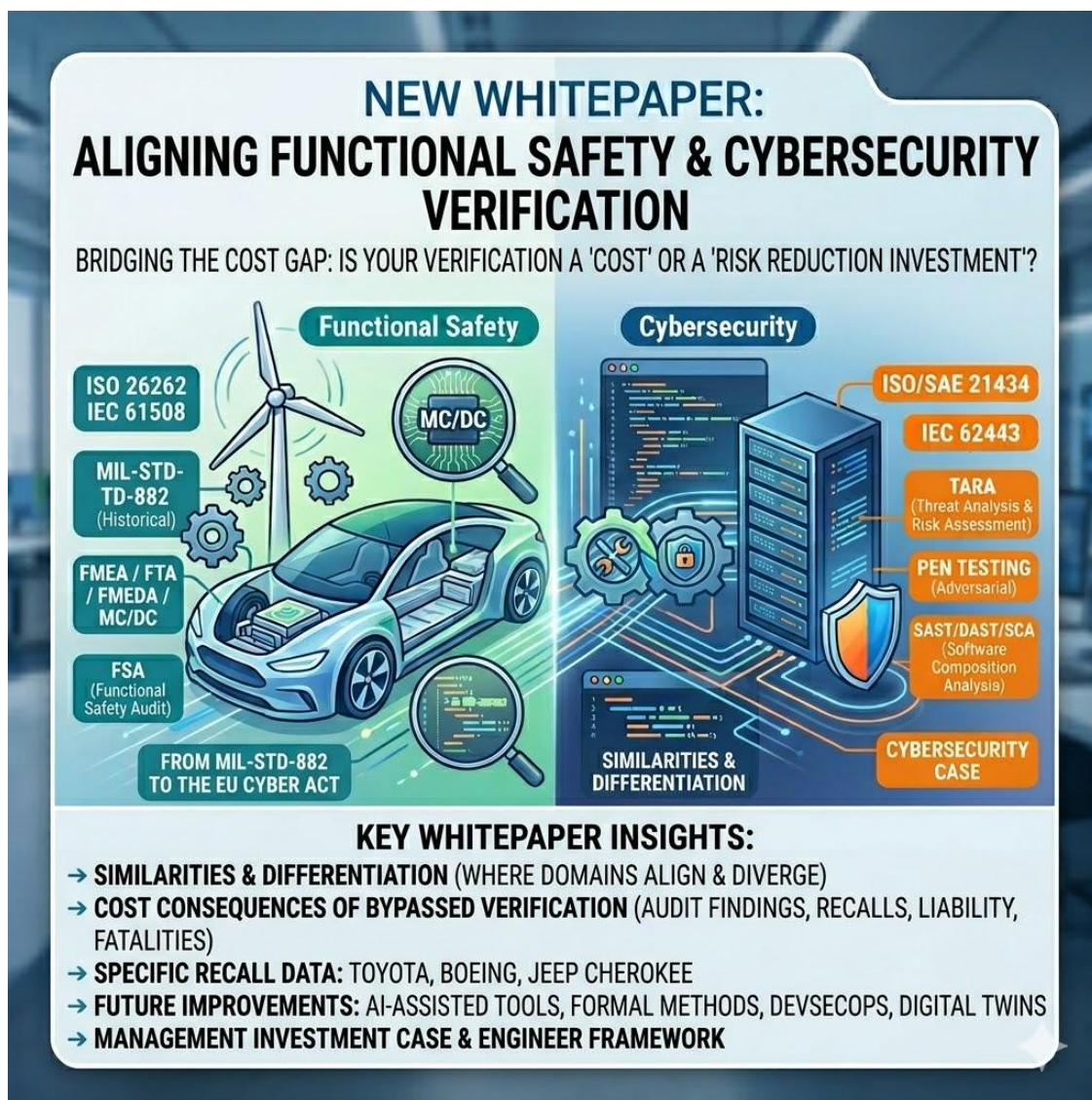


WHITEPAPER

Verification Measures in Functional Safety and Cybersecurity

Why the Investment Matters - A Cross-Domain Analysis for Technical Experts and Management



Revision 1.0

Status: Released

1. Overview

1.1 Change History

Version	Date	Author	Change Description
1.0	March 2026	M. Steinmetz	Initial version - whitepaper created

1.2 Purpose

This whitepaper examines **verification measures as a fundamental engineering practice** in the domains of functional safety and cybersecurity. It addresses a question that is as practical as it is strategic: **are the resources invested in verification measures proportionate to the risk they mitigate?** The paper provides evidence-based arguments for technical experts and decision-makers who must justify, allocate, and govern verification budgets in product development and organisational processes.

The document draws on established normative frameworks, primarily IEC 61508, ISO 26262, ISO/SAE 21434, and IEC 62443, and on publicly documented incident records where inadequate verification contributed to product failures, safety incidents, or security breaches. No speculative content is included; all claims are traceable to referenced sources or established engineering practice.

The paper is intended to serve as a foundation for professional publication and to motivate informed discussion among practitioners, assessors, auditors, and programme managers.

1.3 Definitions

Term	Definition
Verification	Confirmation, through the provision of objective evidence, that specified requirements have been fulfilled. (ISO 9000:2015, 3.8.12) Guiding question: Are we doing it right?
Validation	Confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled. (ISO 9000:2015, 3.8.13) Guiding question: Are we doing the right thing?
Functional Safety	Absence of unreasonable risk due to hazards caused by malfunctioning behaviour of electrical/electronic/programmable electronic safety-related systems. (IEC 61508-4:2010, 3.1.12)
Cybersecurity	Condition in which assets are sufficiently protected against threat scenarios with respect to road vehicles. (ISO/SAE 21434:2021, 3.7)
Safety Integrity Level (SIL)	Discrete level for specifying the safety integrity requirements of the safety functions. (IEC 61508-4:2010, 3.5.6)
Automotive Safety Integrity Level (ASIL)	One of four levels to specify the item's or element's necessary requirements to achieve an acceptable residual risk. (ISO 26262-1:2018, 3.6)

Term	Definition
Cybersecurity Assurance Level (CAL)	Cybersecurity assurance level that represents the targeted rigour of cybersecurity activities. (ISO/SAE 21434:2021, 3.6)
Security Level (SL)	Measure of confidence that the IACS is free from vulnerabilities. (IEC 62443-1-1:2009, 3.2.86)
TARA	Threat Analysis and Risk Assessment - systematic method to identify threat scenarios, their attack feasibility, and resulting risk. (ISO/SAE 21434:2021)
Safety Case	Structured argument, supported by a body of evidence, that provides a compelling, comprehensible and valid case that a system is safe for a given application. (IEC 61508-4:2010)
Residual Risk	Risk remaining after risk reduction measures have been applied. (ISO/IEC Guide 51:2014, 3.9)
Confirmation Measure	Activity used to provide evidence that a work product is adequate for its intended use. (ISO 26262-1:2018, 3.19)

1.4 Abbreviations

Abbreviation	Description
ASIL	Automotive Safety Integrity Level
CAL	Cybersecurity Assurance Level
CSMS	Cybersecurity Management System
DAST	Dynamic Application Security Testing
ECU	Electronic Control Unit
FMEA	Failure Mode and Effects Analysis
FMEDA	Failure Modes, Effects and Diagnostic Analysis
FSA	Functional Safety Assessment
FTA	Fault Tree Analysis
IACS	Industrial Automation and Control System
ICS	Industrial Control System
MC/DC	Modified Condition / Decision Coverage
PSIRT	Product Security Incident Response Team
SAST	Static Application Security Testing
SBOM	Software Bill of Materials
SCA	Software Composition Analysis
SDL	Secure Development Lifecycle
SIL	Safety Integrity Level

Abbreviation	Description
SL	Security Level
SPFM	Single-Point Fault Metric
TARA	Threat Analysis and Risk Assessment
V&V	Verification and Validation

