a state-aligned actor, and follow-up research suggests that the same actor was preparing additional changes beyond the backdoor that was discovered (SentinelOne, 2024; Lins et al., 2024); attribution has not been publicly confirmed, and this paper treats the actor's identity as unknown. What makes this type of threat actor especially dangerous is patience. Rather than exploiting a vulnerability immediately, the attacker may spend years building trust within a project before introducing malicious code, which is a profile that OT incident history has not previously had to accommodate (Hemsley & Fisher, 2018; Miller et al., 2021). Financially motivated groups are not excluded: the ransomware operators who disrupted Colonial Pipeline's operations through its business network demonstrated the willingness to accept physical consequences (Beerman et al., 2023; Oz et al., 2022), and registry-level supply chain attacks for credential and cryptocurrency theft are already common (Ohm et al., 2020; Duan et al., 2021).

The XZ Utils Case

Because the XZ incident anchors the analysis, Table 3 sets out what public analyses report about it, with the sources on which each statement rests; the reconstruction is drawn from vendor and academic analyses rather than from primary investigation by this paper, and details may be refined as further findings appear. The affected component was XZ Utils and its liblzma library, versions 5.6.0 and 5.6.1, tracked as CVE-2024-3094 (Cybersecurity and Infrastructure Security Agency, 2024; Tafani-Dereeper & Knowles, 2024). A contributor persona appeared in 2021, made increasingly substantial contributions during 2022, benefited from a pressure campaign by apparently coordinated accounts urging the project's sole maintainer to hand over responsibility, and obtained co-maintainer authority in 2023 (Lins et al., 2024; SentinelOne, 2024). Over the following months the persona introduced binary test files and build-system changes that, when the release archives were built under particular conditions on x86-64 Linux systems with distribution packaging, extracted and linked an obfuscated payload into liblzma (Tafani-Dereeper & Knowles, 2024; Lins et al., 2024). Because several distributions patch OpenSSH to use the systemd notification library, which links liblzma, the payload could hook the daemon's authentication path and allow a holder of the attacker's private key to execute commands before authentication (Tafani-Dereeper & Knowles, 2024).

Table 3: The XZ Utils Backdoor (CVE-2024-3094): Elements Reported in Public Analyses

Element	What Public Analyses Report	Sources
Affected component	XZ Utils compression tools and the liblzma library, versions 5.6.0 and 5.6.1	CISA (2024); Tafani-Dereeper and Knowles (2024)
Vulnerability identifier	CVE-2024-3094, assigned the maximum CVSS base score	CISA (2024); Tafani-Dereeper and Knowles (2024)
Access path	Persona built trust over roughly two years, obtained co-maintainer authority after a pressure campaign on the sole maintainer	Lins et al. (2024); SentinelOne (2024)
Concealment	Malicious logic stored in binary test files; activating macro present only in release archives, not in the source repository	Tafani-Dereeper and Knowles (2024); Lins et al. (2024)
Activation conditions	Build-time checks for x86-64 Linux, specific toolchains and distribution packaging, limiting exposure to targeted builds	Tafani-Dereeper and Knowles (2024)
Effect	Hooked OpenSSH authentication through the systemd notification linkage used by some distributions; enabled remote command execution for a holder of the attacker's key	Tafani-Dereeper and Knowles (2024); Lins et al. (2024)
Discovery	A downstream engineer investigated unusual sshd CPU usage and latency; public disclosure on 29 March 2024	Tafani-Dereeper and Knowles (2024); CISA (2024)
Reach at discovery	Development and rolling-release distributions; stable enterprise releases largely unaffected	CISA (2024); Tafani-Dereeper and Knowles (2024)
Remediation	Downgrade to an uncompromised version and hunt for indicators; follow-up analyses suggest further changes were being prepared	CISA (2024); SentinelOne (2024)

Figure 3: XZ Utils Campaign Timeline as a Gantt Chart, 2021–2024 (Periods Approximate)

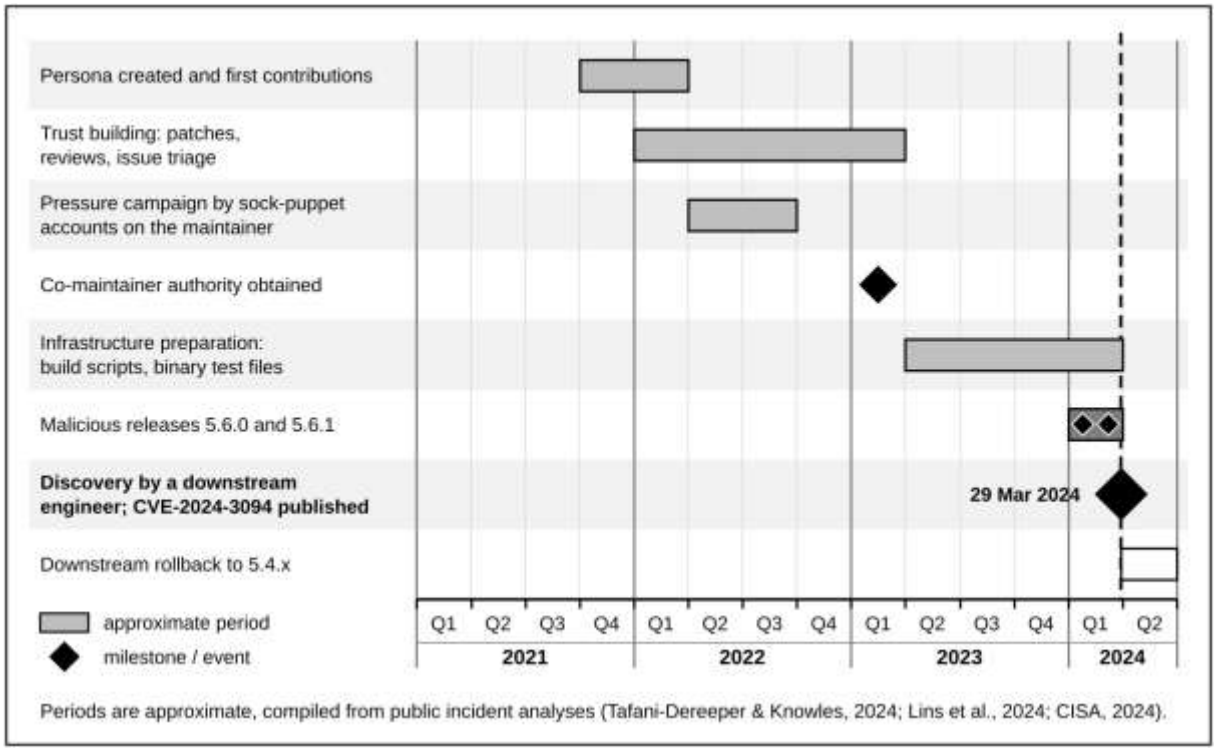


Figure 3 places these elements on a timeline. Two features of the incident matter most for CPS. The first is concealment: the activating macro was present only in the release archive that downstream builders used, not in the repository that reviewers read, and the payload was stored in files that look like corrupted test data (Lins et al., 2024; Tafani-Dereeper & Knowles, 2024). A reviewer of the source repository could not have found it, and a signature on the archive attested only that the project had released it. The second is discovery: the backdoor was found by a downstream engineer investigating

why the OpenSSH daemon on a development system consumed more CPU than expected and responded more slowly, and public disclosure followed on 29 March 2024 (Tafani-Dereeper & Knowles, 2024; Cybersecurity and Infrastructure Security Agency, 2024). The reach at discovery was limited to development and rolling-release distributions, and the recommended remediation was to downgrade to an uncompromised version and hunt for indicators (Cybersecurity and Infrastructure Security Agency, 2024). The incident was therefore contained by luck and by one engineer's attention, not by any control designed for the purpose.

Why the Threat Is Severe for Cyber-Physical Systems

These threats pose severe operational, organisational and societal risks for three reasons. First, they bypass the perimeter. A supply chain compromise can arrive inside a legitimate, signed update, defeating firewalls, network monitoring and signature-based detection that assume the threat comes from outside (Peisert et al., 2021; Boyens et al., 2022). Second, they exploit trusted components that are deeply and often invisibly embedded, so that a single compromised library can affect enormous numbers of downstream systems simultaneously; dependency studies show how few maintainers stand between a popular ecosystem and a large share of its packages (Zimmermann et al., 2019; Zahan et al., 2022). Third, in an OT context, the consequences are physical and potentially safety-critical, and the interconnected nature of infrastructure means that a compromise can cascade across sectors (Rinaldi et al., 2001; Ouyang, 2014). ENISA warns that this type of cascading effect can be catastrophic (European Union Agency for Cybersecurity, 2021). A supply chain foothold in a widely used industrial component is therefore not only a confidentiality problem. It can become a latent capability to disrupt physical processes at scale, held in reserve until the adversary chooses to use it, which is the profile of pre-positioning that security agencies have warned about in critical infrastructure (Cybersecurity and Infrastructure Security Agency, 2023a).

AI Acceleration of the Threat

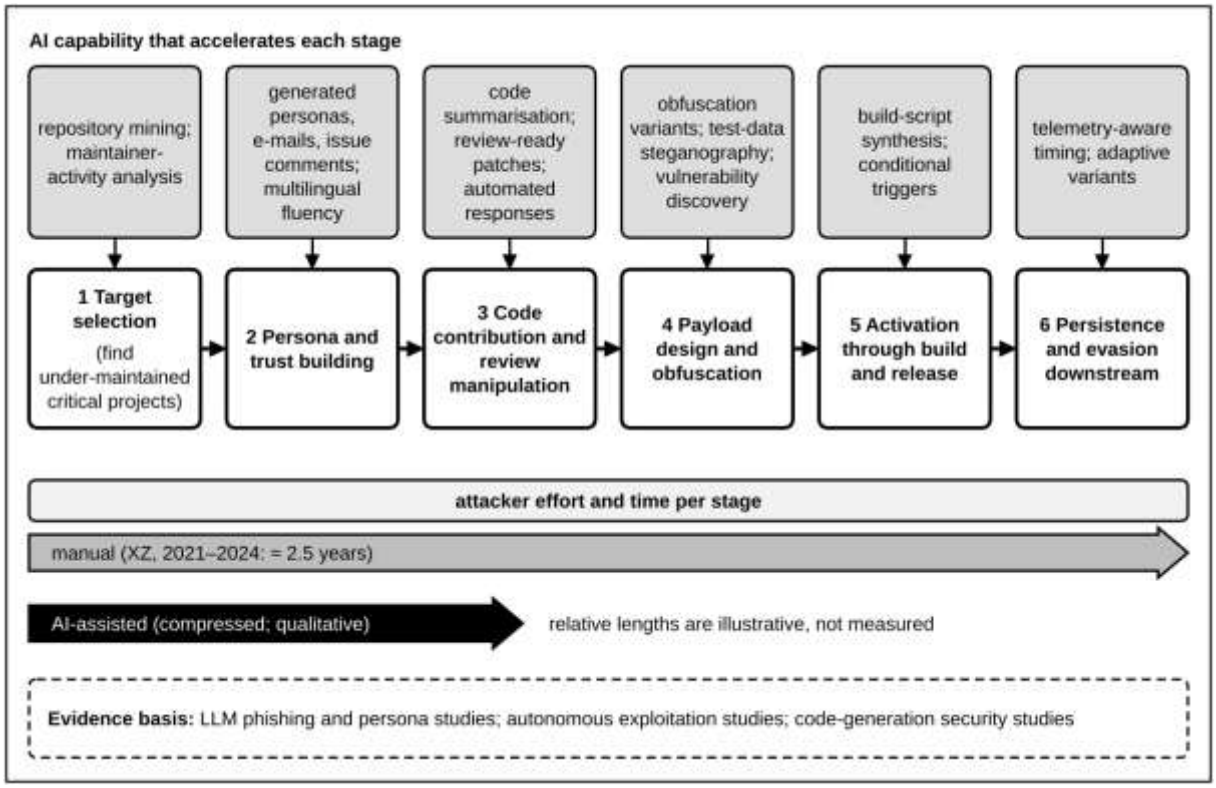
Artificial intelligence makes this threat more practical and more scalable. Attackers can use AI to summarise large codebases, identify weakly maintained open-source projects, generate plausible issue comments, write professional maintainer e-mails, translate technical discussions and create malicious code variants that are harder to recognise during manual review. AI can also help attackers test different obfuscation methods and hide malicious behaviour inside build scripts, test files or dependency chains. ReversingLabs reported that software supply chain attacks grew more sophisticated in 2024 and that attackers increasingly targeted AI, open-source, cryptocurrency and commercial software ecosystems (ReversingLabs, 2025). For CPS defenders, this means the next attack may not require the same level of manual effort that the XZ actor used. AI can reduce the time needed to find a target, build credibility and prepare malicious changes.

Table 4: AI Capabilities Mapped to Supply Chain Attack Stages, with Evidence Base and Confidence

Attack Stage	Manual Effort in the XZ Case	AI Capability That Reduces Effort	Evidence Base	Confidence
Target selection	Knowledge of which low-level libraries reach high-value systems; identification of an overburdened maintainer	Repository mining and maintainer-activity analysis at scale; underproduction metrics	Open-source risk measurement research (Champion & Hill, 2021; Zahan et al., 2023)	Moderate (tooling exists; adversary use inferred)
Persona and trust building	Hundreds of benign commits, reviews and messages over about two years; sock-puppet pressure accounts	Generated personas, fluent multilingual messages, consistent long-running conversation	LLM phishing and persuasion studies (Heiding et al., 2024; Hazell, 2023)	Moderate to high
Code contribution and review manipulation	Hand-written patches; management of reviewers' attention	Automated patch generation, summarisation of large code bases, plausible review responses	Code-generation security studies (Pearce et al., 2022)	Moderate
Payload design and obfuscation	Custom obfuscated payload hidden in test data and build macros	Rapid generation of obfuscation variants; automated vulnerability discovery	Autonomous exploitation research (Fang et al., 2024); offensive AI surveys (Mirsky et al., 2023)	Moderate (evidence mostly from controlled studies)
Activation through build and release	Hand-crafted build-script changes with conditional triggers	Synthesis of build logic and conditional activation	Code generation evidence; no public case of AI-authored build backdoor known to the authors	Low to moderate
Persistence and evasion	Dormant until triggered by a specific key	Telemetry-aware timing and adaptive variants	Survey evidence (Kaloudi & Li, 2020); largely speculative	Low

Table 4 and Figure 4 make the claim specific by mapping AI capabilities to the six stages of the attack and recording the evidence for each with an explicit confidence rating. The strongest evidence concerns the human-facing stages. Persona and trust building, which in the XZ case took about two years of benign contributions, reviews and messages, is exactly the kind of sustained, fluent, multilingual communication that large language models have been shown to produce at negligible marginal cost and with persuasive effect comparable to human experts (Hazell, 2023; Heiding et al., 2024; Mirsky et al., 2023). Target selection is supported by the same metrics that defenders have developed to measure open-source risk, truck factor, underproduction and weak-link signals, which an adversary can compute as easily as a defender (Avelino et al., 2016; Champion & Hill, 2021; Zahan et al., 2022). Evidence for the code-facing stages is moderate: code-generation models produce review-ready patches and a substantial share of insecure code (Pearce et al., 2022), and agentic systems have exploited known vulnerabilities autonomously in controlled settings (Fang et al., 2024), but no public case of an AI-authored build backdoor is known to the author. Evidence for AI-driven persistence and evasion is weakest and largely speculative (Kaloudi & Li, 2020). The effort comparison at the bottom of Figure 4 is qualitative; it records the direction of change that the evidence supports, not a measured compression factor.

