



Why Robot Safety Depends on Cybersecurity

*Protecting the information
that safety decisions rely on*



Learn more about VicOne
by visiting **VicOne.com**
or scanning this QR code:



Preface

A robot receives a command. The command is authorized. The software that carries it out is genuine, unmodified, running exactly as its developers intended. The robot executes the command precisely as instructed. And the outcome is unsafe.

This is not a contradiction. Everything upstream of the robot's action can be correct — the authorization, the code, the hardware, the operator's intent — and the robot can still cause harm if the information its safety decisions depend on has been manipulated before it reaches the decision.

The thesis of this paper is straightforward: many safety mechanisms in modern robots ultimately depend on trusted information they do not independently verify. Cybersecurity determines whether an adversary can deliberately manipulate that information, and in doing so, undermine the safety guarantees built around it.

In 2025, VicOne LAB R7 published *Securing the Rise of AI Robots*,¹ mapping the attack surface of AI-powered robots across five layers and cataloging techniques that target each one. That paper addressed a foundational question: **What can be attacked on an AI robot?**

This paper takes the analysis one step further: **What happens to a robot's safety mechanisms when one of those attacks succeeds?**

The distinction matters because authorization, software integrity, and functional safety address different properties. A command may come from an approved source. The software executing it may be genuine and unmodified. Yet the resulting physical action may still be unsafe if the robot is reasoning from falsified sensor data, corrupted state information, manipulated model output, or another compromised safety-relevant input.

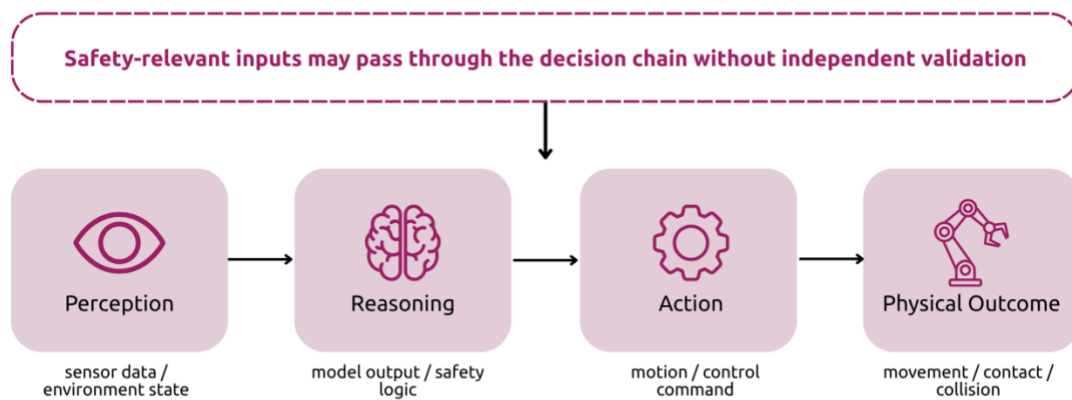


Figure 1. Safety decisions can depend on information that is not independently validated before it influences physical action.

This paper does not argue that AI robots are too dangerous to deploy or that functional safety engineering is obsolete. Functional safety provides rigorous methods for addressing random hardware failures and foreseeable component faults, and those methods remain valid for the failure model they were designed to address.

Cybersecurity introduces a different condition: an intelligent adversary that can deliberately create or reproduce states a safety analysis might otherwise treat as unlikely. The issue is therefore not cybersecurity instead of functional safety, but cybersecurity as a necessary part of maintaining safety assumptions in an adversarial environment.

The argument develops across seven chapters:

- **Chapter 1** distinguishes authorization and correctness from physical safety.
- **Chapter 2** identifies the safety mechanisms modern robots rely on and the trust anchors beneath them.
- **Chapter 3** examines how adversarial manipulation changes the assumptions behind conventional functional safety probability models.
- **Chapter 4** applies the framework to specific cases and failure chains.
- **Chapters 5 and 6** turn from analysis to practice, covering adversarial validation and continuous assurance after deployment.
- **Chapter 7** provides recommendations for key stakeholders, and the conclusion draws the argument together.

The discussion is intended for safety engineers, cybersecurity practitioners, systems integrators, and decision-makers responsible for deploying or governing robotic systems. Each approaches the problem from a different discipline, but all ultimately confront the same question:

If the information a safety mechanism relies on can be manipulated, how much confidence can we place in the safety guarantee built on top of it?

Scope and Limitations

This paper presents a structural framework for examining how cybersecurity weaknesses can affect robotic safety mechanisms. It does not constitute a platform-specific safety or cybersecurity assessment.