1.5 References

Reference	Standard / Document
[IEC61508]	IEC 61508:2010 - Functional Safety of E/E/PE Safety-related Systems, Parts 1–7, IEC, 2nd edition
[ISO26262]	ISO 26262:2018 - Road Vehicles - Functional Safety, Parts 1–12, ISO, 2nd edition
[21434]	ISO/SAE 21434:2021 - Road Vehicles - Cybersecurity Engineering, ISO/SAE, 1st edition
[62443]	IEC 62443 Series - Security for Industrial Automation and Control Systems, Parts 1-1, 2-1, 3-3, 4-1, 4-2, IEC
[15288]	ISO/IEC/IEEE 15288:2023 - Systems and Software Engineering - System Life Cycle Processes, ISO/IEC/IEEE
[15026]	ISO/IEC/IEEE 15026-2:2011 - Systems and Software Engineering - Systems and Software Assurance - Part 2: Assurance Case
[IEEE1012]	IEEE 1012-2016 - Standard for System, Software, and Hardware Verification and Validation
[ISO9000]	ISO 9000:2015 - Quality Management Systems - Fundamentals and Vocabulary
[ISOGuide51]	ISO/IEC Guide 51:2014 - Safety Aspects - Guidelines for their Inclusion in Standards
[UNR155]	UN Regulation No. 155 - Uniform Provisions concerning the Approval of Vehicles with regard to Cyber Security, UNECE, 2021
[UNR156]	UN Regulation No. 156 - Uniform Provisions concerning the Approval of Vehicles with regard to Software Update Processes, UNECE, 2021
[CRA]	Regulation (EU) 2024/2847 - Cyber Resilience Act, European Parliament and Council, 2024
[NIS2]	Directive (EU) 2022/2555 - Network and Information Security Directive 2 (NIS2), European Parliament and Council, 2022
[NASA-GB]	NASA Software Safety Guidebook, NASA-GB-8719.13, 2004
[NHTSA15]	NHTSA/NASA Technical Assessment of Toyota Electronic Throttle Control Systems, NASA, 2011
[MV15]	Miller, C. & Valasek, C. - Remote Exploitation of an Unaltered Passenger Vehicle, DEF CON 23, 2015
[Leveson95]	Leveson, N.G. - Safeware: System Safety and Computers, Addison-Wesley, 1995
[Knight02]	Knight, J.C. - Safety Critical Systems: Challenges and Directions, ICSE 2002
[ACSF]	Boeing 737 MAX: Lessons for Engineering Ethics - IEEE Technology and Society Magazine, 2021
[Therac]	Leveson, N.G. & Turner, C.S. - An Investigation of the Therac-25 Accidents, IEEE Computer, 1993
[ENISA23]	ENISA - Good Practices for Security of IoT - Secure Software Development Lifecycle, ENISA, 2019
[BSI-TR]	BSI TR-03183-2 - Cyber Resilience Requirements for Manufacturers and Products, BSI, 2023

2. Historical Background and Motivation

2.1 The Origins of Systematic Verification in Engineering

The practice of systematic verification in safety-critical engineering did not emerge spontaneously from standardisation committees. It was driven by a series of catastrophic failures that demonstrated with painful clarity the consequences of insufficient evidence gathering in complex technical systems. Long before the first IEC or ISO safety standard was published, incidents in aviation, nuclear power, and defence programmes established the empirical foundation for what verification exists to prevent.

The post-war aerospace and defence sectors were among the first to formalise verification requirements. The US MIL-STD-882 standard, first issued in 1969, introduced the concept of a system safety programme that required documented evidence of hazard identification, analysis, and risk control. The corresponding UK DEF STAN 00-55 and 00-56 standards in the 1980s and 1990s extended this into software safety, recognising that software defects - unlike hardware failures - are systematic rather than random, and that they cannot be mitigated by redundancy alone. Software faults are present in every copy of a product; they do not average out over a population.

This recognition that systematic software faults require systematic verification - not just testing - was the intellectual foundation on which IEC 61508 was built. The broader insight from the defence and nuclear sectors was captured succinctly in Nancy Leveson's foundational work on system safety: accidents are rarely caused by a single component failure, but by the inadequate understanding of system behaviour under combined conditions. Verification exists to make system behaviour knowable.

2.2 The Functional Safety Standards Lineage

IEC 61508, published in its first edition in 1998 and revised in 2010, was conceived as a horizontal standard - applicable to any safety-related electrical, electronic, or programmable electronic system, regardless of industry sector. Its central contribution was not a new set of technical methods, but a structured risk-based framework: the Safety Integrity Level (SIL) concept, which calibrates the required rigour of verification to the magnitude of the risk being controlled. This risk-scaling principle broke from earlier prescriptive approaches and gave engineers a rational basis for deciding how much verification is enough.

ISO 26262, published in 2011 and revised in 2018, adapted this framework specifically for automotive electrical and electronic systems. The adaptation introduced the Automotive Safety Integrity Level (ASIL) classification and a product development process structure that closely follows the V-model lifecycle. The standard explicitly acknowledges IEC 61508 as its normative predecessor and preserves its core verification logic while adding automotive-specific elements such as hazard analysis and risk assessment (HARA) and the confirmation measures system.

Both standards embed a fundamental epistemological claim: that **a system's safety cannot be asserted - it must be argued**. The argument must be **supported by objective evidence** produced **through defined verification activities**. This shift from assertion to evidence-based argumentation is what distinguishes mature safety engineering from good intentions.

2.3 The Cybersecurity Standards Lineage

Cybersecurity standards in engineering contexts developed later and under different pressure. While functional safety standards were driven primarily by accident investigations, cybersecurity standards were driven by the recognition that connected systems create attack surfaces that no amount of reliability engineering can address. The threat model is fundamentally different: in functional safety, the adversary is physics and entropy; in cybersecurity, the adversary is an intelligent actor (human creativity) who adapts to the defences they encounter.

The watershed moment for automotive cybersecurity was the 2015 demonstration by researchers Charlie Miller and Chris Valasek that a production Jeep Cherokee could be remotely exploited over a cellular network, enabling an attacker to affect steering, braking, and transmission [MV15]. This was not a theoretical vulnerability - it was demonstrated in a moving vehicle at highway speed. The manufacturer, Fiat Chrysler Automobiles, responded with a 1.4 million vehicle recall. The incident demonstrated that the automotive industry's existing verification practices, built entirely around failure modes and reliability, were wholly inadequate for adversarial threats.

For industrial control systems (ICS), the Stuxnet malware, discovered in 2010, performed a similar function. It demonstrated that safety-critical industrial infrastructure - in that case uranium enrichment centrifuges - could be targeted by cyber-physical attacks that caused equipment to behave incorrectly while reporting normal operation to operators. Stuxnet exploited the absence of security verification in systems that had been designed exclusively around physical safety.

The normative response came in stages. IEC 62443, developed by the ISA/IEC joint committee, built on Microsoft's Security Development Lifecycle (SDL) methodology and defined security requirements for industrial automation systems across organisational, system, and component levels. ISO/SAE 21434, published in 2021 as a joint ISO/SAE effort, defined cybersecurity engineering requirements for road vehicles from concept through post-production, and is explicitly aligned with the UN Regulation No. 155 (UN R155) type approval requirement that became mandatory in July 2024 for new vehicle type approvals in the EU, Japan, and South Korea [UNR155].

The European regulatory landscape has since expanded significantly. The Cyber Resilience Act (CRA), adopted in late 2024, extends mandatory security verification requirements to virtually all products with digital elements placed on the EU market, introducing a legal duty of care that mirrors the liability logic already present in product safety law [CRA]. The NIS2 Directive, transposed into member state law from 2024, introduces sector-specific security obligations for operators of essential and important services. These regulatory developments mean that cybersecurity verification is no longer a domain of voluntary best practice - it is a legal obligation with enforcement consequences.

2.4 The Common Conceptual Foundation

Despite their different origins, functional safety and cybersecurity verification share a common conceptual ancestor in systems engineering theory. ISO/IEC/IEEE 15288, the international standard for system lifecycle processes, defines verification and validation as distinct but complementary activities applicable to any system lifecycle regardless of domain. Verification confirms that a work product conforms to its specified requirements; validation confirms that the system fulfils its intended use in its operational context [15288].

This distinction - and the structured lifecycle framework it supports - is the intellectual framework that both IEC 61508 and ISO/SAE 21434 implement, each scaled to the risk characteristics of their domain. ISO/IEC/IEEE 15026, the standard for systems and software assurance, provides the complementary framework for how verification

evidence is assembled into an assurance argument: the claim-argument-evidence (CAE) structure that underlies both the functional safety case and the cybersecurity case [15026].