Figure 4: AI Augmentation of the Supply Chain Attack Kill Chain (Effort Comparison Illustrative)



Two further considerations shape the threat. AI assistance is available to defenders too, and the same models that generate convincing personas can flag anomalous contributor behaviour or summarise release-archive differences; the asymmetry is that the attacker needs only one project to succeed, whereas the defender must cover all of them (Brundage et al., 2018; Sarker et al., 2021; Kaur et al., 2023). And AI systems are themselves software supply chain components: models and training pipelines can be poisoned to emit insecure code or carry hidden triggers (Schuster et al., 2021; Gu et al., 2019), so that an organisation adopting AI-assisted development inherits a new upstream dependency with its own provenance problem.

V. FUTURE SCENARIO: SUPPLY CHAIN COMPROMISE REACHING OT

The XZ backdoor was discovered within information technology before the compromised versions reached most stable and widely deployed systems. That is exactly why it belongs in a forward-looking analysis. It demonstrates a proven method whose preconditions are increasing in OT rather than decreasing. A realistic near-future scenario is straightforward. A Linux-based substation gateway, water-treatment historian or manufacturing edge controller depends on an open-source component maintained by a small number of volunteers. An adversary compromises that component upstream by combining patient trust-building with AI-assisted research, message generation, code analysis and obfuscation. The malicious code is then carried into the OT environment inside an ordinary, validly signed firmware or software update. Because the update is trusted and signed, it does not trigger perimeter alerts. Because OT devices are rarely re-verified after installation and remain in service for years, the backdoor can remain dormant until it is triggered. The result would be latent remote access to devices that support physical processes across many sites at once.

Figure 5: Future Scenario as a Sequence Diagram with Control Stop-Points S1–S5

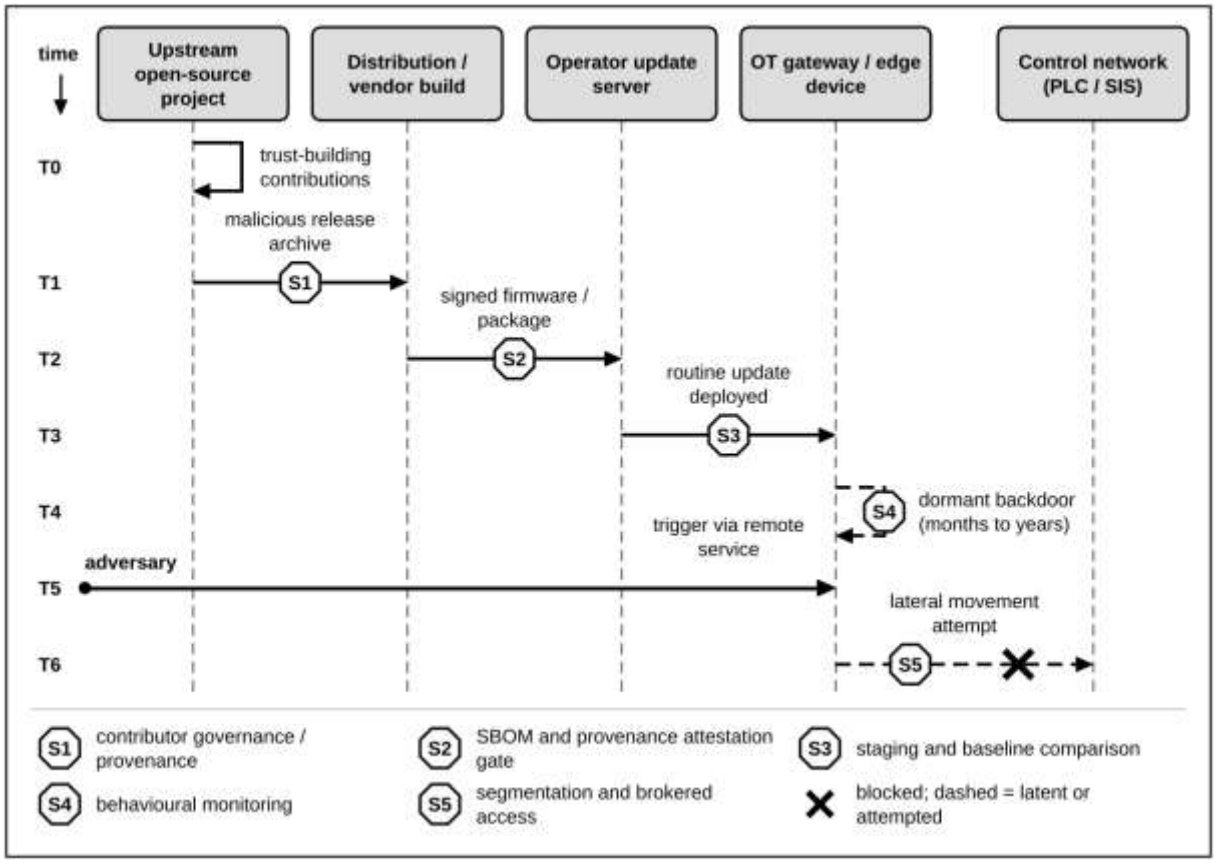


Figure 5 draws the scenario as a sequence diagram across five lifelines, from the upstream project to the control network, with the time steps T0 to T6 and the five stop-points at which the controls developed in Section IX would interrupt it. The sequence is deliberately generic. At T0 the adversary builds trust upstream; at T1 a malicious release archive reaches the distribution or vendor build; at T2 a signed package or firmware image reaches the operator's update server; at T3 a routine update is deployed to the gateway; at T4 the backdoor lies dormant, possibly for months or years; at T5 a trigger arrives through a remote service; and at T6 the adversary attempts lateral movement into the control network. Each step corresponds to a feature of the XZ campaign or of OT practice: the archive-only payload (T1), the valid signature (T2), the routine update (T3), the dormant trigger-gated behaviour (T4 and T5) and the OpenSSH-based remote access path (T5) are all documented in the XZ analyses (Tafani-Dereeper & Knowles, 2024; Lins et al., 2024), while the long dormancy and the lateral reach from a gateway to controllers reflect the known characteristics of OT estates (Makrakis et al., 2021; Stouffer et al., 2023).

Several variants deserve mention because they change which control matters most. In a vendor-build variant, the compromise occurs in the vendor's own build pipeline rather than in an open-source project, as in the SolarWinds case, so that provenance checks on upstream components would not help but attestation of the vendor's build would (Peisert et al., 2021; Open Source Security Foundation, 2023). In a firmware variant, the affected component is embedded in a controller's firmware, so that the operator has no package manager or SBOM to consult and must rely on the vendor's disclosure and on behavioural monitoring (Cui et al., 2013; Costin et al., 2014). In a remote-tooling variant, the compromised component is a vendor's remote-support agent or engineering tool, which already holds the credentials to change controller logic; here segmentation and brokered access are decisive (Stouffer et al., 2023; International Society of Automation, 2024). In each variant the common feature is the same: a trusted artifact that does not match the intent of the people who were supposed to have reviewed it. The XZ incident should therefore be read not as a contained IT event but as a warning for a future OT compromise that current trends make increasingly plausible.

VI. VULNERABILITY ASSESSMENT

The emerging threat succeeds by exploiting vulnerabilities that are technical, organisational and human, and that compound one another. Qudus (2025) characterises CPS vulnerability as systemic, arguing that weak identity and access governance, limited monitoring and legacy technology reinforce each other because control environments lack the compensating defences routine in IT. The software supply chain adds a further dimension: weaknesses in components and processes that the operator neither wrote nor controls. Table 5 registers ten vulnerabilities, classifies them and records how each enabled, or would enable, the compromise; Figure 6 shows how they reinforce one another as a causal-loop diagram.

Table 5: Vulnerability Register: Technical, Organisational and Human Weaknesses Exploited by Supply Chain Compromise

ID	Vulnerability	Class	How It Enabled or Would Enable Compromise	Primary Reference
V1	Release artifacts differ from the reviewed repository	Technical	Malicious macro existed only in the tarball; reviewers of the repository could not see it	Lins et al. (2024); Lamb & Zacchiroli (2022)
V2	Binary test data accepted without provenance	Technical	Payload stored in files that are not human-readable and rarely inspected	Tafani-Dereeper & Knowles (2024)
V3	No SBOM or component inventory in OT	Technical / organisational	Operators cannot tell which devices contain an affected library	Xia et al. (2023); Stalnaker et al. (2024)
V4	Implicit trust in signed updates	Technical / organisational	A valid signature covers whatever the build produced, including a backdoor	Boyens et al. (2022); Thompson (1984)
V5	Long device lifetimes and constrained patching	Technical	Compromised components remain in service long after disclosure	Makrakis et al. (2021); Costin et al. (2014)
V6	Single-maintainer fragility and burnout	Human	A pressure campaign transferred authority to an attacker persona	Eghbal (2016); Avelino et al. (2016)
V7	Susceptibility to AI-scaled social engineering	Human	Convincing messages and personas are cheap to generate and sustain	Heiding et al. (2024); Hazell (2023)
V8	No cybersecurity supply chain risk management programme	Organisational	No owner for supplier assurance, provenance requirements or response playbooks	Boyens et al. (2022); Enck & Williams (2022)
V9	Weak or unverified entropy for keys and identities	Technical	Predictable keys undermine signing, attestation and device identity	Heninger et al. (2012); Turan et al. (2018)
V10	Limited behavioural monitoring of OT devices	Technical / organisational	A dormant backdoor produces no signature; only behaviour reveals it	Stouffer et al. (2023); Sommer & Paxson (2010)

Technical Vulnerabilities

At the technical level, the XZ incident showed how difficult it can be to see inside the software supply chain. The malicious code was not visible in the ordinary source repository. It was hidden in release archives that downstream distributions actually built from, disguised as binary test data, and activated by tampered build scripts under targeted conditions (Tafani-Dereeper & Knowles, 2024; Lins et al., 2024). This opacity (V1 and V2 in Table 5) is made worse by the absence of SBOMs, by the difficulty of proving that a released binary matches reviewed source, and by the limited use of reproducible builds (Lamb & Zacchiroli, 2022; Xia et al., 2023). The deeper problem is the one Thompson identified four decades ago: a signature or a review of source establishes nothing about an artifact unless the entire chain that produced it is also trustworthy (Thompson, 1984). A valid signature on a tampered archive is not a failure of cryptography; it is a correct statement that the project released that archive. The vulnerability lies in what consumers infer from it (V4).

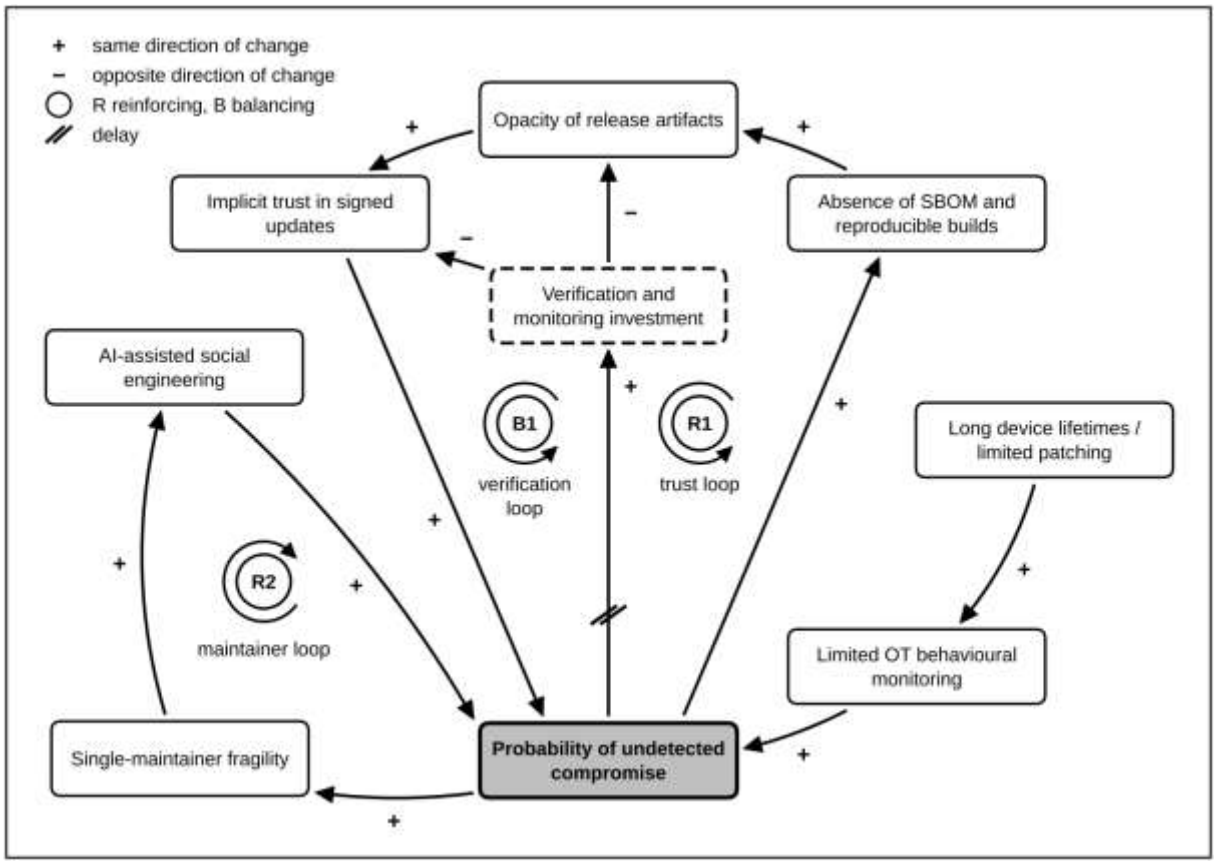
In OT, these weaknesses are magnified. Industrial devices often run outdated components, cannot be patched without downtime or vendor approval, and may remain in service for many years, so that one compromised component can persist long after it is known to be dangerous (V5) (Boyens et al., 2022; Makrakis et al., 2021; Knowles et al., 2015). Firmware analyses show that the same vulnerable library

versions recur across products and years (Costin et al., 2014). Most operators lack a complete inventory of the open-source components in their estates (V3), so that when a vulnerable or malicious component is disclosed they cannot say which devices contain it; the Log4Shell response demonstrated how costly that discovery process is even in IT (Hiesgen et al., 2022; Stalnaker et al., 2024). Behavioural monitoring of OT devices is limited (V10), and intrusion detection in OT has historically focused on network traffic and process variables rather than on the resource use and timing of software on gateways and servers (Hadžiosmanović et al., 2014; Urbina et al., 2016; Sommer & Paxson, 2010). Finally, the cryptographic material on which signing, attestation and device identity rest is often generated on devices with poor entropy or stored without hardware protection (V9), which has produced factorable and duplicated keys at internet scale (Heninger et al., 2012; Turan et al., 2018). These technical gaps make supply chain compromise especially difficult to contain because defenders may not know which devices depend on the vulnerable component, or whether the deployed software matches the reviewed source. For CPS environments, that lack of verification can turn a routine update into a long-term operational risk.