Several caveats are important when interpreting the analysis:

- **Evidence status.** The cases in Chapter 4 are labeled according to the strength of the available evidence. Some represent demonstrated manipulations, while others are composed chains or conceptual scenarios rather than fully demonstrated end-to-end attacks. At the time of writing, no publicly attributed incident documents a complete causal chain from cyber manipulation to physical safety outcome. This absence should not be read as evidence that the risk does not exist; it also reflects the difficulty of reconstructing such a chain when sufficient decision-provenance data are unavailable. Section 6.5 addresses this evidence gap and also informs the paper's recommendations to regulators.
- **Generality.** Table 1 and the human-in-the-loop trust anchor discussed in Chapter 4 describe mechanisms common across industrial, mobile, service, and humanoid robots, but they are not exhaustive. A specific manufacturer's architecture may implement a listed mechanism through a materially different technical approach than the representative implementation described in Section 2.3.
- **Causation and exploitability.** Demonstrating that a named trust anchor can be manipulated does not establish that a specific deployed robot has a practical attack path from an external adversary to a physical safety outcome. Establishing that requires the validation methodology described in Chapter 5, applied to the platform's actual trust anchors, control software, interfaces, and operating environment.
- **Worked example.** The warehouse autonomous mobile robot (AMR) in Section 6.6 is a composite created to illustrate how the paper's framework fits together across chapters. It is not an account of an actual incident or vendor platform. Its value is methodological, not evidentiary in the sense that Chapter 4's individually cited cases are.

Accordingly, this paper should be read as a framework for evaluating the assumptions behind a robot's safety case, not as a completed audit of any specific system. Applying the framework to a deployed platform requires the same trust-anchor identification, reachability analysis, and software-in-the-loop testing this paper describes, carried out by teams with access to that platform's actual control software, sensor suite, and operating environment.

Chapter 1. The Authorization Fallacy and the Moving Safety Boundary

1.1 A command that was right, and an outcome that was wrong

Consider a mobile robot given an authorized instruction: step to the left. The command comes from an approved source, the software is unmodified, and the robot executes exactly as instructed. However, if that step carries the robot off the edge of a loading dock, the outcome is unsafe even though the command and execution are valid.²

The scenario, raised in an industry dialogue on robot safety, illustrates a distinction central to this paper: **authorization establishes whether an action is permitted; it does not establish whether that action is safe in the physical context in which it occurs.**

- **Authorization:** Did the command come from a permitted source?
- **Integrity:** Were the command and software unmodified?
- **Safety:** Can the resulting physical action be executed without causing harm under the current conditions?

Authentication, digital signatures, and encryption can help establish the first two. They do not, by themselves, determine whether the information a robot uses to assess its surroundings accurately reflects the physical environment.

This opens another path to an unsafe outcome. An attacker may not need to take control of the robot or alter its software. Manipulating a safety-relevant input — through sensor spoofing, prompt injection, adversarial perturbation, or another compromised information path^{3,14} — may be enough to cause correctly functioning software to act on an incorrect representation of its surroundings.

1.2 Safety is a moving boundary, not a fixed perimeter

Traditional industrial robot safety often relied on separating people and machines through fixed controls such as fences, cages, or light curtains. That model is effective when the operating boundary is static.

Mobile robots, service robots, and humanoids operate differently. They move through shared environments, interact with people, and encounter conditions that can change from one moment to the next. As a result, the same action can be safe in one context and hazardous in another. For example, a movement that is routine in an open warehouse aisle may become unsafe near a stairwell or loading edge.

Determining which condition applies depends on the robot's perception of its surroundings. Sensors, models, localization systems, and communications provide the information that allows control logic to judge whether an action remains within safe operating conditions.

1.3 Why this reframes the cybersecurity conversation for robots

Cybersecurity incidents involving robots can extend beyond data, system availability, or digital integrity into the physical environment. When a compromised input or decision leads to a fall, collision, or unintended application of force, the consequence cannot simply be rolled back after the fact.

This physical dimension changes the role of cybersecurity in robotic systems. Protecting software, communications, models, and sensor inputs not only preserves system integrity; it also preserves the assumptions on which physical safety mechanisms depend.

For robots, cybersecurity and functional safety therefore intersect where digital information influences physical action.

1.4 A reusable way to ask the question

Rather than examining vulnerabilities in isolation, this paper evaluates them in relation to the safety mechanism they may affect. For each mechanism, four questions guide the analysis:

1. What does the mechanism trust?
2. Can an attacker manipulate that trust anchor?
3. If the trust anchor is manipulated, what physical behavior changes?
4. Does that changed behavior invalidate a safety claim?