The implication for practitioners and managers is significant: **verification in both domains** is not an isolated technical activity. It **is part of a coherent evidential system designed to demonstrate that residual risk is acceptable**. Understanding this system - its logical structure, its normative basis, and its real-world consequences when it fails - is the precondition for making sound investment decisions about verification resources.

3. Verification Measures in Functional Safety

3.1 The Normative Framework

Functional safety verification in electrical and electronic systems is governed primarily by IEC 61508 for general industrial applications and ISO 26262 for automotive road vehicles. Both standards adopt the same underlying principle: the rigour of verification activities must be commensurate with the Safety Integrity Level (SIL in IEC 61508) or Automotive Safety Integrity Level (ASIL in ISO 26262) assigned to the safety function being verified. This risk-scaling principle ensures that verification investment is proportionate to potential harm - not to programme convenience or cost pressure.

IEC 61508 uses a four-level scale (SIL 1–4), where SIL 4 represents the most demanding integrity requirements and is associated with safety functions whose failure could result in catastrophic and irreversible consequences. ISO 26262 uses a five-level scale (QM, ASIL A–D), where ASIL D corresponds to functions where a failure could result in severe or fatal injury with high probability, and QM (Quality Managed) applies to elements where the residual safety risk is acceptable without specific functional safety measures.

The standards define verification requirements across the product development lifecycle: from system level (ISO 26262 Part 4, IEC 61508 Part 1) through hardware design (ISO 26262 Part 5, IEC 61508 Part 2) to software development (ISO 26262 Part 6, IEC 61508 Part 3). At each lifecycle level, specific verification methods are either required (R) or recommended (HR/M - highly recommended or recommended) depending on the integrity level. This tiered method table structure is one of the most practical tools the standards provide for project planning.

3.2 Verification Methods in Detail

3.2.1 Reviews and Inspections

Reviews are the most broadly applicable verification method and are required at every lifecycle stage in both standards. They encompass requirements reviews, design reviews, code reviews, and work product reviews. The essential requirement that distinguishes safety-relevant reviews from routine quality reviews is independence: the reviewer must not have been responsible for creating the artefact being reviewed. At higher ASIL/SIL levels, independence requirements become more stringent - ASIL D (ISO 26262 Part 2, Clause 6) requires a functional safety assessment by a person or group that is independent of the project itself.

In practice, reviews serve three verification functions simultaneously. They confirm that a work product is internally consistent and complete. They verify traceability - that requirements are correctly derived from higher-level requirements and that design decisions are traceable to requirements. And they identify systematic errors that testing cannot reliably detect, such as requirement ambiguities, incorrect assumptions about operating conditions, or interface specification mismatches. The value of reviews is therefore not merely confirmatory; reviews are generative of safety evidence in a way that testing alone is not.

3.2.2 Analysis Methods

Analysis methods in functional safety verification are primarily directed at the hardware and system levels, where failure mode modelling provides the quantitative foundation for the safety case.

Failure Mode and Effects Analysis (FMEA) is a bottom-up, inductive method that systematically enumerates the failure modes of each component, their causes, and their effects on the system. In the context of ISO 26262, FMEA

supports the argument that all relevant failure modes have been identified and that each safety goal is protected against single-point failures. FMEA is required at both the system and hardware levels and is a normative input to the hardware architectural metrics.

Fault Tree Analysis (FTA) is a top-down, deductive method that starts from a feared event (a safety goal violation or a hazardous event) and systematically identifies the combinations of component failures that could cause it. FTA is particularly valuable for identifying common cause failures and latent faults that FMEA may miss because it approaches the problem from the effect rather than the cause.

Failure Mode, Effects and Diagnostic Analysis (FMEDA) extends FMEA to include the diagnostic coverage of each failure mode - the fraction of failures detected by on-board diagnostic mechanisms. FMEDA output directly populates the hardware architectural metrics required by ISO 26262 Part 5 and IEC 61508 Part 2: the Single-Point Fault Metric (SPFM) and Latent Fault Metric (LFM) for ISO 26262, and the Safe Failure Fraction (SFF) for IEC 61508. These metrics provide the quantitative argument that the hardware architecture achieves the required level of risk reduction.

The critical verification obligation associated with FMEDA is that the diagnostic coverage values used in the calculation must be verified - either through testing, historical data from comparable systems, or analysis. Assumed diagnostic coverage values that have not been verified represent one of the most common sources of inflated hardware architectural metrics in practice, and represent a significant finding risk in independent functional safety assessments.

3.2.3 Software Testing

Software testing in functional safety contexts is distinguished from general software testing by its structural coverage requirements. ISO 26262 Part 6 and IEC 61508 Part 3 specify that software unit testing must demonstrate structural coverage of the source code at a level commensurate with the ASIL/SIL:

- Statement coverage: required at ASIL A and SIL 1 - every executable statement must be executed at least once.
- Branch coverage: required at ASIL B/C and SIL 2/3 - every conditional branch (both the true and false outcomes) must be exercised.
- Modified Condition/Decision Coverage (MC/DC): required at ASIL D and SIL 3/4 - each condition within a compound decision must independently affect the decision outcome. MC/DC was adopted from the aviation DO-178B standard and provides a significantly more powerful fault-detection capability than branch coverage for safety-critical logic.

Coverage metrics are not goals in themselves - they are proxies for verification completeness. A test suite that achieves 100% MC/DC on poorly designed test cases provides much weaker evidence than one where test cases are systematically derived from requirements. Both standards therefore require that test cases be traced to requirements, that the test environment be representative of the production environment, and that deviations from the test specification be documented and justified.

Software integration testing and hardware-software integration testing verify the interfaces between modules and between software and hardware, respectively. These are among the most productive verification activities for finding faults in practice, because many systematic errors occur at interfaces - incorrect data types, missing range checks, incorrect assumption of hardware timing, and so on.

Back-to-back testing, a verification method specifically identified in ISO 26262 Part 6 for model-based development, involves executing the same test cases against both the model and the generated code and comparing results. This method verifies that code generation has not introduced deviations from the model behaviour.

3.2.4 Static Analysis

Static analysis - the analysis of source code without execution - is required for software at ASIL B and above in ISO 26262 Part 6. It covers coding guideline compliance (with MISRA C or equivalent rules widely used in automotive), control flow analysis, data flow analysis, and complexity metrics such as cyclomatic complexity.

Static analysis serves a different verification function than testing. Testing detects faults by observing incorrect behaviour under specific inputs. Static analysis detects certain classes of faults - uninitialised variables, undefined behaviour, dead code, overly complex control structures - that may not manifest under any available test case but that represent latent systematic defects. At high ASIL levels, both methods are required because they provide complementary evidence: testing demonstrates correct behaviour over a defined input space; static analysis provides coverage over the structural properties of the code.

3.3 Confirmation Measures

ISO 26262 Part 2 defines a set of measures it calls confirmation measures, which operate at a level above the object-level technical verification activities. These are functional safety audits, functional safety assessments (FSA), and functional safety reviews. They are not redundant with the verification activities described above - they verify that the verification activities themselves were correctly planned, scoped, executed, and documented.

The functional safety assessment (FSA) is the most demanding confirmation measure. It requires an independent review of the entire safety lifecycle, including the safety plan, HARA, safety concept, design verification evidence, test reports, analysis records, and the safety case. At ASIL D, independence must be external to the development project; at ASIL C, independence within the organisation but outside the project is generally acceptable. The FSA produces a report that either confirms the adequacy of the safety lifecycle or identifies open points that must be resolved before the product can be released.

From an audit perspective, the **confirmation measures are the most frequently underestimated aspect of the functional safety framework**. Many organisations plan and execute the technical verification activities adequately but treat the FSA as a project milestone to be completed rather than as an independent check on the validity of the safety argument. An FSA conducted by someone with no genuine independence from the development team, or conducted in the final week before SOP, provides far weaker assurance than the standard intends.