Organisational and Human Vulnerabilities

The organisational and human vulnerabilities are arguably more decisive. The XZ backdoor was made possible by the fragility of open-source maintenance. A critical component was maintained by a single, overburdened volunteer, and the attacker exploited that situation through a coordinated pressure campaign that used apparently fictitious accounts to push for a new maintainer until trust and commit authority were transferred (SentinelOne, 2024; Lins et al., 2024). This was a social-engineering attack on the human trust relationships that hold the supply chain together (V6 and V7), and it does not fit neatly into traditional perimeter defence. The precondition is widespread: a majority of popular projects depend on one or two developers (Avelino et al., 2016), maintainers leave when time and interest run out (Coelho & Valente, 2017), and the most important packages are not the best resourced (Eghbal, 2016; Champion & Hill, 2021). AI makes this type of manipulation more dangerous because it can help attackers imitate normal contributor behaviour, write convincing messages and maintain multiple online personas more easily (Hazell, 2023; Heiding et al., 2024).

Figure 6: Causal-Loop Diagram of Compounding Vulnerabilities with Reinforcing Loops R1 and R2 and Balancing Loop B1 (Author's Judgement)



On the operator's side, most critical-infrastructure organisations lack a formal C-SCRM programme (V8), extend implicit trust to vendor updates, and have no contractual means of obtaining SBOMs, provenance or build-practice disclosures from suppliers (Boyens et al., 2022; Enck & Williams, 2022; Boyes, 2015). Procurement, engineering and security functions are often separate, so that no one owns the question of whether a device's software can be trusted after it has been installed. Figure 6 summarises how these vulnerabilities reinforce one another: opacity of release artifacts and the absence of SBOMs sustain implicit trust in signed updates, which together with limited monitoring raise the probability of undetected compromise; single-maintainer fragility and AI-assisted social engineering raise the probability that an adversary obtains upstream authority in the first place; and the only balancing loop runs through investment in verification and monitoring, which is typically triggered by an incident rather than by foresight. The loop structure is the author's judgement, informed by the literature cited; it is a qualitative model rather than an estimated one. Exploitation therefore does not require breaching the operator directly. It may only require compromising something the operator already trusts.

VII. IMPACT ASSESSMENT

Because the XZ backdoor was caught before broad deployment, its realised impact was limited, but its potential impact and the impact of the broader threat class it represents are best understood across the dimensions that matter for CPS. Table 6 contrasts the impact realised in the XZ incident with the potential impact of an OT-reaching compromise on each dimension, and Figure 7 shows how an impact on a widely deployed component propagates across interdependent infrastructures.

Table 6: Impact Dimensions: Realised Impact of the XZ Incident Versus Potential Impact of an OT-Reaching Compromise

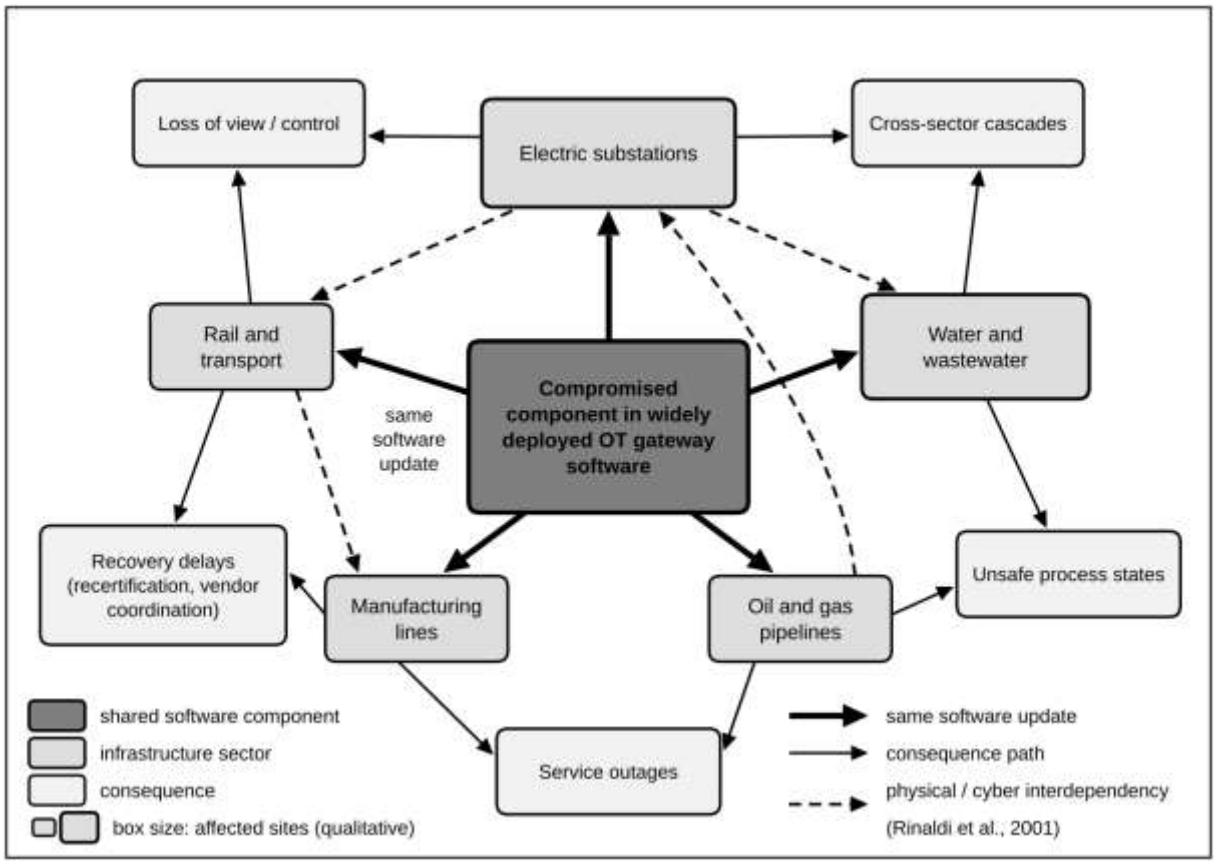
Impact Dimension	Realised in XZ (2024)	Potential in an OT-Reaching Compromise	Qualitative Severity
Operational continuity and service delivery	None in production; rolling-release systems downgraded	Coordinated loss of control or view across many sites served by the same gateway software	Very high
Human safety	None	Manipulation of dosing, pressure, protective relaying or shutdown functions through compromised gateways	Very high (site dependent)
Integrity of control decisions	Integrity of sshd authentication undermined on affected builds	Untrustworthy software beneath every control decision; silent setpoint or logic manipulation	Very high
Availability	Minor latency and CPU effects that led to discovery	Outages during quarantine, rollback and recertification	High
Confidentiality	Potential remote access to affected hosts	Exposure of process data, credentials and network layout for later attacks	Medium
Financial and organisational	Distribution rebuilds and incident response effort	Fleet-wide remediation, vendor coordination, recertification, regulatory reporting	High
Cross-sector and societal	None	Cascading effects through interdependent infrastructures; erosion of trust in software updates	High to very high

For operational continuity and critical-infrastructure services, a supply chain compromise that reaches OT could grant an adversary remote access to control-relevant devices across many sites at once, enabling coordinated disruption that is difficult to contain because it originates from trusted software rather than from an external intrusion. In the XZ case, the payload provided pre-authentication remote code execution through OpenSSH, precisely the kind of foothold that, on an industrial gateway, could be used to manipulate downstream controllers (Tafani-Dereeper & Knowles, 2024). The Ukrainian grid attacks showed what coordinated loss of control across several sites looks like in practice, and they required the adversary to compromise each utility individually (Cherepanov, 2017; Slowik, 2019); a

shared compromised component would remove that requirement.

For human safety, the concern is that compromised software components can eventually influence physical processes such as chemical dosing, pressure control, protective relaying or automated shutdown functions. The Triton incident demonstrated that an adversary with a foothold on an engineering workstation could reach a safety instrumented system and modify its logic (Johnson et al., 2017; Kriaa et al., 2015). For confidentiality, integrity and availability, supply chain attacks are especially dangerous because they attack integrity first. They weaken the trustworthiness of the software itself and, as a result, the trustworthiness of the control decisions that software supports. This is different from many traditional IT incidents, where confidentiality often receives the most attention. In CPS environments, integrity and availability of control are usually the highest priorities (Feng & Hu, 2023; Cárdenas et al., 2011).

Figure 7: Impact Cascade Network Across Interdependent Infrastructures Sharing a Compromised OT Gateway Component (Node Size Qualitative)



The financial and organisational consequences would be substantial and diffuse. Remediating a compromised, deeply embedded component across a fleet of long-lived OT devices is slow and expensive, frequently requiring vendor coordination and recertification, and safety-certified systems cannot simply be patched without re-validation (Kriaa et al., 2015; Stouffer et al., 2023). Organisational resilience, meaning the capacity to detect, respond to and recover from an incident, would be tested precisely where OT is often weakest: visibility into what software is running and whether it can be trusted (Linkov et al., 2013; Ganin et al., 2016). Finally, because critical infrastructure sectors are interconnected, ENISA warns that supply chain compromises can produce cascading effects (European Union Agency for Cybersecurity, 2021). Figure 7 draws on the interdependency framework of Rinaldi et al. (2001): a compromised gateway component shared by electric substations, water utilities, pipelines, manufacturing and transport would produce loss of view or control in each sector, and the physical and cyber interdependencies between sectors, electricity for water pumping and rail traction, gas for generation, transport for manufacturing inputs, would carry the disruption further (Ouyang, 2014; Ten et al., 2010). The societal impact of a successful attack on a widely used industrial component

could therefore extend well beyond any single operator. This makes a structured risk assessment useful because it connects each major attack condition to its likely impact and to the controls most likely to reduce that risk.

VIII. RISK ASSESSMENT SUMMARY

The risk assessment follows the qualitative approach of NIST SP 800-30, in which risk is characterised by the likelihood that a threat event occurs and the impact if it does, with both rated on ordinal scales and combined into an overall rating (Joint Task Force Transformation Initiative, 2012). The method is widely used because it is transparent and because the data needed for quantitative estimation of rare, adversarial events in OT do not exist (Kure et al., 2018; Cherdantseva et al., 2016). Its limitations are well known: ordinal ratings can be inconsistent across assessors, the combination of likelihood and impact into a single level loses information, and a matrix can rank risks incorrectly when the underlying quantities are correlated (Cox, 2008; Aven, 2016). The ratings in Table 7 should therefore be read as a structured statement of the author's judgement, informed by the evidence presented in Sections IV to VII, rather than as measured probabilities, and organisations applying them should calibrate the scales to their own context and, where data allow, replace them with quantitative estimates (Kaplan & Garrick, 1981; Hubbard & Seiersen, 2016).

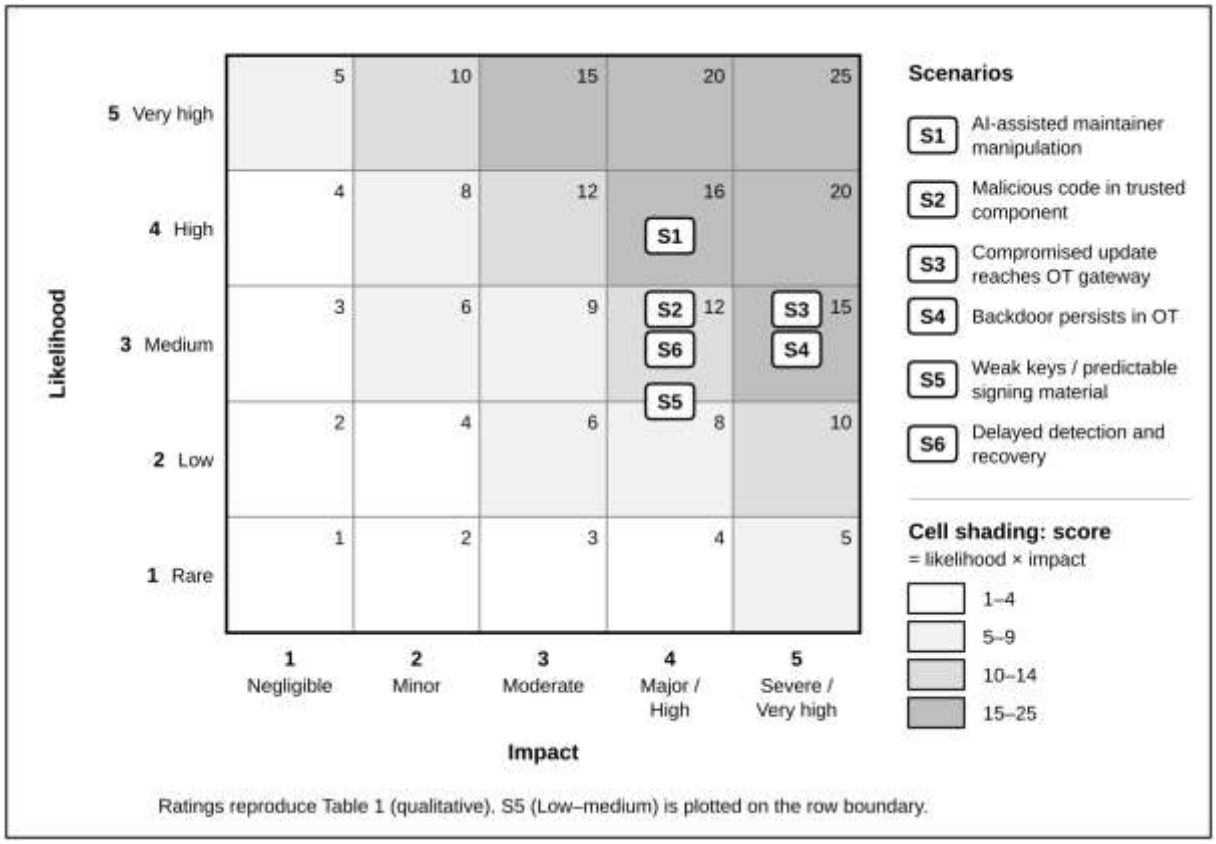
Table 7: Risk Assessment for AI-Accelerated Software Supply Chain Compromise in CPS

Risk Scenario	Likelihood	Impact	Overall Risk	Recommended Control
AI-assisted maintainer manipulation of an open-source project	High	High	High	Contributor governance, supplier risk review, and provenance checks
Malicious code inserted into a trusted software component	Medium	High	High	Reproducible builds, signed provenance, secure code review, and SBOM validation
Compromised update reaches an OT gateway or edge device	Medium	Very high	High	Update staging, attestation, software quarantine, and trusted rollback baselines
Backdoor persists inside the OT environment	Medium	Very high	High	Continuous monitoring, behavioral baselines, segmentation, and anomaly detection
Weak cryptographic keys or predictable signing material	Low to medium	High	Medium to high	Strong entropy sources, hardware random number generators, key rotation, and secure key storage
Delayed detection and recovery after component compromise	Medium	High	High	Known-good rollback, incident drills, dependency inventory, and software quarantine procedures

The risk levels are qualitative and reflect the likelihood and impact of AI-accelerated software supply chain compromise in CPS and OT environments.