3.4 The Safety Case as the Integration Point

All verification activities in functional safety ultimately feed into the safety case - the structured argument that the product is safe for its intended use. ISO 26262 does not mandate a specific safety case format, but the claim-argument-evidence (CAE) structure defined in ISO/IEC/IEEE 15026 Part 2 is widely used and aligned with the standard's intent [15026].

In the CAE structure, the top-level claim is the safety goal: for example, 'The electronic power steering system shall not apply unintended steering torque in a manner that exceeds the severity threshold S3 / probability P2 / controllability C2.' The argument is the chain of logical steps that leads from the engineering decisions to the

fulfilment of the claim. The evidence is the verification output: test reports, FMEDA worksheets, review minutes, static analysis reports, FSA records.

The safety case serves two functions. Internally, it disciplines the development process by making the logical structure of the safety argument visible and auditable - gaps in the argument structure correspond to gaps in verification coverage. Externally, it is the primary deliverable for product assessment by a technical service, notified body, or type approval authority. Without a coherent safety case with complete and credible evidence, type approval cannot be granted.

4. Verification Measures in Cybersecurity

4.1 The Normative Framework

Cybersecurity verification for engineered products is governed by two primary normative frameworks: ISO/SAE 21434 for road vehicles and IEC 62443 for industrial automation and control systems (IACS). Both standards have been developed within the past fifteen years and reflect the maturation of the cybersecurity engineering discipline as it has expanded from IT security practice into safety-critical hardware product development.

ISO/SAE 21434, published in 2021, covers the entire lifecycle of cybersecurity engineering for road vehicles - from concept through development, production, operation, maintenance, and decommissioning. Its verification requirements appear primarily in Clause 10 (system-level product development), Clause 11 (hardware), Clause 12 (software), and Clause 13 (validation). The standard introduces the Cybersecurity Assurance Level (CAL 1–4) as the scaling factor that determines verification rigour, directly analogous to ASIL in ISO 26262. CAL is determined by the Threat Analysis and Risk Assessment (TARA), which is itself a verification activity - it verifies that the threat model is complete, that attack feasibilities are correctly assessed, and that the risk treatment decisions are justified.

IEC 62443 is a multi-part series. IEC 62443-2-1 defines Cybersecurity Management System (CSMS) requirements for operators of industrial automation systems. IEC 62443-3-3 defines system security requirements at four Security Levels (SL 1–4). IEC 62443-4-1 defines the Secure Development Lifecycle (SDL) process requirements for product suppliers, and IEC 62443-4-2 defines component security requirements. IEC 62443-4-1 is the most verification-rich part of the series, defining nine SDL practice groups from security management (SM) through incident response (IR), each with detailed requirements (RQ-xx) against which verification activities are mapped. A supplier's development process can be independently certified against IEC 62443-4-1, providing a process-level assurance certificate that applies to all products from that development environment.

4.2 Threat Analysis and Risk Assessment (TARA)

The TARA is the foundational analytical verification activity in cybersecurity engineering. It is required by ISO/SAE 21434 (Clause 15) and is structurally analogous to the HARA in ISO 26262 - it identifies the assets, threat scenarios, attack paths, and feasibility factors that determine the risk level and thereby the required depth of verification for each security measure.

The TARA process in ISO/SAE 21434 comprises several key steps. Asset identification establishes which system properties require protection - typically confidentiality, integrity, availability, and authenticity of specific data or functions. Threat scenario identification applies structured methods such as STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) to enumerate potential attacks on the identified assets. Attack path analysis traces the technical steps an attacker would need to take to realise a threat scenario, grounding the feasibility assessment in the specific system architecture rather than in generic threat catalogues.

The attack feasibility assessment is the most technically demanding aspect of the TARA and represents a key verification challenge. Feasibility factors defined in ISO/SAE 21434 include elapsed time, specialist expertise required, knowledge of the system, window of opportunity, and equipment needed. Each factor is rated to produce an overall feasibility score that, combined with an impact assessment, yields the risk value. Critically, the feasibility assessment requires verification against the actual implemented system - not the intended design -

because architectural decisions made during development may create attack paths that were not anticipated in the initial threat model.

The TARA is not a one-time verification activity. ISO/SAE 21434 requires cybersecurity monitoring and update processes (Clauses 8 and 9) that include periodic reassessment of the TARA in response to newly disclosed vulnerabilities, changed threat landscape conditions, and in-service feedback. This ongoing obligation reflects the fundamental difference between the functional safety and cybersecurity risk models: functional safety risks are fixed by the product design and do not change after release; cybersecurity risks are dynamic and adversarially driven.

4.3 Security Testing Methods

4.3.1 Penetration Testing

Penetration testing is the adversarial verification method that most directly simulates a real-world attack. In the context of ISO/SAE 21434, penetration testing is required as part of the cybersecurity validation activities in Clause 13 for items with non-trivial CAL assignments. In IEC 62443-4-1, penetration testing is addressed in the security testing requirements (RQ-08-03) as a mandatory component of the SDL for products targeting Security Level 2 and above.

Automotive penetration testing has evolved significantly since the mid-2010s and now encompasses a structured set of attack surfaces: external network interfaces (cellular, V2X, Wi-Fi, Bluetooth), diagnostic interfaces (UDS over CAN, DoIP over Ethernet), in-vehicle network communications (CAN, LIN, Ethernet, FlexRay), external connectors, and update mechanisms (OTA, USB). The depth of testing required depends on the CAL/SL assignment - a CAL 4 item with a remote attack surface and high impact requires substantially more extensive penetration testing than a CAL 1 item accessible only through physical means.

A critical quality requirement for penetration testing is tester independence: the testing must be conducted by professionals who were not involved in the design of the security measures being tested. This independence requirement mirrors the confirmation measure independence requirements in functional safety and exists for the same reason - the designers of a security architecture have inherent blind spots about its weaknesses.

4.3.2 Fuzz Testing

Fuzz testing involves the automated generation of malformed, unexpected, or randomly mutated inputs to interfaces and protocols, with the objective of identifying crash-inducing inputs, protocol implementation errors, and boundary violations that may be exploitable. IEC 62443-4-1 addresses fuzz testing in the security testing requirements (RQ-08-02), and it is referenced in ISO/SAE 21434 as an applicable verification method for software-level cybersecurity verification.

In automotive and industrial contexts, fuzz testing is applied to external interfaces (CAN, Ethernet, Bluetooth, diagnostic protocols such as UDS), to over-the-air update message handling, and to API endpoints exposed by connectivity components. The technique is particularly effective at finding implementation-level vulnerabilities in communication stack parsers that are not exposed by specification-based test cases, because fuzzing explores the space beyond the specification boundary.

4.3.3 Static and Dynamic Analysis Tools

Static Application Security Testing (SAST) tools analyse source code, bytecode, or binary code without execution to identify security-relevant coding defects: buffer overflows, injection vulnerabilities, use of deprecated cryptographic primitives, memory management errors, and insecure randomness. IEC 62443-4-1 requires the use of security-focused static analysis (RQ-07-01, RQ-07-02) as part of the implementation review requirements of the SDL.

Dynamic Application Security Testing (DAST) complements SAST by testing the running application. DAST tools send crafted inputs to running interfaces and monitor for error conditions, unexpected behaviour, and security-relevant outputs. For embedded systems, this overlaps significantly with penetration testing methodology; for web-based management interfaces of IACS components, commercial DAST tools are directly applicable.

Software Composition Analysis (SCA) addresses a verification challenge that is specific to modern software development: the use of third-party and open-source components. SCA tools parse a Software Bill of Materials (SBOM) or analyse the build environment to identify component versions and cross-reference them against known vulnerability databases (NVD, OSV, vendor advisories). ISO/SAE 21434 addresses SBOM management in its requirements for third-party component management (Clause 12), and the CRA introduces a mandatory SBOM obligation for CE-marked products in the EU [CRA]. SCA is therefore both a verification activity and a compliance obligation in the current regulatory environment.

4.4 Cybersecurity Assessment and Audit

Analogous to the functional safety assessment in ISO 26262, cybersecurity verification in ISO/SAE 21434 culminates in a cybersecurity assessment (Clause 15) that provides an independent evaluation of the cybersecurity case. The assessment examines whether the TARA was complete and correctly scoped, whether the cybersecurity goals derived from the TARA are adequate, whether the implemented cybersecurity measures address the identified risks, and whether the verification evidence supports the risk acceptance decision.

For IEC 62443, the analogous process is the CSMS audit (per IEC 62443-2-1 at the operator level) and the SDL certification audit (per IEC 62443-4-1 at the supplier level). An IEC 62443-4-1 certification issued by an accredited certification body confirms that the supplier's development process systematically meets the SDL process requirements - it is a process-level assurance certificate, not a product-level certificate, but it provides a significant input to product-level security assessments by demonstrating that the verification activities described in the SDL were consistently applied.

The TISAX® (Trusted Information Security Assessment Exchange) scheme, widely used in the automotive supply chain, provides an equivalent assessment mechanism for information security management aligned with ISO/IEC 27001. While TISAX is primarily an information security framework rather than a product cybersecurity framework, it provides the organisational security foundation on which product cybersecurity engineering processes depend.

4.5 The Cybersecurity Case

As in functional safety, cybersecurity verification activities collectively build towards a cybersecurity case - the structured argument that the cybersecurity risks of a product are acceptably treated. ISO/SAE 21434 Clause 15 explicitly requires a cybersecurity case and defines its content: the scope, the cybersecurity goals and their

assignment to TARA findings, the cybersecurity claims for each item, the evidence supporting those claims, and the rationale for risk acceptance decisions.

The cybersecurity case differs from the safety case in one important structural respect: it does not aim to demonstrate zero residual risk (which is unachievable against an adaptive adversary) but rather to demonstrate that risks have been treated to a level commensurate with the CAL assignment, and that the organisation is equipped to respond to residual risks that materialise post-production through the PSIRT and post-production monitoring processes.

5. Similarities and Required Differentiation

5.1 The Shared Epistemological Structure

A reader encountering the verification frameworks of ISO 26262 and ISO/SAE 21434 for the first time will notice a **striking structural similarity**: both require reviews, analyses, testing, and independent confirmation at each lifecycle stage; both scale their requirements to an integrity or assurance level derived from a risk assessment; both culminate in a structured argument supported by objective evidence; and both require independent confirmation measures that verify the adequacy of the process itself. This similarity is not coincidental.

The common intellectual ancestor is the systems engineering tradition embodied in ISO/IEC/IEEE 15288, which defines verification and validation as generic lifecycle processes applicable to any engineered system [15288]. The assurance case structure shared by both domains traces to ISO/IEC/IEEE 15026 [15026]. Both functional safety and cybersecurity standards are specialisations of this generic framework, adapted to the risk characteristics of their respective domains. Understanding this shared architecture allows practitioners to transfer competencies between domains and allows organisations to build integrated verification programmes rather than parallel, redundant ones.

Table 5.1 summarises the structural parallels between the two domains.

Verification Element	Functional Safety	Cybersecurity
Risk scaling	ASIL A–D (ISO 26262) / SIL 1–4 (IEC 61508)	CAL 1–4 (21434) / SL 1–4 (IEC 62443)
Risk assessment method	HARA (ISO 26262 Part 3)	TARA (ISO/SAE 21434 Cl. 15)
Analysis	FMEA, FTA, FMEDA	STRIDE, threat modelling, attack path analysis
Testing	Unit / integration / system testing with coverage metrics	Pen testing, fuzz testing, SAST / DAST / SCA
Code-level verification	Static analysis (MISRA, complexity)	SAST, secure coding guidelines
Confirmation measure	Functional Safety Assessment (FSA)	Cybersecurity Assessment (Cl. 15 / IEC 62443-4-1 audit)
Structured argument	Safety case (CAE structure per 15026)	Cybersecurity case (CAE structure per 21434 Cl. 15)
Independence requirement	Scales with ASIL - external for ASIL D	Required for assessment; tester independence for pen test
Post-production obligation	Warranty monitoring, field feedback	PSIRT, vulnerability monitoring, TARA update

Table 5.1: Structural parallels between functional safety and cybersecurity verification

5.2 Required Differentiations

5.2.1 The Nature of the Adversary

The most fundamental distinction between functional safety and cybersecurity risk is the nature of the threat. Functional safety addresses random hardware failures and systematic faults - phenomena governed by physics,

chemistry, and the laws of probability. Probability distributions can be measured, defect densities can be estimated, and diagnostic coverage can be verified. The 'adversary' in functional safety is not intelligent: a voltage transient does not adapt its behaviour in response to the defensive measures it encounters.

Cybersecurity addresses adversarial threats from intelligent actors who observe, learn, and adapt (human creativity). An attacker who discovers that one attack path is blocked will search for an alternative. Countermeasures that are effective against today's attack techniques may be ineffective against techniques developed after deployment. This adaptive nature of the threat means that cybersecurity risk cannot be fully characterised at development time - it evolves throughout the product's operational life. The cybersecurity verification framework must therefore include not only development-phase verification but ongoing monitoring, post-production vulnerability management, and the organisational capability to respond when residual risks materialise.

5.2.2 Verification Horizon and Post-Production Obligations

In functional safety, the verification horizon largely ends at product release. Once a product achieves its safety integrity level and the safety case is accepted, the safety risk profile is fixed - additional verification activities are only triggered by design changes or field incidents. The functional safety standards do include requirements for monitoring and field feedback, but these are relatively lightweight compared to the development-phase verification obligations.

In cybersecurity, the verification horizon extends indefinitely after product release. ISO/SAE 21434 dedicates an entire clause (Clause 8) to the post-development cybersecurity processes, including vulnerability management, security monitoring, and incident response. IEC 62443 requires the operator to maintain a CSMS that includes ongoing security monitoring and response capabilities. UN R155 requires OEMs to maintain a type-approved CSMS for the entire production lifetime of a vehicle type and for a defined period after production ends [UNR155].

This difference in verification horizon has direct budgetary implications. Cybersecurity verification is not a project cost that ends at product launch - it is an operational cost that must be provisioned for the product's entire market life.

5.2.3 Quantification and Acceptance Criteria

Functional safety provides clear quantitative acceptance criteria for hardware: ISO 26262 Part 5 specifies numerical targets for SPFM and LFM by ASIL level, and IEC 61508 specifies PFH (probability of dangerous failure per hour) targets by SIL level. These targets can be verified by calculation using FMEDA, and the verification is reproducible - two engineers using the same data and method will obtain the same result.

Cybersecurity does not offer equivalent quantitative metrics. Attack feasibility is rated on ordinal scales rather than probability distributions; the impact of a security compromise depends on attacker intent and capability, which cannot be quantified in the same way as failure rates. The cybersecurity case must therefore rely more heavily on qualitative arguments (note: importance of threat modelling), supported by the depth and scope of verification activities, rather than on quantitative targets. This difference in the nature of the evidence required places a greater burden on the judgement of assessors and auditors and makes consistent verification of cybersecurity cases across supply chain participants more challenging.

5.2.4 Independence Structures

Both domains require independence for confirmation measures, but the independence structures differ in important ways. ISO 26262 defines independence levels precisely - dependent, independent within the project, independent within the organisation, and external - and maps them to ASIL levels with normative force. An ASIL D product that has not undergone an externally independent FSA does not comply with the standard, regardless of the technical quality of the development work.

ISO/SAE 21434 requires independence for the cybersecurity assessment but is less prescriptive about what 'independence' means in practice. IEC 62443-4-1 certification does impose third-party independence through the requirement for an accredited certification body, but this is process-level certification rather than product-level assessment. The practical result is that cybersecurity independence requirements, while genuinely present in the standards, leave more room for interpretation - and therefore more room for insufficient independence to masquerade as adequate independence.