Table 7 presents six risk scenarios that together trace the attack path from upstream manipulation to delayed recovery, and Figure 8 places them on a five-by-five likelihood–impact heat map by mapping the rating words to ordinal positions. AI-assisted maintainer manipulation (S1) is rated high on both dimensions: the preconditions are measurable and widespread, the XZ campaign proved the technique, and the evidence that AI lowers the cost of the social-engineering stage is the strongest in Table 4. The insertion of malicious code into a trusted component (S2), its delivery to an OT gateway (S3) and its persistence there (S4) are each rated medium in likelihood because they require the earlier stage to have succeeded and because distribution-level scrutiny does catch some attempts, but S3 and S4 are rated very high in impact because they place the adversary inside the control environment with latent access across sites. Weak cryptographic keys or predictable signing material (S5) is rated low to medium in likelihood, because the weakness is well understood and hardware random number generators are widely available, yet high in impact because it undermines every verification control that the other scenarios rely on. Delayed detection and recovery (S6) is rated medium and high because most operators lack the inventory and rollback capability that a rapid response requires.

Figure 8: Likelihood–Impact Heat Map of the Six Risk Scenarios in Table 7 (Ratings Qualitative)



The recommended controls in the final column are developed in Sections IX to XI. No single control is expected to stop the attack; rather, each scenario is paired with the control that most directly reduces its likelihood or impact, so that the controls together form the sequence of stop-points along the attack path shown in Figure 5, from contributor governance and provenance checks at the source (S1 and S2), through build attestation, SBOM validation, update staging and quarantine (S2 and S3), to monitoring, segmentation, key management and rehearsed rollback inside the OT environment (S4 to S6).

IX. ZERO TRUST ANALYSIS

Zero Trust Architecture rests on the premise that no user, device, software component or update should be trusted simply because it comes from a familiar source or sits inside a trusted network. Every access request and every deployment decision should be verified continuously (Rose et al., 2020). NIST SP 800-207 explains that organisations should treat all data sources and computing services as resources requiring protection, secure every communication regardless of location, grant access on a least-privilege and per-session basis, make access decisions dynamically from identity, posture and context, monitor the integrity and security posture of all assets, and collect as much information as possible about the current state of assets, networks and communications to improve policy (Rose et al., 2020). The architecture realises these tenets through a policy decision point, comprising a policy engine and a policy administrator, and policy enforcement points that mediate every access to a resource (Rose et al., 2020; He et al., 2022; Syed et al., 2022). Ojo (2025) and Feng and Hu (2023) argue that these principles map well onto CPS because excessive trust is a recurring weakness in industrial environments, and NIST SP 800-82 Rev. 3 now discusses zero trust for OT directly while cautioning that legacy devices cannot participate in dynamic authentication without intermediaries (Stouffer et al., 2023). For this paper, the most important adaptation is to extend zero trust from users and networks to the software supply chain itself. Table 8 maps each tenet of SP 800-207 from its conventional reading to that extension, and Figure 9 shows the resulting architecture.

Table 8: NIST SP 800-207 Zero Trust Tenets Extended from Users and Sessions to the Software Supply Chain

NIST SP 800-207 Tenet	Conventional Reading (users and sessions)	Extension to the Software Supply Chain	Supporting Guidance
All data sources and computing services are resources	Inventory and classify servers, applications and data	Treat every component, library, firmware image and build service as a resource with an owner and a trust state	Boyens et al. (2022); NTIA (2021)
All communication is secured regardless of location	Encrypt and authenticate sessions inside the perimeter	Authenticate update channels end to end; sign and verify artifacts, not just transport	Samuel et al. (2010); Kuppusamy et al. (2018)
Access to resources is granted per session	Re-authenticate and re-authorise each session	Admit each software deployment individually after provenance and policy checks	Open Source Security Foundation (2023)
Access is determined by dynamic policy	Use identity, device posture and behaviour in decisions	Include SBOM status, provenance level, vulnerability intelligence and build-platform assurance in admission policy	Rose et al. (2020); CISA (2023b)
Integrity and security posture of all assets are monitored	Continuous device posture assessment	Measure firmware and software integrity remotely; compare against attested baselines	Coker et al. (2011); Sailer et al. (2004)
Authentication and authorisation are dynamic and strictly enforced	Multi-factor authentication; least privilege	Machine identity for devices and build services; separation of signing authority; key rotation	Barker (2020); Trusted Computing Group (2019)
Collect information on the current state of assets, network and communications	Telemetry for security analytics	Behavioural baselines after each update; telemetry retention for forensic reconstruction	Stouffer et al. (2023)

Figure 9: Zero Trust Architecture Extended to the Software Supply Chain: Policy Decision Point, Session Enforcement, and a Software Admission Enforcement Point (after NIST SP 800-207)

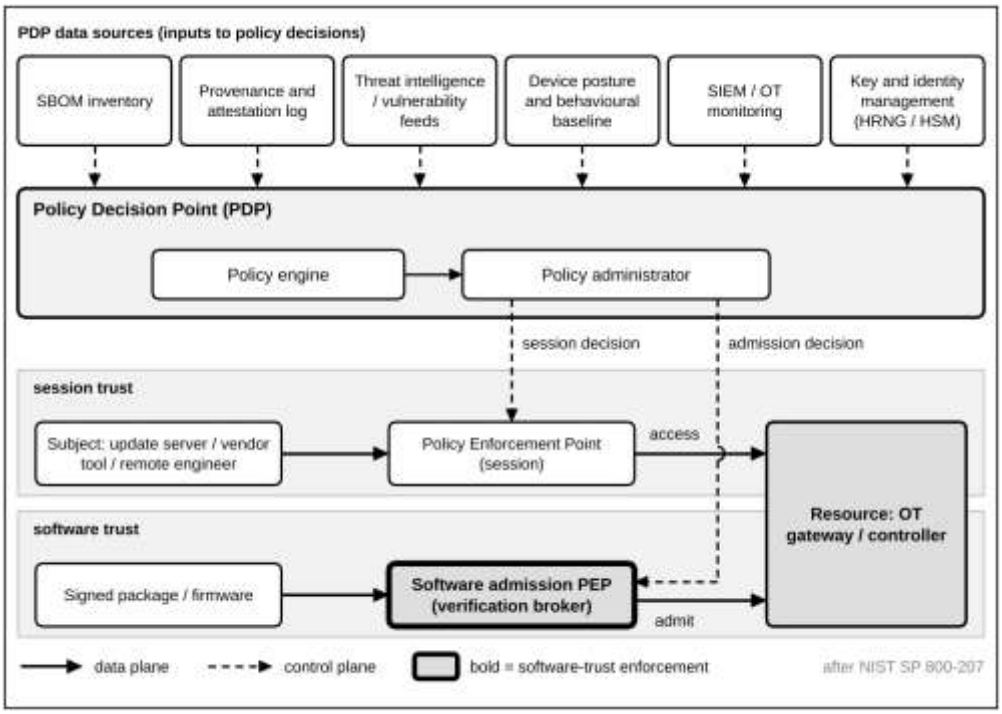


Figure 9 preserves the SP 800-207 structure, a policy decision point above a data plane in which a policy enforcement point mediates a subject's access to a resource, and adds a second enforcement point in bold: a software admission point, or verification broker, placed on the path between a signed package or firmware image and the OT device that would run it. The two paths are labelled session trust and software trust. The data sources that feed the policy engine are extended accordingly, from identity and device posture to SBOM inventory, provenance and attestation logs, threat and vulnerability intelligence, behavioural baselines, OT monitoring and key management. The rest of this section develops the five adaptations that give the software trust path its content.

Component Identity, Provenance, and Device Trust

The foundational adaptation is to treat software components as entities whose identity and integrity must be verified rather than assumed. In practice, this means requiring and validating software bills of materials so that operators know which components, including libraries such as liblzma, are running on control-relevant devices (National Telecommunications and Information Administration, 2021; Boyens et al., 2022). The minimum elements of an SBOM, supplier, component name, version, unique identifiers, dependency relationships, author and timestamp, are sufficient to answer the first question in any supply chain incident, namely where the affected component is deployed (National Telecommunications and Information Administration, 2021). Empirical studies show that generation and consumption tooling is immature and that consumers rarely act on SBOMs they receive (Xia et al., 2023; Stalnaker et al., 2024), which is why this paper treats the SBOM as an input to automated policy rather than as a document to be read. It also means verifying cryptographic provenance and using reproducible builds where possible so that a released binary can be tied to reviewed source (Lamb & Zacchiroli, 2022). Under such a model, an opaque or unverifiable component would fail a trust check before deployment, directly countering the concealment that made the XZ backdoor possible (Boyens et al., 2022). Device trust completes the picture: a device should be able to prove, through hardware-rooted measurement, what software it is running, so that the inventory reflects reality rather than records (Sailer et al., 2004; Coker et al., 2011; Ambrosin et al., 2020).

Provenance Attestation: Verifying the Build Itself

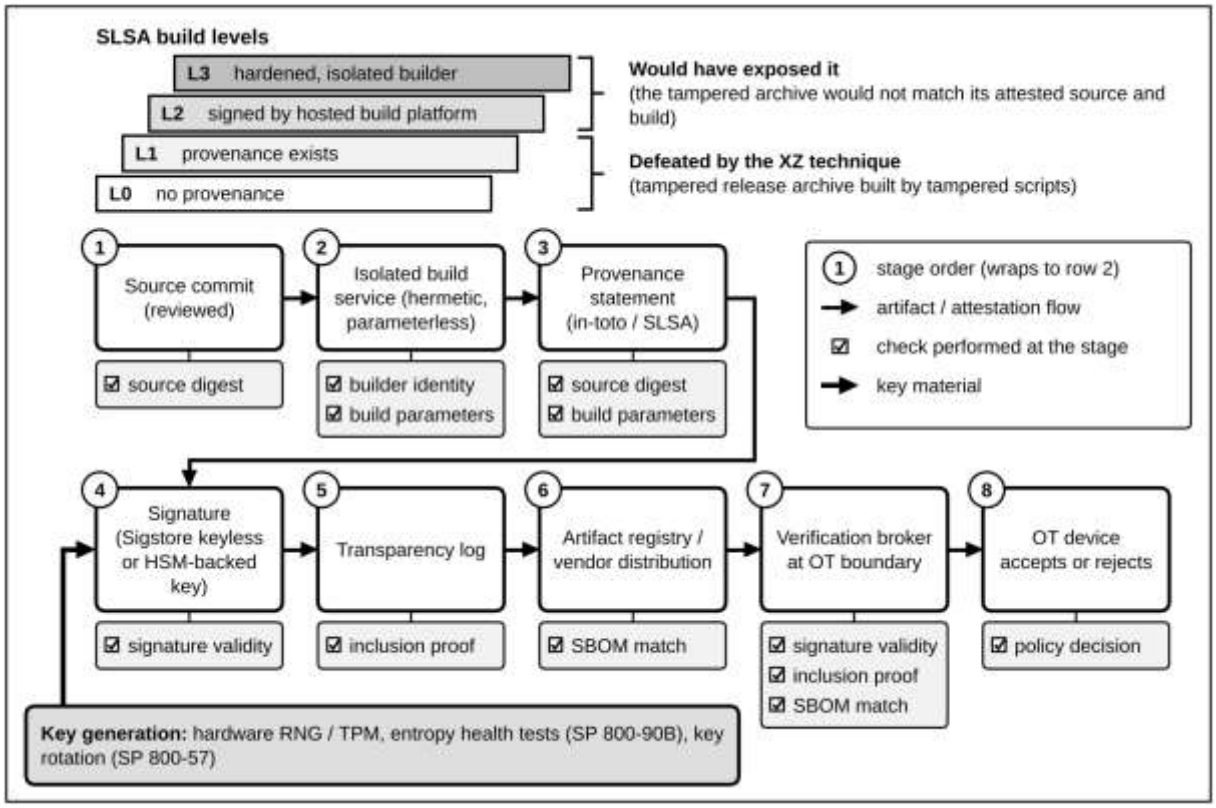
Verifying component identity matters only if the verification cannot be forged by the same adversary who compromised the component. Provenance attestation helps solve that problem. Frameworks such as SLSA and signing tools such as Sigstore can create signed records that show how an artifact was built, which source commit was used, which build system produced it and which inputs were included (Newman et al., 2022; Open Source Security Foundation, 2023; Torres-Arias et al., 2019).

Table 9: SLSA Build Levels and Complementary Measures, with Their Relevance to the XZ Utils Technique

SLSA Build Level	Requirement (summary)	Assurance Provided	Relevance to the XZ Technique
L0	No provenance	None; consumers trust the artifact by its source and signature alone	Equivalent to the status quo for most OT firmware and packages
L1	Provenance exists, generated by the build process, describing source and build steps	Consumers can see what the producer claims to have built; no protection against a tampered build	Tampered build scripts could have emitted misleading provenance
L2	Provenance generated and signed by a hosted build platform	Forgery requires compromising the platform, not just the project	Provenance would show the archive-only macro as an unexpected input
L3	Hardened, isolated builder; provenance unforgeable by the project's own build steps	Strong link between reviewed source and released artifact	Source-to-artifact comparison would expose a release that does not match the repository
Complementary: reproducible builds	Independent rebuild yields bit-for-bit identical output	Detects divergence between source and binary without trusting any single builder	Reproduction from the repository would have failed, revealing the tampering
Complementary: transparency log	Signatures and provenance recorded in an append-only public log	Detects unauthorised or unexpected signing events	Supports after-the-fact audit of what was signed and when

Higher assurance levels require provenance to be generated by an isolated build service rather than by the build scripts themselves. This directly addresses the XZ technique because the malicious payload was assembled through tampered build scripts in the release archive. Table 9 sets out the SLSA build levels and what each would have meant for that technique, and Figure 10 draws the resulting trust chain from source commit to device acceptance.