6. Consequences of Bypassed or Insufficient Verification

6.1 Categories of Verification Failure

Before examining specific cases, it is useful to distinguish the ways in which verification can fail in practice. Five categories are observed repeatedly in incident investigations and audit findings, each with different technical and organisational causes:

- a) **Planned but not executed:** the verification activity appears in the safety or security plan, a report exists, but the underlying work was either not done or was done superficially. This produces paper compliance while leaving defects or vulnerabilities undetected. It is the most dangerous category because it creates false confidence in a system's risk profile.
- b) **Executed but not documented:** the work was done but there is no retrievable evidence. From an assessor's or auditor's perspective, this is indistinguishable from not done. The evidence obligation is not bureaucratic formalism - it exists because an independent assessor cannot otherwise distinguish real verification from asserted verification.
- c) **Documented but incorrectly scoped:** testing or analysis was executed against the wrong configuration, an incomplete set of requirements, or a non-representative build. Coverage metrics satisfy numeric targets; the actual risk coverage is unknown.
- d) **Independence requirement violated:** verification was performed by the person or team responsible for the artefact being verified. At higher ASIL/SIL/CAL levels, this violates the normative independence requirement regardless of the technical quality of the work, because independence is a structural safeguard against systematic blind spots.
- e) **Methods not commensurate with integrity level:** a lower-cost method was substituted for a required higher-rigour method - for example, branch coverage used instead of MC/DC at ASIL D, or a design review used instead of a formal FSA - without documented justification or equivalence argument.

6.2 Historical Incidents in Functional Safety

6.2.1 The Therac-25 Radiation Therapy Accidents (1985–1987)

The Therac-25 medical linear accelerator, produced by Atomic Energy of Canada Limited (AECL), was responsible for six documented cases of radiation overdose between 1985 and 1987, resulting in the deaths of at least three patients. The accidents have become the most extensively studied case study in software safety engineering and are described in detail in Leveson and Turner's 1993 investigation [Therac].

The Therac-25 removed hardware safety interlocks that had been present in earlier models, relying instead on software to prevent dangerous configurations. The software had been partially reused from the earlier hardware-interlocked models, where the software errors were harmless because the hardware provided an independent safety barrier. Without adequate software verification, a race condition in the operator interface software went undetected. When certain keying sequences were entered at a specific speed, the machine could be commanded to deliver high-energy electron beams at full power without the beam-spreading hardware in place, resulting in exposures orders of magnitude beyond therapeutic doses.

The verification failures were systemic: there was no formal V&V plan, no independent code review, no structured testing of the race condition scenarios, and no meaningful separation between the development team and the

verification function. The hardware safety functions had been removed and replaced with software without any corresponding increase in the rigour of software verification. This case directly informed the development of IEC 62304 (software for medical devices) and contributed substantially to the intellectual development of IEC 61508.

6.2.2 Toyota Unintended Acceleration (2009–2010)

From 2009 to 2010, Toyota faced more than 8 million vehicle recalls related to reports of unintended acceleration. The NHTSA engaged NASA to conduct an independent technical assessment of Toyota's Electronic Throttle Control Systems (ETCS). The NASA technical assessment, published in 2011, found evidence of deficiencies in the software development and verification processes [NHTSA15].

Key findings relevant to functional safety verification included: inadequate stack overflow analysis (the potential for task stack overflows to corrupt adjacent memory areas had not been systematically analysed or tested), insufficient worst-case execution time analysis, use of global variables in ways that created unpredictable interactions under specific timing conditions, and software architecture that did not implement the kind of systematic fault containment that rigorous IEC 61508 or ISO 26262 verification would have required. The software had been developed under what the investigation characterised as inadequate software safety processes.

The commercial consequences were severe: direct recall costs exceeded US\$3 billion, legal settlements in class action and personal injury suits added substantially to this figure, and the reputational damage affected Toyota's market position for several years.

6.2.3 Boeing 737 MAX MCAS (2018–2019)

The accidents involving Lion Air Flight JT610 (October 2018) and Ethiopian Airlines Flight ET302 (March 2019), which together killed 346 people, were attributed to the behaviour of the Manoeuvring Characteristics Augmentation System (MCAS) software. Subsequent investigations by the Joint Authorities Technical Review (JATR) and the House Transportation Committee identified specific failures in the safety assessment and software verification processes [ACSF].

The safety analysis had categorised a single angle-of-attack (AOA) sensor failure as a 'hazardous' rather than 'catastrophic' condition, based on an assumption that pilots would be able to identify and respond to MCAS activation using a published runaway trim procedure. The verification testing did not cover the full operational envelope in which MCAS could activate, including the scenario where repeated MCAS activations made the stabiliser trim forces too large for the manual trim wheel to overcome. The independence of the safety assessment was also questioned: elements of the assessment had been delegated to Boeing under the FAA's Organisation Designation Authorization programme, reducing the degree of independent scrutiny applied.

The 737 MAX was grounded worldwide for nearly 20 months. The human cost - 346 fatalities - cannot be quantified financially, but the direct cost to Boeing has been estimated at over US\$20 billion including compensation, legal settlements, and production disruption.

6.3 Historical Incidents in Cybersecurity

6.3.1 Jeep Cherokee Remote Exploit (2015)

In 2015, security researchers Charlie Miller and Chris Valasek demonstrated the remote exploitation of a production 2014 Jeep Cherokee over the Sprint cellular network, using a vulnerability in the Uconnect infotainment system [MV15]. The exploit allowed the researchers to control heating, ventilation, air conditioning,

the windscreen wipers, the radio, and - most critically - the transmission and brakes, while a test driver operated the vehicle at highway speed with no foreknowledge of the attack.

The technical root cause was a combination of insufficient network segmentation (the cellular modem was on the same network as CAN-connected ECUs), an unauthenticated D-Bus interface accessible from the infotainment system, and the absence of any meaningful security testing of the cellular interface at the system integration level. The cellular interface had never been subjected to adversarial security testing. The attack surface was discoverable, accessible from the internet, and exploitable with modest attacker capability.

Fiat Chrysler issued a recall affecting 1.4 million vehicles - the first recall in automotive history explicitly motivated by a cybersecurity vulnerability. The incident was a primary catalyst for the development of UN R155 and the mandatory CSMS requirements that followed [UNR155].

6.3.2 Stuxnet and Industrial Control System Vulnerabilities (2010)

The Stuxnet malware, discovered in 2010 and attributed by multiple intelligence assessments to a state-sponsored actor, targeted Siemens S7 programmable logic controllers used in uranium enrichment centrifuges at the Natanz facility in Iran. Stuxnet exploited four previously unknown Windows vulnerabilities and one known vulnerability to propagate and install itself on engineering workstations. It then reprogrammed the PLCs to drive centrifuges at speeds that caused physical damage while reporting normal operating parameters to control room displays.

From a verification perspective, Stuxnet demonstrated that the absence of security verification in safety-critical industrial systems creates exploitable vulnerabilities that have physical consequences. The PLCs and their network connections had been designed and verified entirely within a functional safety and reliability framework - there was no threat model, no adversarial testing, and no security boundary between the engineering workstations and the operational control network. Verification had been comprehensive within the domain it covered and completely absent in the domain that mattered.

Stuxnet substantially shaped the development of IEC 62443 and elevated industrial control system security from an academic concern to an operational priority for critical infrastructure operators and their regulators.

6.4 Audit and Assessment Consequences

Beyond the catastrophic product failures described above, the more common consequence of insufficient verification encountered in audit practice is not immediate product failure but delayed or blocked certification and approval. A product that reaches the functional safety assessment or cybersecurity assessment phase with incomplete verification evidence will encounter one of three outcomes: a hold point that delays product release pending remediation, a conditional approval with documented deviations and accepted residual risks, or a negative assessment conclusion requiring rework.

The cost asymmetry here is severe and consistently underestimated in programme planning. A rigorous ASIL D software verification campaign - MC/DC testing, independent code review, FMEA at hardware level, FMEDA with verified diagnostic coverage values, and a complete FSA - represents a defined, plannable cost that can be budgeted at the start of development. The cost of remediating verification gaps discovered during an FSA in the final weeks before a scheduled SOP is typically five to ten times higher, due to rework, re-testing, timeline extensions, and in some cases fundamental design changes driven by assessment findings.

In cybersecurity, the equivalent scenario involves a cybersecurity assessment or UN R155 type approval review discovering an incomplete TARA, missing penetration test coverage, or absent SCA results for third-party

components. Type approval delays have direct commercial consequences in automotive programmes, where Job 1 dates (first saleable product to roll off the production line) are tied to commercial contracts and production capacity commitments. A two-month delay caused by an incomplete cybersecurity evidence package has consequences that extend far beyond the direct cost of the missing verification activity.

6.5 Legal and Regulatory Consequences

The legal consequences of insufficient verification have grown substantially as regulations have embedded verification obligations into product liability and market access frameworks. Under EU product liability law, a manufacturer who cannot demonstrate that required verification activities were performed faces a significantly weakened position in personal injury litigation involving a product defect. The inability to produce verification evidence is not merely a compliance failure - it is an indicator that the manufacturer did not exercise the duty of care that a reasonable and responsible engineering organisation would apply.

Under the CRA, manufacturers of products with digital elements must demonstrate conformity with essential cybersecurity requirements, including the obligation to identify and document vulnerabilities in their products and to perform security testing. The CRA introduces market surveillance mechanisms and can result in product withdrawal from the EU market for non-compliant products [CRA]. Under UN R155, an OEM whose CSMS approval is withdrawn or whose vehicle type approval is challenged due to cybersecurity inadequacies faces both production suspension and potential recall obligations.

The trajectory is clear: verification obligations that were once purely professional engineering standards enforced by industry bodies are becoming legally enforceable product safety and market access requirements. The investment calculus for verification measures must therefore incorporate not only the internal programme cost of discovering defects late, but the external regulatory and liability costs of insufficient verification reaching the market.

7. Future Improvements and Extensions to Verification Measures

7.1 Automation and AI-Assisted Verification

The most significant near-term development in verification practice is the increasing automation of activities that have historically required manual effort. This trend is driven both by the growing complexity of the systems being verified - modern automotive ECUs execute millions of lines of code across multiple operating systems - and by advances in the tools available to verification engineers.

In functional safety software verification, automated test generation tools are becoming increasingly capable of generating test cases that satisfy structural coverage requirements (including MC/DC) without manual test case design. Model-based testing tools can automatically generate tests from formal specifications, providing both coverage evidence and traceability to requirements. These tools do not eliminate the need for skilled verification engineers - test quality still requires human judgement - but they significantly reduce the effort required to achieve required coverage metrics, allowing verification engineers to focus on higher-value activities such as corner case identification and requirements completeness review.

In cybersecurity, AI-assisted vulnerability analysis tools are being applied to SAST, fuzz test case generation, and attack path analysis. Large-scale vulnerability research has demonstrated that machine learning models trained on large codebases can identify certain classes of security vulnerabilities with higher recall than traditional rule-based SAST tools. Automated fuzz testing frameworks with smart mutators - tools that learn from the target's response to previous inputs to generate more effective test cases - are becoming standard practice in product security engineering at leading automotive suppliers.

The verification governance challenge posed by AI-assisted tools is tool qualification. Both ISO 26262 Part 8 and IEC 62443-4-1 impose requirements on tools used in safety and security-relevant processes: the tool must be qualified to demonstrate that its output is trustworthy. Qualification frameworks for AI-based tools are an active area of standards development, with the ISO/IEC JTC 1/SC 42 committee and the EU AI Act's technical standardisation activities both addressing aspects of this challenge.

7.2 Formal Methods at Scale

Formal methods - mathematical techniques for specifying and verifying system behaviour - have long been recognised as providing the strongest possible verification evidence: formal proof that a system satisfies a specification, rather than evidence based on testing or analysis. In practice, their application has been limited by the scalability challenges of applying formal verification to realistic software artefacts of production complexity.

This limitation is diminishing. Advances in model checking, satisfiability modulo theories (SMT) solvers, and abstract interpretation have made formal verification of bounded software artefacts - device drivers, cryptographic protocol implementations, operating system kernels - increasingly practical. The seL4 verified microkernel, deployed in several safety-critical and security-critical systems, demonstrates that formal verification of realistic software components is achievable. The CompCert formally verified C compiler, which guarantees that the compiled executable exactly implements the source code semantics, addresses one of the most significant sources of systematic verification uncertainty in embedded software development: whether the compiler itself introduces errors.

For functional safety, the most promising near-term application of formal methods is property verification on safety-critical software modules - demonstrating formally that a safety monitor, a watchdog implementation, or

a safety shutdown sequence has the specified properties under all possible inputs. For cybersecurity, formal verification of cryptographic protocol implementations and authentication state machines addresses a class of vulnerabilities - protocol logic flaws - that is particularly difficult to detect through testing.

7.3 Continuous Verification in DevSecOps Environments

The automotive and industrial sectors are increasingly adopting continuous integration and continuous deployment (CI/CD) practices, driven by the need to deliver over-the-air (OTA) software updates efficiently and by the adoption of agile development methodologies. These practices create both challenges and opportunities for verification.

The challenge is that traditional verification practices were designed for milestone-based V-model development cycles, where verification activities are conducted at defined phase gates. CI/CD pipelines that produce new software builds multiple times per day cannot accommodate week-long test campaigns at each build. The verification strategy must be adapted to a continuous model where lightweight verification checks run automatically on every build, and more comprehensive verification campaigns are triggered by specific events - a significant design change, a new release candidate, or a newly disclosed vulnerability relevant to the software stack.

The opportunity is that CI/CD pipelines create a natural integration point for automated verification tools: SAST, SCA, and automated regression testing can run on every commit, providing earlier and more granular defect detection than milestone-based testing. The verification evidence generated by these continuous checks accumulates throughout the development cycle and provides a richer basis for the final product assessment than a single end-of-project test campaign.

ISO/SAE 21434 explicitly accommodates this model through its cybersecurity management obligations (Clause 8 and Clause 9), which require ongoing vulnerability monitoring and update processes that are inherently continuous rather than project-milestone-based. The challenge for functional safety is adapting the ASIL-scaled verification obligations of ISO 26262 to a continuous development model - a topic that is actively addressed in industry working groups and that ISO/SAE will need to address in future standard revisions.

7.4 Integration of Functional Safety and Cybersecurity Verification

As engineered systems increasingly face both safety and cybersecurity risks simultaneously, the industry is moving towards integrated safety-security verification programmes that address both risk dimensions in a coordinated way. The case for integration is compelling: a safety-critical system that is cybersecurity-compromised may fail in ways that violate safety goals; and cybersecurity measures that are not verified for their safety implications may themselves introduce safety hazards.

The concept of safety-security co-analysis, in which TARA and HARA are conducted jointly to identify scenarios where security threats translate into safety hazards, is gaining traction in automotive development. ISO/SAE 21434 acknowledges the relationship with ISO 26262 in its normative references, and the standard explicitly addresses the identification of cybersecurity events that can affect safety goals. Practical guidance on how to conduct safety-security co-analysis is provided in the EVITA project findings and in technical documents from bodies such as SAE and the AUTOSAR consortium.

Future verification frameworks will likely formalise this integration more explicitly, providing specific guidance on how verification evidence generated for safety purposes can be used in the cybersecurity case and vice versa, and

on how the confirmation measure structures of the two domains can be harmonised when a single product development project must satisfy both ISO 26262 and ISO/SAE 21434. This integration has not only technical but commercial benefits: an integrated verification programme eliminates duplicated effort, reduces total verification cost, and produces a more coherent risk argument than two parallel programmes that treat safety and security as independent concerns.

7.5 Digital Twins for Verification

The digital twin concept - a continuously updated virtual model of a physical system that reflects its actual state and behaviour - offers significant potential for extending verification capabilities, particularly in the post-production phase. For cybersecurity, a digital twin environment allows security researchers and PSIRT teams to reproduce and analyse newly discovered vulnerabilities against a controlled replica of a production system, enabling the development and verification of patches before they are deployed to the vehicle fleet.

For functional safety, digital twin environments enable regression testing of software updates against a comprehensive suite of previously verified test cases, without requiring access to physical test hardware for every verification run. This is particularly valuable in the context of OTA update processes (ISO 26262 Part 1 and UN R156 [UNR156]), where each software update must be verified before deployment and the verification must be repeatable and traceable.

7.6 Strengthening Verification in the Supply Chain

Modern automotive and industrial products are assembled from software and hardware components sourced from a complex supply chain spanning hundreds of suppliers across multiple tiers. The verification challenge at the OEM level is not simply to verify the final integrated system but to establish with confidence that the verification activities performed by suppliers are adequate for the safety and security integrity levels of the components they provide.