Figure 10: Provenance Attestation Chain from Source to Device, SLSA Build Levels, and Key Generation Inputs to the Signature Stage



At level L0 there is no provenance and consumers trust the artifact by its origin and signature alone, which is the status quo for most OT firmware and packages. At L1 provenance exists but is generated by the project's own build process, so a tampered build could emit misleading provenance. At L2 the provenance is generated and signed by a hosted build platform, so forging it requires compromising the platform rather than the project, and the archive-only macro would appear as an input that the repository does not contain. At L3 the builder is hardened and isolated and the provenance cannot be forged by the project's own build steps, so a comparison of reviewed source to released artifact would expose a release that does not match the repository (Open Source Security Foundation, 2023). Reproducible builds offer an independent check that does not require trusting any single builder (Lamb & Zacchiroli, 2022), and a transparency log makes every signing event auditable after the fact (Newman et al., 2022). In-toto layouts generalise this to any pipeline by declaring which functionaries may perform which steps and verifying the chain of link metadata at delivery (Torres-Arias et al., 2019). For OT, the practical consequence is that vendors should be asked at what SLSA level they build, whether their builds are reproducible and whether provenance accompanies each firmware image, and that the verification broker in Figure 9 should check that provenance before admission.

Cryptographic Entropy for Signing and Attestation

Strong software signing and attestation also depend on the quality of the cryptographic keys behind them. If signing keys, authentication tokens or attestation secrets are generated from weak or predictable randomness, the verification system can become easier to attack even when the architecture looks sound. Heninger et al. showed that devices generating keys at first boot with insufficient entropy produced RSA keys that shared factors with other devices' keys, and hence could be factored, on a

measurable fraction of internet-facing hosts (Heninger et al., 2012). Cloudflare's lava lamp entropy system is a widely cited illustration of the principle. Cloudflare explains that it uses images of a wall of lava lamps as a physical source of randomness to help seed cryptographic processes, because the changing shapes and movement of the lamps are difficult to predict (Cloudflare, n.d.). This idea does not stop a supply chain attack by itself, but it strengthens the cryptographic foundation that supply chain defences depend on.

In a CPS and OT context, this principle can be applied without literally copying Cloudflare's lava lamp wall. The practical recommendation is to use high-quality hardware random number generators and diverse entropy sources when generating signing keys, device identities, attestation keys, session tokens and update-validation credentials. NIST SP 800-90B specifies how an entropy source should be designed and tested, including continuous health tests that detect a failed or degraded source (Turan et al., 2018); SP 800-90A specifies the deterministic random bit generators that consume that entropy (Barker & Kelsey, 2015); and SP 800-57 sets out key-management practice, including key lifetimes, rotation and the separation of signing authority (Barker, 2020). Hardware roots of trust such as the Trusted Platform Module provide protected key generation and storage on the device itself (Trusted Computing Group, 2019). Entropy health checks, key rotation, secure key storage and separation of signing authority should therefore be part of the process. This supporting control fits naturally with zero trust because the model depends on trustworthy proof. Software provenance, signed builds, machine identity and device attestation are only as reliable as the cryptographic material that protects them. Looking further ahead, signing and attestation schemes will need to migrate to post-quantum algorithms over the lifetime of devices being deployed today, which argues for cryptographic agility in the verification broker (Bernstein & Lange, 2017; National Institute of Standards and Technology, 2024b).

Least Privilege, Segmentation, and Policy Enforcement

Zero trust also limits the blast radius of any component that is compromised despite these controls. Least-privilege scoping ensures that a service or device can perform only its defined function, so that a subverted library on a gateway cannot freely reach control processes. Micro-segmentation, aligned with the zones-and-conduits model of ISA/IEC 62443, isolates OT assets behind policy enforcement points so that a foothold in one component does not confer authority over the wider control network (Feng & Hu, 2023; International Society of Automation, 2024; International Electrotechnical Commission, 2023). Because the XZ backdoor sought remote access through OpenSSH, an environment that tightly restricted and brokered remote administrative access, through a jump host in the industrial demilitarised zone with per-session authorisation and approved devices, would have constrained an attacker's ability to exploit it even after the compromised component was installed (Stouffer et al., 2023; Cybersecurity and Infrastructure Security Agency, 2023a). The same enforcement points should govern the update servers and vendor tools themselves: an update server that can push software to every gateway is among the most privileged assets in the estate and should be treated as such, with its own machine identity, restricted conduits and logged sessions (Kuppusamy et al., 2018; Samuel et al., 2010).

Continuous Monitoring

Perhaps the most instructive lesson of the XZ case for zero trust is the value of continuous monitoring. The backdoor was discovered not by signature detection but because an engineer noticed anomalous behaviour, including unusual CPU consumption and latency introduced by the malicious code (Tafani-Dereeper & Knowles, 2024). A zero trust posture institutionalises this kind of behavioural vigilance by monitoring devices and components for deviations from expected activity rather than relying only on known-bad signatures. For a threat that arrives inside trusted, signed software, behavioural monitoring is often the remaining line of defence. The OT anomaly-detection literature has shown that process-aware and traffic-aware models can detect manipulation that signatures cannot (Hadžiosmanović et al., 2014; Urbina et al., 2016; Goh et al., 2017; Kravchik & Shabtai, 2018), while also warning that machine-learning detectors must be evaluated against realistic baselines and operational costs before they are trusted (Sommer & Paxson, 2010). Monitoring after a software update is the specific case that matters here: a post-update baseline of resource use, timing, network connections and privilege behaviour, compared against the pre-update baseline, is the control that would have caught the XZ payload earlier and more reliably than chance. NIST guidance on C-SCRM supports this approach by

emphasising visibility, monitoring, response planning and supplier risk management as part of a broader resilience programme (Boyens et al., 2022), and sector-specific threat intelligence allows emerging component compromises to be recognised quickly (Tounsi & Rais, 2018).

The practical solution is to create several stop-points in the attack path, which Figure 5 labelled S1 to S5. First, suppliers should be required to provide SBOMs, signed provenance and secure build attestations before software is accepted into OT (S1 and S2). Second, updates should be tested in a staging environment and compared against trusted baselines before deployment (S3). Third, OT devices should be segmented so that a compromised library on one gateway cannot automatically reach PLCs, HMIs or safety-related systems (S5). Fourth, administrative access should require strong identity verification and approved devices. Fifth, monitoring should look for unusual CPU use, timing changes, unexpected network connections and new privilege behaviour after updates (S4). These controls make the solution more realistic because they do not assume that every malicious component will be found before installation. Instead, they reduce the chance of installation, limit the damage if installation occurs, and improve the ability to detect and recover.

X. RESPONSE AND RECOVERY STRATEGY

Because this paper anticipates a future threat rather than analysing a past incident, the emphasis falls on the response capabilities that critical-infrastructure organisations should develop now so that they are prepared when a supply chain compromise reaches their OT environment. These capabilities map onto the familiar sequence of preparation, detection and analysis, containment, eradication, recovery and post-incident activity described in NIST SP 800-61 and the recovery guidance of SP 800-184 (Cichonski et al., 2012; Bartock et al., 2016), but each must be adapted to a threat that hides inside trusted software. Table 10 maps the phases to the capabilities required, notes which must be prepared in advance, and proposes a readiness measure for each; Figure 11 draws the resulting process as a state machine for a suspected compromised component.

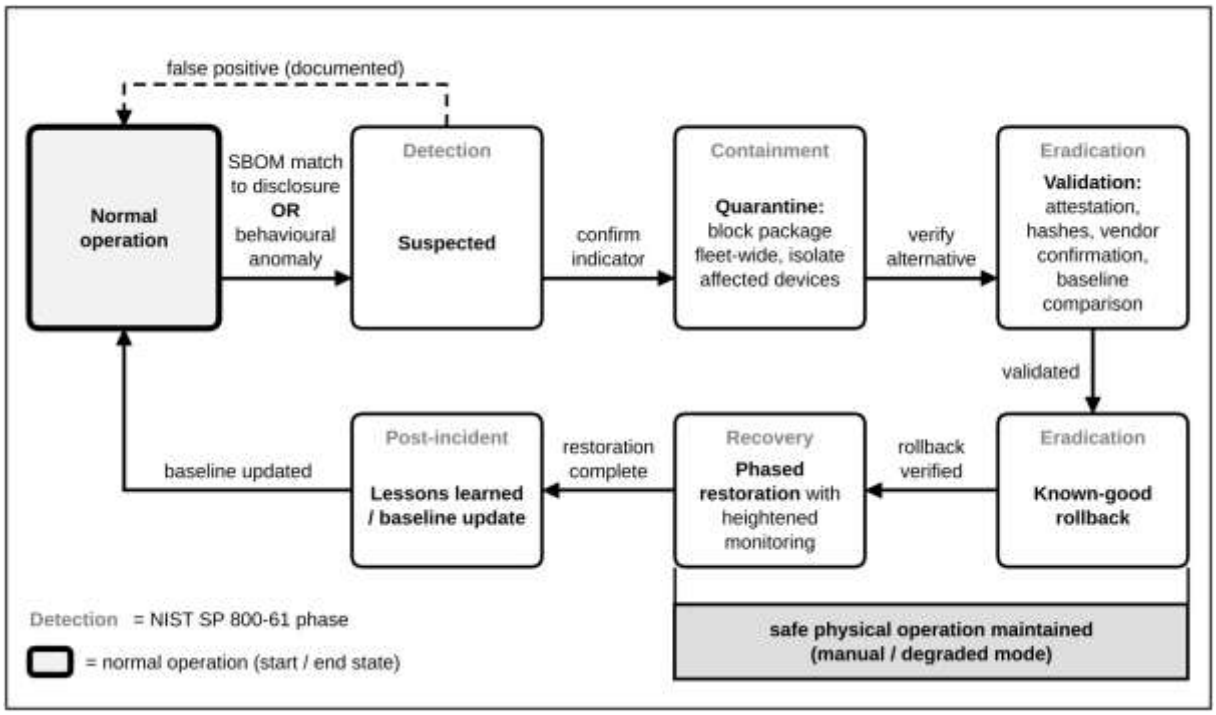
Table 10: Response and Recovery Capabilities for a Supply Chain Compromise in OT, Mapped to NIST SP 800-61 and SP 800-184 Phases

Phase (NIST SP 800-61 / 800-184)	Capability Required for a Supply-Chain Compromise in OT	Prepared in Advance	Metric or Evidence of Readiness
Preparation	SBOM-driven dependency inventory; validated software baselines; offline copies of firmware and configurations	Yes	Share of control-relevant devices with current SBOM; age of last baseline validation
Detection and analysis	Automated match of disclosures to inventory; behavioural monitoring of gateways and servers	Yes	Time from public disclosure to list of affected assets; alert coverage of post-update behaviour
Containment	Software quarantine: pause deployment, block package fleet-wide, isolate affected zones without halting safe operations	Yes	Time to fleet-wide block; existence of manual or degraded operating procedures
Eradication	Removal or rollback to known-good versions; confirmation through attestation, hashes and vendor statements	Partly	Share of devices restored to attested baseline; discrepancies found during validation
Recovery	Phased restoration with heightened monitoring; coordination with vendors, IT security and leadership	Partly	Restoration time per site; incidents during restoration
Post-incident activity	Lessons learned; supplier re-assessment; update of baselines, policies and drills	Yes	Closed actions from the review; drill frequency

Detection capability should centre on a comprehensive asset and dependency inventory driven by SBOMs, so that when a vulnerable or malicious component is disclosed an operator can quickly determine whether and where it is running. This inventory should be paired with behavioural monitoring of OT devices, since unusual resource use, authentication behaviour, network traffic or timing changes may be the earliest sign of a compromised component (Tafani-Dereeper & Knowles, 2024). Containment depends on segmentation prepared before the incident. The organisation should be able to isolate an affected gateway, historian, edge node or update server without shutting down

safe operations across the entire plant, which requires manual or degraded operating procedures to have been designed and exercised in advance (Stouffer et al., 2023; Cybersecurity and Infrastructure Security Agency, 2023a).

Figure 11: Response and Recovery State Machine for a Suspected Compromised Component, with NIST SP 800-61 Phases



Recovery planning must account for the reality that OT cannot always simply patch. Organisations should create trusted rollback paths, maintain validated software baselines, and keep offline copies of critical firmware, controller configurations and installation packages (Bartock et al., 2016). In the XZ response, a key action was returning to a known-good version, and the same principle applies to OT environments (Cybersecurity and Infrastructure Security Agency, 2024). Before restoring operations, the organisation should confirm that the software being reinstalled is trustworthy through attestation, hash verification, vendor confirmation and comparison with known-good baselines. Restoration should then proceed in phases, with heightened monitoring and clear communication among OT engineers, IT security, vendors and leadership. The state machine in Figure 11 makes two features explicit: a documented false-positive exit from the suspected state, so that the quarantine process is not discouraged by the cost of acting on weak signals, and a bar spanning quarantine to restoration that represents the obligation to maintain safe physical operation throughout, which distinguishes OT response from IT response (Kriaa et al., 2015; Stouffer et al., 2023).

The most impact-focused response capability is a software quarantine process for OT. If a component, update or vendor package is suspected of compromise, the organisation should be able to pause deployment, block the package from other sites, isolate systems that already received it and move critical services to a known-good configuration. This process should be practised through tabletop exercises and technical drills. The goal is not only to remove the malicious component but also to preserve safe physical operations while the investigation continues. Resilience research offers a useful framing: the measure of a response programme is the shape of the performance curve through the disruption, how far capability falls and how quickly it recovers, rather than whether the disruption was prevented (Linkov et al., 2013; Ganin et al., 2016).

XI. FUTURE SECURITY STRATEGY

A durable security strategy for the supply chain threat should be sequenced by organisational risk, addressing the highest-consequence, most tractable gaps first and building toward a comprehensive zero trust and C-SCRM posture. Table 11 sets out six priorities with their objectives, representative

controls, primary guidance and a suggested measure for each, and Figure 12 arranges them on a thirty-six-month roadmap whose durations are illustrative.

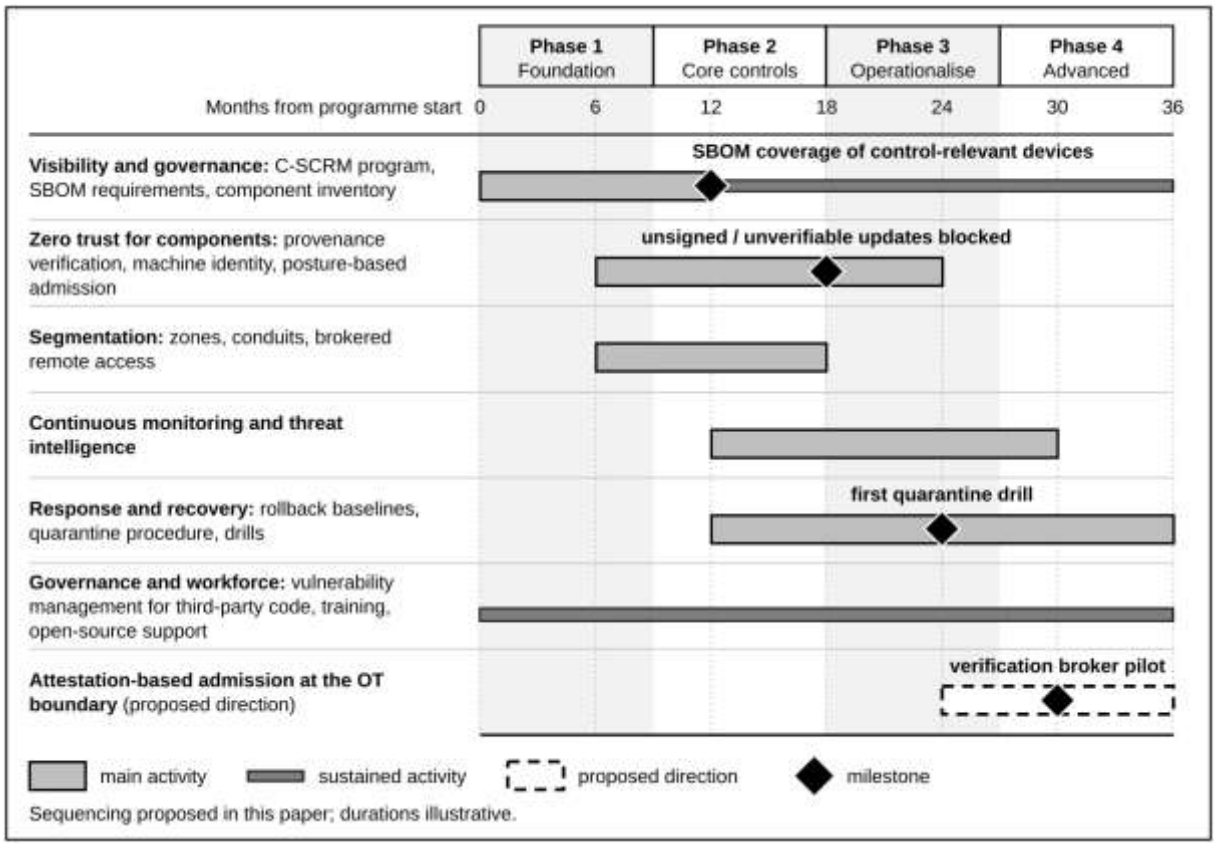
Table 11: Sequenced Security Strategy: Priorities, Controls, Guidance and Measures

Priority	Objective	Representative Controls	Primary Guidance	Suggested Measure
1. Visibility and governance	Know what software runs in OT and who is responsible for its assurance	C-SCRM programme; SBOM requirements in procurement; component inventory	NIST SP 800-161r1; NTIA minimum elements	SBOM coverage of control-relevant devices
2. Zero trust for components and devices	Admit only software whose provenance and integrity are verified	Provenance attestation; signature verification; machine identity; posture-based admission; strong entropy for keys	NIST SP 800-207; SLSA; SP 800-90B; SP 800-57	Share of updates with verified provenance; unsigned updates blocked
3. Segmentation	Prevent a compromised component from reaching control functions	Zones and conduits; brokered remote access; restricted update servers	ISA/IEC 62443-3-2; NIST SP 800-82 Rev. 3	Conduits with enforced policy; blocked lateral attempts
4. Continuous monitoring and intelligence	Detect behavioural deviation and recognise emerging component compromises	OT telemetry; post-update baselines; sector threat intelligence	NIST SP 800-82 Rev. 3; CISA CPG	Monitoring coverage; time from disclosure to assessment
5. Response and recovery	Contain and recover without unsafe operation	Software quarantine procedure; rollback baselines; drills	NIST SP 800-61; SP 800-184	Drill frequency; time to fleet-wide block
6. Governance and workforce	Sustain the programme and the ecosystem it depends on	Third-party vulnerability management; training; open-source support	NIST CSF 2.0 Govern; Secure by Design	Third-party vulnerabilities tracked to closure; projects supported

The first priority is visibility and governance: establishing a C-SCRM programme consistent with NIST SP 800-161r1, requiring SBOMs from vendors, and building a complete inventory of the software running in OT environments (Boyens et al., 2022; National Telecommunications and Information Administration, 2021). Without knowing what components are present, an operator can neither assess exposure nor respond when a component is compromised, so this foundational capability precedes all others. The Govern function introduced in the NIST Cybersecurity Framework 2.0 provides the organisational home for this programme and makes supply chain risk management an explicit outcome (National Institute of Standards and Technology, 2024a).

The second priority is to extend zero trust controls to components and devices by verifying provenance and integrity, enforcing least privilege and machine identity, and conditioning network admission on device posture (Rose et al., 2020; Cybersecurity and Infrastructure Security Agency, 2023b). Strong cryptographic entropy should support these controls because software signing, device attestation and machine identity all depend on unpredictable keys and tokens (Turan et al., 2018; Barker, 2020). The third priority is segmentation, so that a compromised component cannot move from an IIoT gateway or edge node into the control network (International Society of Automation, 2024; Stouffer et al., 2023). The fourth priority is continuous monitoring and threat-intelligence integration. This means instrumenting OT to detect abnormal behaviour, retaining enough telemetry for investigation, and using sector-specific threat intelligence so that emerging component compromises are recognised quickly (Tounsi & Rais, 2018; Cybersecurity and Infrastructure Security Agency, 2023a). The fifth priority is response and recovery, developed in Section X.

Figure 12: Strategy Roadmap for Extending Zero Trust to the OT Software Supply Chain (Durations Illustrative)



Underpinning these technical measures are governance and workforce priorities. Vulnerability management must expand to include third-party and open-source components, with defined processes for tracking disclosures and applying trusted updates or rollbacks. Security-awareness and secure-development training should also emphasise that suppliers, maintainers, package repositories and build systems are part of the attack surface. Mature organisations should support the resilience of the open-source projects they depend on through funding, contribution or participation, since the XZ case showed how the fragility of a single volunteer-maintained project can become a systemic risk (Ladisa et al., 2023; Eghbal, 2016; Champion & Hill, 2021). Ojo (2025) emphasises that zero trust delivers its benefits in critical infrastructure only when this governance layer supports the technical controls. Regulation is moving in the same direction: the European Union's Cyber Resilience Act obliges manufacturers of products with digital elements to handle vulnerabilities and provide security updates across a defined support period (European Parliament and Council, 2024), and Secure by Design principles ask vendors to take ownership of customer security outcomes (Cybersecurity and Infrastructure Security Agency et al., 2023), which gives operators contractual leverage that they did not previously have.

Priority solutions should be practical and measurable. Critical-infrastructure operators should require SBOMs and provenance evidence in procurement contracts, block unsigned or unverifiable updates from OT deployment, maintain a tested rollback plan for high-risk components, restrict update servers and vendor tools to tightly controlled network segments, and monitor software behaviour after deployment. They should also require suppliers to disclose build practices, dependency-management processes, known exploited vulnerabilities and incident-notification procedures. These actions turn zero trust from a general principle into a set of controls that can reduce operational risk, and the measures in Table 11 allow progress to be tracked. The roadmap in Figure 12 proposes four phases: foundation, in which governance and inventory are established; core controls, in which provenance verification, machine identity and segmentation are introduced; operationalisation, in which monitoring, response procedures and drills mature; and an advanced phase in which attestation-based

admission is piloted. The sequence is the author's proposal and should be adapted to each organisation's starting point and risk profile.

A Proposed Direction: Attestation-Based Defense for OT

Building on these controls, this paper proposes a forward-looking direction for OT: continuous automated provenance attestation at the point where industrial devices accept software. In this model, every firmware image, library and update used by a control-relevant device would carry independently generated, cryptographically signed provenance. The device, or a verification broker placed in front of it, would reject anything that could not be verified against policy (Open Source Security Foundation, 2023; Torres-Arias et al., 2019). The broker is the software admission enforcement point of Figure 9 and the verification stage of Figure 10; its inputs are the SBOM inventory, the provenance and attestation log, vulnerability intelligence and the device's own measured state, and its output is a per-deployment admission decision made by the policy engine rather than a one-time signature check (Rose et al., 2020). The cryptographic keys used for this process should be generated and protected with strong entropy sources. Cloudflare's lava lamp system is a useful example because it shows how physical randomness can strengthen cryptographic trust (Cloudflare, n.d.), and hardware random number generators with SP 800-90B health tests provide the equivalent for industrial devices (Turan et al., 2018; Trusted Computing Group, 2019).

This approach moves trust away from a one-time signature check and toward continuous verification of software lineage. It should also be paired with behavioural attestation, in which the system learns normal device behaviour and flags unexpected resource use, timing changes or network activity that may indicate a compromised component, extending the remote-attestation literature from what software is loaded to how it behaves (Abera et al., 2016; Ambrosin et al., 2020). Placing the broker at the OT boundary rather than on every device respects the constraints of legacy equipment: controllers that cannot verify provenance themselves can still be protected if nothing reaches them without passing the broker, which is consistent with the intermediary approach that SP 800-82 Rev. 3 recommends for devices that cannot participate in zero trust directly (Stouffer et al., 2023). The design is a direction rather than a validated architecture, and Section XII identifies the evaluation it would require.

XII. FUTURE RESEARCH

The XZ case and the broader supply chain threat expose limitations in current research and practice that define a clear agenda. The most pressing is the gap between zero trust theory and its implementation in resource-constrained OT. Much existing guidance is oriented toward enterprise IT and cloud environments, and the adaptation of continuous verification, attestation and SBOM-driven controls to legacy industrial devices that cannot tolerate latency or be readily patched remains underdeveloped (Feng & Hu, 2023; Buck et al., 2021). Future research should produce reference architectures and lightweight verification mechanisms tailored to industrial constraints, together with empirical evaluation of their operational impact on real-time control, using the testbeds and datasets that the CPS security community has developed (Conti et al., 2021; Eckhart & Ekelhart, 2019).

A second direction concerns the verification of software provenance and integrity at scale. Reproducible builds, cryptographic attestation and automated SBOM generation and validation are promising but still immature, particularly for the firmware and embedded software common in OT (Lamb & Zacchiroli, 2022; Xia et al., 2023; Carmody et al., 2021). Research is needed to make these techniques practical across heterogeneous industrial supply chains, to measure how often vendor firmware can be reproduced from source, and to establish what level of provenance assurance is achievable for safety-certified systems whose change control is governed by functional-safety standards (Kriaa et al., 2015).

A third direction follows from the human core of the XZ attack: methods for assessing and strengthening the security of open-source projects, especially critical single-maintainer components, and for detecting slow, socially engineered infiltration of trust. Existing metrics of truck factor, underproduction, weak links and security practice provide a starting point (Avelino et al., 2016; Champion & Hill, 2021; Zahan et al., 2022; Zahan et al., 2023), but none was designed to detect a patient adversary who behaves well for years. Future research should also examine how AI changes this threat. Specifically, researchers should study how attackers may use AI to find under-resourced projects, generate believable maintainer communications, create malicious code variants and test obfuscation

methods, and how defenders may use the same capabilities to flag anomalous contributor behaviour, diff release archives against repositories and review binary test data (Brundage et al., 2018; Mirsky et al., 2023; Heiding et al., 2024). The security of AI development tools as supply chain components in their own right deserves parallel attention (Schuster et al., 2021; Gu et al., 2019).

Finally, the risk assessment in Section VIII is qualitative because the data for quantitative estimation do not exist. Research that collects base rates for upstream compromise, time-to-discovery and dwell time in OT, and that evaluates the cost and effectiveness of the controls proposed here, would allow the ordinal ratings to be replaced with calibrated estimates (Hubbard & Seiersen, 2016; Aven, 2016). Addressing these limitations would move the field from reacting to incidents like XZ toward anticipating and preventing them. Progress in these areas would also give critical-infrastructure operators the evidence they need to justify investment in supply chain security before an incident forces the decision.

XIII. FINAL RECOMMENDATIONS

The XZ Utils backdoor demonstrates that the highest-priority emerging risk to cyber-physical systems is implicit trust in the software supply chain. The risk comes from assuming that a signed component from a familiar source is automatically safe to run. As OT converges with IT and industrial devices depend more heavily on open-source software, this assumption can give adversaries a scalable path toward physical processes through trusted channels that perimeter defences may not see. Artificial intelligence does not create this path, but the evidence reviewed here indicates that it lowers the cost of the patient social engineering, code manipulation and obfuscation that the path requires, so that attacks of the XZ type should be expected to become more frequent and less dependent on exceptionally resourced actors.

The immediate organisational priorities are clear. Operators should establish visibility into the software running in their OT environments by requiring SBOMs and building a component inventory, stand up a C-SCRM programme consistent with NIST SP 800-161r1, and institute behavioural monitoring capable of surfacing the anomalous activity that, in the XZ case, was the only thing that exposed the attack (Boyens et al., 2022; Tafani-Dereeper & Knowles, 2024). These measures are foundational and comparatively tractable, and they enable everything that follows.

In the longer term, organisations should extend Zero Trust Architecture to components and devices by verifying provenance and integrity, enforcing least privilege and machine identity, and segmenting OT so that a compromised component cannot reach control processes (Rose et al., 2020; National Institute of Standards and Technology, 2024a). They should also integrate continuous monitoring, mature vulnerability management, workforce training, supplier risk management, strong cryptographic entropy for signing and attestation, and threat intelligence, and they should prepare a software quarantine and known-good rollback capability before it is needed. The attestation-based admission control proposed in Section XI offers a direction in which these controls converge at the one point that every update must pass. The strategic conclusion is that resilience against this threat requires an architecture that refuses to trust software implicitly, verifies it continuously, and remains able to detect, contain and recover when a trusted component turns hostile.