Current practice relies primarily on contractual agreements, supplier audits, and the review of supplier-provided verification evidence packages. This approach is labour-intensive and of variable effectiveness - the quality of a supplier's verification evidence package is a better indicator of that supplier's documentation capability than of the actual rigour of their verification activities. Future improvements in this area are likely to include stronger process certification requirements (IEC 62443-4-1 certification for cybersecurity, expanded use of functional safety management system audits for safety) and the development of standardised evidence exchange formats that allow verification evidence to flow through the supply chain in a machine-readable form, enabling automated consistency checking at the OEM integration level.

The EU CRA's supply chain traceability requirements and the mandatory SBOM obligation provide a regulatory push in this direction for cybersecurity. An equivalent regulatory impulse for functional safety verification evidence exchange is not yet present, but the commercial pressures associated with type approval delays caused by inadequate supplier evidence are driving OEMs to impose progressively more demanding verification evidence requirements on their supply chains.

8. Conclusion: Is the Investment Worth It?

This whitepaper has examined **verification measures across the domains** of functional safety and cybersecurity **from four perspectives**: what the normative frameworks require, how the two domains are similar and how they differ, what happens when verification is bypassed or insufficient, and how verification practice is likely to evolve in the future. The evidence supports a clear answer to the central question: **yes, the investment in verification measures is worth it** - not as a matter of regulatory compliance, but as a matter of engineering economics and organisational risk management.

The **cost comparison is asymmetric in the extreme**. A rigorous verification programme planned and budgeted from the beginning of development represents a defined, manageable cost. The consequences of insufficient verification - late-stage assessment findings, type approval delays, field recalls, litigation, regulatory enforcement actions, and in the worst cases, fatal accidents - are costs that dwarf the original verification budget by factors of ten to one hundred or more. The Toyota recall costs exceeded US\$3 billion. The Boeing 737 MAX programme cost exceeds US\$20 billion. The Fiat Chrysler Jeep Cherokee recall was triggered by the absence of security testing that represents a small fraction of the development budget for a connectivity module.

The **argument for verification investment is** not primarily about compliance - it is **about the irreducible epistemic challenge of complex system development**. Engineers who build complex systems cannot know with confidence, from design alone, whether those systems behave correctly under all relevant conditions and threats. Verification is the process by which this confidence is earned, not asserted. A safety case without verification evidence is a wish, not an argument. A cybersecurity case without penetration testing evidence is an assumption, not a demonstration.

For management audiences, the **most important implication is that verification budgets are risk reduction investments**, not overhead costs. The question is not whether to invest in verification but how to invest most effectively - which verification activities, at which integrity levels, with what degree of independence, and at which points in the development lifecycle provide the best return on the verification investment. The standards provide a structured answer to this question through their integrity-level-scaled method tables and independence requirements. The historical record provides the empirical confirmation.

For technical experts, the most important implication is that **the verification programme must be designed for its purpose** - not for its appearance. Verification activities that are planned but not genuinely executed, that are documented but not traceable to actual work, or that satisfy coverage metrics through superficial test cases rather than through requirement-derived, design-aware test design, provide false assurance. **The independence requirements in the standards exist precisely to create a structural check on this tendency**. An external functional safety assessor who is genuinely independent will find what a project-internal reviewer is structurally unlikely to find.

Looking forward, the technical capabilities available to verification engineers are improving significantly. Automated testing tools, AI-assisted vulnerability analysis, formal methods at practical scale, and integrated safety-security verification frameworks are all making verification more efficient and more effective. The challenge is to adopt these capabilities within the evidence-based argumentation framework that the standards define - not as substitutes for that framework, but as tools for filling it with stronger and more comprehensive evidence.

Verification measures are the engineering profession's most important tool for making the invisible visible: transforming unstated assumptions about system behaviour into examined, tested, analysed, and documented

facts. In safety-critical and security-critical systems, the cost of leaving assumptions unexamined is borne not by the engineers who made them but by the users, operators, and bystanders who depend on those systems to behave correctly.

About the Author

Markus Steinmetz works as an auditor and assessor at TÜV NORD CERT GmbH in the Business Field Specific Management Systems, focusing on IT and information security management systems. In this role he conducts audits and assessments against standards including ISO 27001, ISO/SAE 21434, and IEC 62443.

Independently of this employment, Markus operates as a freelance specialist in the areas of functional safety and cybersecurity, supporting organisations in the automotive and industrial automation sectors. His freelance work covers ISO 26262, ISO/SAE 21434, IEC 62443, and IEC 61508, among other with a particular focus on bridging the gap between functional safety and cybersecurity in integrated development programmes.

The views and content presented in this whitepaper reflect his personal professional expertise and experience and are not associated with or representative of TÜV NORD CERT GmbH or any affiliated entity.

Contact:

- Email: markus@steinmetz-security.de
- Website: <https://steinmetz-security.de>
- LinkedIn: www.linkedin.com/in/markus-steinmetz-78052817

ANNEX

A. Comparison of Verification Method Scaling by Integrity Level

The following table summarises the verification method scaling across both domains, providing a practical reference for project planning.

Verification Method	ASIL A	ASIL B / SIL 1	ASIL C / SIL 2	ASIL D / SIL 3	SIL 4
Requirement review	R	R	R	R	R
Design review	R	R	R	R	R
Code review / walkthrough	R	R	R	R	R
Static analysis	HR	HR	R	R	R
Statement coverage	R	R	R	R	R
Branch coverage	HR	R	R	R	R
MC/DC	-	HR	HR	HR	R
FMEA	R	R	R	R	R
FTA	HR	HR	R	R	R
FMEDA	-	R	R	R	R
Functional safety review	R	R	R	R	R
Functional safety assessment	-	R (dep.)	R (ind.)	R (ind.)	R (ext. ind.)

Table A.1: Indicative verification method requirements by ASIL level (ISO 26262). R = Required, HR = Highly Recommended, dep. = dependent independence acceptable, ind. = independent within project, ext. ind. = independent outside project.

B. Cost-Benefit Reference Data

The following table provides reference data on the cost consequences of insufficient verification, drawn from publicly documented incidents. These figures are illustrative and should not be used as precise estimates; they are intended to calibrate the order-of-magnitude cost asymmetry between prevention and remediation.

Incident	Proximate Cause	Verification Gap	Documented Cost
Therac-25 (1985–87)	Race condition in SW	No V&V plan, no independent review, no race condition testing	3 fatalities; criminal and civil liability
Toyota ETCS (2009–10)	SW systematic faults	Inadequate stack analysis, no worst-case execution time analysis	>US\$3 billion in recalls and settlements
Boeing 737 MAX (2018–19)	MCAS safety assessment gap	Incomplete failure scenario coverage, delegation of assessment	>US\$20 billion; 346 fatalities
Jeep Cherokee (2015)	Uconnect cellular interface	No adversarial security testing on cellular interface	1.4M vehicle recall; first cyber-motivated automotive recall

Table B.1: Reference cost data for verification failures. Sources: [Therac], [NHTSA15], [ACSF], [MV15].

C. Regulatory and Standards Landscape Overview

The following table provides an orientation to the current regulatory and standards landscape relevant to verification measures in functional safety and cybersecurity.

Instrument	Sector	Verification Obligation	Status / Applicability
IEC 61508:2010	General industry	SIL-scaled V&V across hardware and software lifecycle	Voluntary; referenced by sector standards
ISO 26262:2018	Automotive	ASIL-scaled V&V; FSA; confirmation measures	Contractual/type approval basis
ISO/SAE 21434:2021	Automotive	TARA; pen test; cybersecurity assessment	Basis for UN R155 type approval
IEC 62443-4-1	IACS	SDL process including security testing (RQ-08)	Certifiable; widely required in supply chain
UN R155 (2021)	Automotive (type approval)	CSMS type approval; cybersecurity verification evidence	Mandatory from July 2024 (new types, EU/Japan/Korea)
EU CRA (2024)	Products with digital elements	Security testing, vulnerability management, SBOM	Mandatory - transitional periods apply
EU NIS2 (2022)	Critical infrastructure operators	Security risk management including verification	Mandatory - transposed from Oct 2024

Table C.1: Regulatory and standards landscape for verification measures in functional safety and cybersecurity.

===== END OF DOCUMENT =====