REFERENCES

[1]. Abera, T., Asokan, N., Davi, L., Ekberg, J.-E., Nyman, T., Paverd, A., Sadeghi, A.-R., & Tsudik, G. (2016). C-FLAT: Control-flow attestation for embedded systems software. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 743–754). ACM. https://doi.org/10.1145/2976749.2978358

[2]. Alexander, O., Belisle, M., & Steele, J. (2020). *MITRE ATT&CK for industrial control systems: Design and philosophy*. The MITRE Corporation. https://attack.mitre.org/docs/ATTACK_for_ICS_Philosophy_March_2020.pdf

[3]. Ambrosin, M., Conti, M., Lazzeretti, R., Rabbani, M. M., & Ranise, S. (2020). Collective remote attestation at the Internet of Things scale: State-of-the-art and future challenges. *IEEE Communications Surveys & Tutorials*, *22*(4), 2447–2461. https://doi.org/10.1109/COMST.2020.3008879

[4]. Assante, M. J., & Lee, R. M. (2015). *The industrial control system cyber kill chain*. SANS Institute. https://www.sans.org/white-papers/36297/

[5]. Avelino, G., Passos, L., Hora, A., & Valente, M. T. (2016). A novel approach for estimating truck factors. In *2016 IEEE 24th International Conference on Program Comprehension (ICPC)* (pp. 1–10). IEEE. https://doi.org/10.1109/ICPC.2016.7503718

[6]. Aven, T. (2016). Risk assessment and risk management: Review of recent advances on their foundation. *European Journal of Operational Research*, *253*(1), 1–13. https://doi.org/10.1016/j.ejor.2015.12.023

[7]. Barker, E., & Kelsey, J. (2015). *Recommendation for random number generation using deterministic random bit generators* (NIST Special Publication 800-90A Rev. 1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-90Ar1

[8]. Barker, E. (2020). *Recommendation for key management: Part 1 – General* (NIST Special Publication 800-57 Part 1 Rev. 5). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-57pt1r5

[9]. Bartock, M., Cichonski, J., Souppaya, M., Smith, M., Witte, G., & Scarfone, K. (2016). *Guide for cybersecurity event recovery* (NIST Special Publication 800-184). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-184

[10]. Beerman, J., Berent, D., Falter, Z., & Bhunia, S. (2023). A review of Colonial Pipeline ransomware attack. *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW)*, 8–15. https://doi.org/10.1109/CCGridW59191.2023.00017

[11]. Bernstein, D. J., & Lange, T. (2017). Post-quantum cryptography. *Nature*, *549*(7671), 188–194. https://doi.org/10.1038/nature23461

[12]. Boyens, J., Smith, A., Bartol, N., Winkler, K., Holbrook, A., & Fallon, M. (2022). *Cybersecurity supply chain risk management practices for systems and organizations* (NIST Special Publication 800-161 Rev. 1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-161r1

[13]. Boyes, H. (2015). Cybersecurity and cyber-resilient supply chains. *Technology Innovation Management Review*, *5*(4), 28–34. https://doi.org/10.22215/timreview/888

[14]. Brundage, M., Avin, S., Clark, J., Toner, H., Eckersley, P., Garfinkel, B., Dafoe, A., Scharre, P., Zeitzoff, T., Filar, B., Anderson, H., Roff, H., Allen, G. C., Steinhardt, J., Flynn, C., Ó hÉigeartaigh, S., Beard, S., Belfield, H., Farquhar, S., . . . Amodei, D. (2018). *The malicious use of artificial intelligence: Forecasting, prevention, and mitigation* (arXiv:1802.07228). arXiv. https://arxiv.org/abs/1802.07228

[15]. Buck, C., Olenberger, C., Schweizer, A., Völter, F., & Eymann, T. (2021). Never trust, always verify: A multivocal literature review on current knowledge and research gaps of zero-trust. *Computers & Security*, *110*, 102436. https://doi.org/10.1016/j.cose.2021.102436

[16]. Cappos, J., Samuel, J., Baker, S., & Hartman, J. H. (2008). A look in the mirror: Attacks on package managers. In *Proceedings of the 15th ACM Conference on Computer and Communications Security* (pp. 565–574). ACM. https://doi.org/10.1145/1455770.1455841

[17]. Carmody, S., Coravos, A., Fahs, G., Hatch, A., Medina, J., Woods, B., & Corman, J. (2021). Building resilient medical technology supply chains with a software bill of materials. *npj Digital Medicine*, *4*, 34. https://doi.org/10.1038/s41746-021-00403-w

[18]. Champion, K., & Hill, B. M. (2021). Underproduction: An approach for measuring risk in open source software. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 388–399). IEEE. https://doi.org/10.1109/SANER50967.2021.00043

[19]. Cherdantseva, Y., Burnap, P., Blyth, A., Eden, P., Jones, K., Soulsby, H., & Stoddart, K. (2016). A review of cyber security risk assessment methods for SCADA systems. *Computers & Security*, *56*, 1–27. https://doi.org/10.1016/j.cose.2015.09.009

[20]. Cherepanov, A. (2017). *WIN32/INDUSTROYER: A new threat for industrial control systems*. ESET. https://web-assets.esetstatic.com/wls/2017/06/Win32_Industroyer.pdf

[21]. Cichonski, P., Millar, T., Grance, T., & Scarfone, K. (2012). *Computer security incident handling guide* (NIST Special Publication 800-61 Rev. 2). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-61r2

[22]. Cloudflare. (n.d.). *How do lava lamps help with Internet encryption?* Cloudflare. https://www.cloudflare.com/learning/ssl/lava-lamp-encryption/

[23]. Coelho, J., & Valente, M. T. (2017). Why modern open source projects fail. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017)* (pp. 186–196). ACM. https://doi.org/10.1145/3106237.3106246

[24]. Coker, G., Guttman, J., Loscocco, P., Herzog, A., Millen, J., O'Hanlon, B., Ramsdell, J., Segall, A., Sheehy, J., & Sniffen, B. (2011). Principles of remote attestation. *International Journal of Information Security*, *10*(2), 63–81. https://doi.org/10.1007/s10207-011-0124-7

[25]. Conti, M., Donadel, D., & Turrin, F. (2021). A survey on industrial control system testbeds and datasets for security research. *IEEE Communications Surveys & Tutorials*, *23*(4), 2248–2294. https://doi.org/10.1109/COMST.2021.3094360

[26]. Costin, A., Zaddach, J., Francillon, A., & Balzarotti, D. (2014). A large-scale analysis of the security of embedded firmwares. In *Proceedings of the 23rd USENIX Security Symposium* (pp. 95–110). USENIX Association. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/costin

[27]. Cox, L. A., Jr. (2008). What's wrong with risk matrices? *Risk Analysis*, *28*(2), 497–512. https://doi.org/10.1111/j.1539-6924.2008.01030.x

[28]. Cui, A., Costello, M., & Stolfo, S. J. (2013). When firmware modifications attack: A case study of embedded exploitation. In *Proceedings of the 2013 Network and Distributed System Security Symposium*. Internet Society. https://www.ndss-symposium.org/ndss2013/ndss-2013-programme/when-firmware-modifications-attack-case-study-embedded-exploitation/

[29]. Cybersecurity and Infrastructure Security Agency, National Security Agency, Federal Bureau of Investigation, & international partners. (2023). *Shifting the balance of cybersecurity risk: Principles and approaches for secure by design software*. Cybersecurity and Infrastructure Security Agency. https://www.cisa.gov/resources-tools/resources/secure-by-design

[30]. Cybersecurity and Infrastructure Security Agency. (2023a). *Cross-sector cybersecurity performance goals*. Cybersecurity and Infrastructure Security Agency. https://www.cisa.gov/cross-sector-cybersecurity-performance-goals

[31]. Cybersecurity and Infrastructure Security Agency. (2023b). *Zero trust maturity model, version 2.0*. Cybersecurity and Infrastructure Security Agency. https://www.cisa.gov/zero-trust-maturity-model

[32]. Cybersecurity and Infrastructure Security Agency. (2024). *Reported supply chain compromise affecting XZ Utils data compression library, CVE-2024-3094*. Cybersecurity and Infrastructure Security Agency. https://www.cisa.gov/news-events/alerts/2024/03/29/reported-supply-chain-compromise-affecting-xz-utils-data-compression-library-cve-2024-3094

[33]. Cárdenas, A. A., Amin, S., Lin, Z.-S., Huang, Y.-L., Huang, C.-Y., & Sastry, S. (2011). Attacks against process control systems: Risk assessment, detection, and response. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security* (pp. 355–366). ACM. https://doi.org/10.1145/1966913.1966959

[34]. Dhirani, L. L., Armstrong, E., & Newe, T. (2021). Industrial IoT, cyber threats, and standards landscape: Evaluation and roadmap. *Sensors*, *21*(11), 3901. https://doi.org/10.3390/s21113901

[35]. Duan, R., Alrawi, O., Kasturi, R. P., Elder, R., Saltaformaggio, B., & Lee, W. (2021). Towards measuring supply chain attacks on package managers for interpreted languages. In *Proceedings of the 2021 Network and Distributed System Security Symposium*. Internet Society. https://doi.org/10.14722/ndss.2021.23055

[36]. Eckhart, M., & Ekelhart, A. (2019). Digital twins for cyber-physical systems security: State of the art and outlook. *Security and Quality in Cyber-Physical Systems Engineering*, 383–412. https://doi.org/10.1007/978-3-030-25312-7_14

[37]. Eghbal, N. (2016). *Roads and bridges: The unseen labor behind our digital infrastructure*. Ford Foundation. https://www.fordfoundation.org/work/learning/research-reports/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/

[38]. Enck, W., & Williams, L. (2022). Top five challenges in software supply chain security: Observations from 30 industry and government organizations. *IEEE Security & Privacy*, *20*(2), 96–100. https://doi.org/10.1109/MSEC.2022.3142338

[39]. European Parliament and Council. (2024). *Regulation (EU) 2024/2847 on horizontal cybersecurity requirements for products with digital elements (Cyber Resilience Act)*. Official Journal of the European Union. https://eur-lex.europa.eu/eli/reg/2024/2847/oj

[40]. European Union Agency for Cybersecurity. (2021). *ENISA threat landscape for supply chain attacks*. ENISA. https://www.enisa.europa.eu/publications/threat-landscape-for-supply-chain-attacks

[41]. Executive Office of the President. (2021). *Improving the nation's cybersecurity (Executive Order 14028)*. Federal Register, 86 FR 26633. https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity

[42]. Falliere, N., Murchu, L. O., & Chien, E. (2011). *W32.Stuxnet dossier (Version 1.4)*. Symantec Corporation. https://docs.broadcom.com/doc/security-response-w32-stuxnet-dossier-11-en

[43]. Fang, R., Bindu, R., Gupta, A., & Kang, D. (2024). *LLM agents can autonomously exploit one-day vulnerabilities* (arXiv:2404.08144). arXiv. https://arxiv.org/abs/2404.08144

[44]. Feng, X., & Hu, S. (2023). Cyber-physical zero trust architecture for industrial cyber-physical systems. *IEEE Transactions on Industrial Cyber-Physical Systems*, *1*, 394–405. https://doi.org/10.1109/TICPS.2023.3333850

[45]. Galloway, B., & Hancke, G. P. (2013). Introduction to industrial control networks. *IEEE Communications Surveys & Tutorials*, *15*(2), 860–880. https://doi.org/10.1109/SURV.2012.071812.00124

[46]. Ganin, A. A., Massaro, E., Gutfraind, A., Steen, N., Keisler, J. M., Kott, A., Mangoubi, R., & Linkov, I. (2016). Operational resilience: Concepts, design and analysis. *Scientific Reports*, *6*, 19540. https://doi.org/10.1038/srep19540

[47]. Ghadge, A., Weiß, M., Caldwell, N. D., & Wilding, R. (2020). Managing cyber risk in supply chains: A review and research agenda. *Supply Chain Management: An International Journal*, *25*(2), 223–240. https://doi.org/10.1108/SCM-10-2018-0357

[48]. Giraldo, J., Urbina, D., Cardenas, A., Valente, J., Faisal, M., Ruths, J., Tippenhauer, N. O., Sandberg, H., & Candell, R. (2018). A survey of physics-based attack detection in cyber-physical systems. *ACM Computing Surveys*, *51*(4), Article 76. https://doi.org/10.1145/3203245

[49]. Goh, J., Adepu, S., Tan, M., & Lee, Z. S. (2017). Anomaly detection in cyber physical systems using recurrent neural networks. In *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)* (pp. 140–145). IEEE. https://doi.org/10.1109/HASE.2017.36

[50]. Gu, T., Liu, K., Dolan-Gavitt, B., & Garg, S. (2019). BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, *7*, 47230–47244. https://doi.org/10.1109/ACCESS.2019.2909068

[51]. Hadžiosmanović, D., Sommer, R., Zambon, E., & Hartel, P. H. (2014). Through the eye of the PLC: Semantic security monitoring for industrial processes. In *Proceedings of the 30th Annual Computer Security Applications Conference* (pp. 126–135). ACM. https://doi.org/10.1145/2664243.2664277

[52]. Hammi, B., & Zeadally, S. (2023). Software supply-chain security: Issues and countermeasures. *Computer*, *56*(7), 54–66. https://doi.org/10.1109/MC.2023.3273491

[53]. Hazell, J. (2023). *Spear phishing with large language models* (arXiv:2305.06972). arXiv. https://arxiv.org/abs/2305.06972

[54]. He, Y., Huang, D., Chen, L., Ni, Y., & Ma, X. (2022). A survey on zero trust architecture: Challenges and future trends. *Wireless Communications and Mobile Computing*, *2022*, 6476274. https://doi.org/10.1155/2022/6476274

[55]. Heiding, F., Schneier, B., Vishwanath, A., Bernstein, J., & Park, P. S. (2024). Devising and detecting phishing emails using large language models. *IEEE Access*, *12*, 42131–42146. https://doi.org/10.1109/ACCESS.2024.3375882

[56]. Hemsley, K. E., & Fisher, R. E. (2018). *History of industrial control system cyber incidents* (INL/CON-18-44411). Idaho National Laboratory. https://doi.org/10.2172/1505628

[57]. Heninger, N., Durumeric, Z., Wustrow, E., & Halderman, J. A. (2012). Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *Proceedings of the 21st USENIX Security Symposium* (pp. 205–220). USENIX Association. https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/heninger

[58]. Hiesgen, R., Nawrocki, M., Schmidt, T. C., & Wählisch, M. (2022). *The race to the vulnerable: Measuring the Log4j Shell incident* (arXiv:2205.02544). arXiv. https://arxiv.org/abs/2205.02544

[59]. Hubbard, D. W., & Seiersen, R. (2016). *How to measure anything in cybersecurity risk*. John Wiley & Sons. https://doi.org/10.1002/9781119162315

[60]. Humayed, A., Lin, J., Li, F., & Luo, B. (2017). Cyber-physical systems security—A survey. *IEEE Internet of Things Journal*, *4*(6), 1802–1831. https://doi.org/10.1109/JIOT.2017.2703172

[61]. Hutchins, E. M., Cloppert, M. J., & Amin, R. M. (2011). *Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains*. Lockheed Martin Corporation. https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf

[62]. International Electrotechnical Commission. (2023). *Understanding IEC 62443*. International Electrotechnical Commission. https://www.iec.ch/blog/understanding-iec-62443

[63]. International Society of Automation. (2024). *ISA/IEC 62443 series of standards*. International Society of Automation. https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards

[64]. Johnson, B., Caban, D., Krotofil, M., Scali, D., Brubaker, N., & Glyer, C. (2017). *Attackers deploy new ICS attack framework "TRITON" and cause operational disruption to critical infrastructure*. Mandiant. https://cloud.google.com/blog/topics/threat-intelligence/attackers-deploy-new-ics-attack-framework-triton

[65]. Joint Task Force Transformation Initiative. (2012). *Guide for conducting risk assessments* (NIST Special Publication 800-30 Rev. 1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-30r1

[66]. Kaloudi, N., & Li, J. (2020). The AI-based cyber threat landscape: A survey. *ACM Computing Surveys*, *53*(1), Article 20. https://doi.org/10.1145/3372823

[67]. Kaplan, S., & Garrick, B. J. (1981). On the quantitative definition of risk. *Risk Analysis*, *1*(1), 11–27. https://doi.org/10.1111/j.1539-6924.1981.tb01350.x

[68]. Kaur, R., Gabrijelčič, D., & Klobučar, T. (2023). Artificial intelligence for cybersecurity: Literature review and future research directions. *Information Fusion*, *97*, 101804. https://doi.org/10.1016/j.inffus.2023.101804

[69]. Knowles, W., Prince, D., Hutchison, D., Disso, J. F. P., & Jones, K. (2015). A survey of cyber security management in industrial control systems. *International Journal of Critical Infrastructure Protection*, *9*, 52–80. https://doi.org/10.1016/j.ijcip.2015.02.002

[70]. Kravchik, M., & Shabtai, A. (2018). Detecting cyber attacks in industrial control systems using convolutional neural networks. In *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy* (pp. 72–83). ACM. https://doi.org/10.1145/3264888.3264896

[71]. Kriaa, S., Pietre-Cambacedes, L., Bouissou, M., & Halgand, Y. (2015). A survey of approaches combining safety and security for industrial control systems. *Reliability Engineering & System Safety*, *139*, 156–178. https://doi.org/10.1016/j.ress.2015.02.008

[72]. Kuppusamy, T. K., DeLong, L. A., & Cappos, J. (2018). Uptane: Security and customizability of software updates for vehicles. *IEEE Vehicular Technology Magazine*, *13*(1), 66–73. https://doi.org/10.1109/MVT.2017.2778751

[73]. Kure, H. I., Islam, S., & Razzaque, M. A. (2018). An integrated cyber security risk management approach for a cyber-physical system. *Applied Sciences*, *8*(6), 898. https://doi.org/10.3390/app8060898

[74]. Ladisa, P., Plate, H., Martinez, M., & Barais, O. (2023). SoK: Taxonomy of attacks on open-source software supply chains. In *2023 IEEE Symposium on Security and Privacy (SP)* (pp. 1509–1526). IEEE. https://doi.org/10.1109/SP46215.2023.10179304

[75]. Lamb, C., & Zacchiroli, S. (2022). Reproducible builds: Increasing the integrity of software supply chains. *IEEE Software*, *39*(2), 62–70. https://doi.org/10.1109/MS.2021.3073045

[76]. Langner, R. (2011). Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, *9*(3), 49–51. https://doi.org/10.1109/MSP.2011.67

[77]. Linkov, I., Eisenberg, D. A., Plourde, K., Seager, T. P., Allen, J., & Kott, A. (2013). Resilience metrics for cyber systems. *Environment Systems and Decisions*, *33*(4), 471–476. https://doi.org/10.1007/s10669-013-9485-y

[78]. Lins, M., Mayrhofer, R., Roland, M., Hofer, D., & Schwaighofer, M. (2024). *On the critical path to implant backdoors and the effectiveness of potential mitigation techniques: Early learnings from XZ* (arXiv:2404.08987). arXiv. https://arxiv.org/abs/2404.08987

[79]. Makrakis, G. M., Kolias, C., Kambourakis, G., Rieger, C., & Benjamin, J. (2021). Industrial and critical infrastructure security: Technical analysis of real-life security incidents. *IEEE Access*, *9*, 165295–165325. https://doi.org/10.1109/ACCESS.2021.3133348

[80]. McLaughlin, S., Konstantinou, C., Wang, X., Davi, L., Sadeghi, A.-R., Maniatakos, M., & Karri, R. (2016). The cybersecurity landscape in industrial control systems. *Proceedings of the IEEE*, *104*(5), 1039–1057. https://doi.org/10.1109/JPROC.2015.2512235

[81]. Miller, T., Staves, A., Maesschalck, S., Sturdee, M., & Green, B. (2021). Looking back to look forward: Lessons learnt from cyber-attacks on industrial control systems. *International Journal of Critical Infrastructure Protection*, *35*, 100464. https://doi.org/10.1016/j.ijcip.2021.100464

[82]. Mirsky, Y., Demontis, A., Kotak, J., Shankar, R., Gelei, D., Yang, L., Zhang, X., Pintor, M., Lee, W., Elovici, Y., & Biggio, B. (2023). The threat of offensive AI to organizations. *Computers & Security*, *124*, 103006. https://doi.org/10.1016/j.cose.2022.103006

[83]. National Institute of Standards and Technology. (2024a). *The NIST Cybersecurity Framework (CSF) 2.0* (NIST CSWP 29). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.CSWP.29

[84]. National Institute of Standards and Technology. (2024b). *Module-lattice-based key-encapsulation mechanism standard* (FIPS 203). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.FIPS.203

[85]. National Telecommunications and Information Administration. (2021). *The minimum elements for a software bill of materials (SBOM)*. U.S. Department of Commerce. https://www.ntia.gov/report/2021/minimum-elements-software-bill-materials-sbom

[86]. Newman, Z., Meyers, J. S., & Torres-Arias, S. (2022). Sigstore: Software signing for everybody. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (pp. 2353–2367). ACM. https://doi.org/10.1145/3548606.3560596

[87]. Ohm, M., Plate, H., Sykosch, A., & Meier, M. (2020). Backstabber's knife collection: A review of open source software supply chain attacks. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2020), Lecture Notes in Computer Science, vol. 12223* (pp. 23–43). Springer. https://doi.org/10.1007/978-3-030-52683-2_2

[88]. Ojo, A. O. (2025). Adoption of Zero Trust Architecture (ZTA) in the protection of critical infrastructure. *Path of Science*, *11*(1), Article 5001. https://doi.org/10.22178/pos.113-2

[89]. Okafor, C., Schorlemmer, T. R., Torres-Arias, S., & Davis, J. C. (2022). SoK: Analysis of software supply chain security by establishing secure design properties. In *Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED '22)* (pp. 15–24). ACM. https://doi.org/10.1145/3560835.3564556

[90]. Open Source Security Foundation. (2023). *Supply-chain Levels for Software Artifacts (SLSA) specification, version 1.0*. OpenSSF. https://slsa.dev/spec/v1.0/

[91]. Ouyang, M. (2014). Review on modeling and simulation of interdependent critical infrastructure systems. *Reliability Engineering & System Safety*, *121*, 43–60. https://doi.org/10.1016/j.ress.2013.06.040

[92]. Oz, H., Aris, A., Levi, A., & Uluagac, A. S. (2022). A survey on ransomware: Evolution, taxonomy, and defense solutions. *ACM Computing Surveys*, *54*(11s), Article 238. https://doi.org/10.1145/3514229

[93]. Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., & Karri, R. (2022). Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions. In *2022 IEEE Symposium on Security and Privacy (SP)* (pp. 754–768). IEEE. https://doi.org/10.1109/SP46214.2022.9833571

[94]. Peisert, S., Schneier, B., Okhravi, H., Massacci, F., Benzel, T., Landwehr, C., Mannan, M., Mirkovic, J., Prakash, A., & Michael, J. B. (2021). Perspectives on the SolarWinds incident. *IEEE Security & Privacy*, *19*(2), 7–13. https://doi.org/10.1109/MSEC.2021.3051235

[95]. Qudus, L. (2025). Resilient systems: Building secure cyber-physical infrastructure for critical industries against emerging threats. *International Journal of Research Publication and Reviews*, *6*(1), 3330–3346. https://doi.org/10.55248/gengpi.6.0125.0514

[96]. ReversingLabs. (2025). *The 2025 software supply chain security report*. ReversingLabs. https://www.reversinglabs.com/sscs-report-2025

[97]. Rinaldi, S. M., Peerenboom, J. P., & Kelly, T. K. (2001). Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE Control Systems Magazine*, *21*(6), 11–25. https://doi.org/10.1109/37.969131

[98]. Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *Zero trust architecture* (NIST Special Publication 800-207). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207

[99]. Sadeghi, A.-R., Wachsmann, C., & Waidner, M. (2015). Security and privacy challenges in industrial Internet of Things. *Proceedings of the 52nd Annual Design Automation Conference*, Article 54. https://doi.org/10.1145/2744769.2747942

[100]. Sailer, R., Zhang, X., Jaeger, T., & van Doorn, L. (2004). Design and implementation of a TCG-based integrity measurement architecture. In *Proceedings of the 13th USENIX Security Symposium* (pp. 223–238). USENIX Association. https://www.usenix.org/conference/13th-usenix-security-symposium/design-and-implementation-tcg-based-integrity-measurement

[101]. Samuel, J., Mathewson, N., Cappos, J., & Dingledine, R. (2010). Survivable key compromise in software update systems. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (pp. 61–72). ACM. https://doi.org/10.1145/1866307.1866315

[102]. Sarker, I. H., Furhad, M. H., & Nowrozy, R. (2021). AI-driven cybersecurity: An overview, security intelligence modeling and research directions. *SN Computer Science*, *2*(3), 173. https://doi.org/10.1007/s42979-021-00557-0

[103]. Schuster, R., Song, C., Tromer, E., & Shmatikov, V. (2021). You autocomplete me: Poisoning vulnerabilities in neural code completion. In *Proceedings of the 30th USENIX Security Symposium* (pp. 1559–1575). USENIX Association. https://www.usenix.org/conference/usenixsecurity21/presentation/schuster

[104]. SentinelOne. (2024). *XZ Utils backdoor: Threat actor planned to inject further vulnerabilities*. SentinelOne. https://www.sentinelone.com/blog/xz-utils-backdoor-threat-actor-planned-to-inject-further-vulnerabilities/

[105]. Slowik, J. (2019). Evolution of ICS attacks and the prospects for future disruptive events. In *Dragos, Inc.*. https://www.dragos.com/wp-content/uploads/Evolution-of-ICS-Attacks-and-the-Prospects-for-Future-Disruptive-Events-Joseph-Slowik-1.pdf

[106]. Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy* (pp. 305–316). IEEE. https://doi.org/10.1109/SP.2010.25

[107]. Stalnaker, T., Wintersgill, N., Chaparro, O., Di Penta, M., German, D. M., & Poshyvanyk, D. (2024). BOMs away! Inside the minds of stakeholders: A comprehensive study of bills of materials for software systems. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)* (pp. 1–13). ACM. https://doi.org/10.1145/3597503.3623347

[108]. Stouffer, K., Pease, M., Tang, C., Zimmerman, T., Pillitteri, V., Lightman, S., Hahn, A., Saravia, S., Sherule, A., & Thompson, M. (2023). *Guide to operational technology (OT) security* (NIST Special Publication 800-82 Rev. 3). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-82r3

[109]. Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., & Thomas, C. B. (2018). *MITRE ATT&CK: Design and philosophy*. The MITRE Corporation. https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf

[110]. Syed, N. F., Shah, S. W., Shaghaghi, A., Anwar, A., Baig, Z., & Doss, R. (2022). Zero trust architecture (ZTA): A comprehensive survey. *IEEE Access*, *10*, 57143–57179. https://doi.org/10.1109/ACCESS.2022.3174679

[111]. Tafani-Dereeper, C., & Knowles, K. (2024). *The XZ Utils backdoor (CVE-2024-3094): Everything you need to know, and more*. Datadog Security Labs. https://securitylabs.datadoghq.com/articles/xz-backdoor-cve-2024-3094/

[112]. Ten, C.-W., Manimaran, G., & Liu, C.-C. (2010). Cybersecurity for critical infrastructures: Attack and defense modeling. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, *40*(4), 853–865. https://doi.org/10.1109/TSMCA.2010.2048028

[113]. Thompson, K. (1984). Reflections on trusting trust. *Communications of the ACM*, *27*(8), 761–763. https://doi.org/10.1145/358198.358210

[114]. Torres-Arias, S., Afzali, H., Kuppusamy, T. K., Curtmola, R., & Cappos, J. (2019). in-toto: Providing farm-to-table guarantees for bits and bytes. In *Proceedings of the 28th USENIX Security Symposium* (pp. 1393–1410). USENIX Association. https://www.usenix.org/conference/usenixsecurity19/presentation/torres-arias

[115]. Tounsi, W., & Rais, H. (2018). A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & Security*, *72*, 212–233. https://doi.org/10.1016/j.cose.2017.09.001

[116]. Trusted Computing Group. (2019). *Trusted Platform Module library specification, family 2.0, revision 1.59*. Trusted Computing Group. https://trustedcomputinggroup.org/resource/tpm-library-specification/

[117]. Turan, M. S., Barker, E., Kelsey, J., McKay, K. A., Baish, M. L., & Boyle, M. (2018). *Recommendation for the entropy sources used for random bit generation* (NIST Special Publication 800-90B). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-90B

[118]. Urbina, D. I., Giraldo, J. A., Cardenas, A. A., Tippenhauer, N. O., Valente, J., Faisal, M., Ruths, J., Candell, R., & Sandberg, H. (2016). Limiting the impact of stealthy attacks on industrial control systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1092–1105). ACM. https://doi.org/10.1145/2976749.2978388

[119]. Vu, D.-L., Pashchenko, I., Massacci, F., Plate, H., & Sabetta, A. (2020). Typosquatting and combosquatting attacks on the Python ecosystem. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (pp. 509–514). IEEE. https://doi.org/10.1109/EuroSPW51379.2020.00074

[120]. Ward, R., & Beyer, B. (2014). BeyondCorp: A new approach to enterprise security. *;login:*, *39*(6), 6–11. https://www.usenix.org/publications/login/dec14/ward

[121]. Xia, B., Bi, T., Xing, Z., Lu, Q., & Zhu, L. (2023). An empirical study on software bill of materials: Where we stand and the road ahead. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)* (pp. 2630–2642). IEEE. https://doi.org/10.1109/ICSE48619.2023.00219

[122]. Yadav, G., & Paul, K. (2021). Architecture and security of SCADA systems: A review. *International Journal of Critical Infrastructure Protection*, *34*, 100433. https://doi.org/10.1016/j.ijcip.2021.100433

[123]. Zahan, N., Kanakiya, P., Hambleton, B., Shohan, S., & Williams, L. (2023). OpenSSF Scorecard: On the path toward ecosystem-wide automated security metrics. *IEEE Security & Privacy*, *21*(6), 76–88. https://doi.org/10.1109/MSEC.2023.3279773

[124]. Zahan, N., Zimmermann, T., Godefroid, P., Murphy, B., Maddila, C., & Williams, L. (2022). What are weak links in the npm supply chain? In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '22)* (pp. 331–340). ACM. https://doi.org/10.1145/3510457.3513044

[125]. Zimmermann, M., Staicu, C.-A., Tenny, C., & Pradel, M. (2019). Small world with high risks: A study of security threats in the npm ecosystem. In *Proceedings of the 28th USENIX Security Symposium* (pp. 995–1010). USENIX Association. https://www.usenix.org/conference/usenixsecurity19/presentation/zimmerman