Applied to the loading-dock scenario in Section 1.1, the relevant mechanism is the obstacle- or edge-detection logic expected to stop the robot before it leaves the platform. Its trust anchor is the depth or occupancy estimate the logic relies on.

If that estimate is manipulated — for example, through a reflective surface affecting a depth camera, a LiDAR dropout,^{4,5} or a stale cached map that no longer reflects the environment — the robot may incorrectly determine that stable ground remains ahead. The safety logic may still execute as designed; what has changed is the information it depends on.

The resulting chain is straightforward:

**Authorized Command → Correct Software → Correct Execution →
Trusted Safety-Relevant Information? → NO → Unsafe Physical Outcome**

Chapter 2 catalogs the safety mechanisms and trust anchors to which these questions apply; Chapters 4 and 5 then apply them in case analysis and validation.

Chapter 2. The Trust Anchors Behind Robot Safety

2.1 Many safety mechanisms depend on trusted inputs

A robot's safety depends on a set of engineered mechanisms, many of which rely on software, sensors, firmware, communications, or computed state. These mechanisms often depend on information about the robot and its environment.

That dependence introduces a cybersecurity consideration into the safety case. A safety mechanism may function exactly as designed while acting on information that has been manipulated upstream. In such cases, the weakness may lie not in the safety logic itself, but in the trustworthiness of the input or state on which that logic depends.

These dependencies connect robotic safety to the five-layer attack surface mapped in VicOne LAB R7's 2025 report: physical, perception, AI model, wireless communication, software/cloud.¹ A safety-relevant input may be exposed through one or more of these layers.

This does not apply equally to every safety mechanism. Physical emergency stops, mechanical brakes, and independent hardware interlocks can provide protection without relying on the same software-mediated decision chain. Many modern robotic safety mechanisms, however, depend on software, sensing, communications, or computed state, making the integrity of those inputs relevant to the safety assumptions they support.

2.2 The mechanism catalog

The trust-anchor problem becomes clearer when examined mechanism by mechanism. Table 1 maps common robotic safety mechanisms to the software, sensing, or control functions that implement them; the information or state they depend on; potential manipulation paths; and the resulting safety consequence.

The table is not intended to suggest that each mechanism has a single implementation or attack path. Rather, it provides a representative view of where cybersecurity dependencies can enter the safety case.

Safety mechanism	Implemented by	Trust anchor	Manipulation path	Potential consequence
Emergency stop/power cutoff	Safety PLC, fieldbus, motor firmware	Stop signal/control path; anomaly detector	Suppressed or forged bus message ^{6,7}	Delayed or absent stop
Torque/force limiting	Joint controller firmware, torque sensing	Torque/joint-state reading	Injected sensor/bus values ⁸⁻¹¹	Excessive contact force

Safe speed monitoring	Perception stack + motion controller	Human-proximity estimate	Perception manipulation, instruction injection ^{12–15}	Unsafe speed near people
Geofencing	Localization stack + path planner	Position estimate (GPS/UWB/SLAM)	GNSS spoofing, pose drift ^{16–18}	Boundary violation
Obstacle avoidance	Perception model + planner	Object detection/occupancy	Adversarial patch, LiDAR spoofing ^{19–22}	Collision or unnecessary stop
Safe state/degraded mode	Anomaly detector + state machine	Same sensor inputs as detection logic	Baseline poisoning ^{60,61}	Failure to enter safe state (omission)
Sensor cross-validation	Fusion/voting logic	Independence of cross-checked sensors	Coordinated multi-sensor manipulation ^{23–25}	False confidence in a phantom or a miss
Watchdog/liveness monitoring	Timer + recovery logic	Process responsiveness only	N/A — defends a different failure mode	Does not catch a correctly running process acting on false input

Table 1. Safety mechanisms and their trust anchors

The mechanisms in Table 1 can be read more clearly at the level of the safety function each one performs and the trust anchor it depends on:

- **Emergency stop** halts motion or cuts power when a hazard is detected. A safety PLC or supervisory process asserts a stop signal to the motor controllers, so the mechanism depends on that signal reaching its destination and on the logic that determines when it should be triggered.
- **Torque and force limiting** keeps contact force below an injury threshold. Current-sensing or torque sensors feed a control loop that caps commanded torque, making the resulting torque, current, or joint-state estimate a key trust anchor.
- **Safe speed monitoring** adjusts permitted speed based on proximity to people. A perception stack estimates human distance, and a motion controller uses that estimate before issuing velocity commands.
- **Geofencing** restricts the robot to a defined operating area. A localization stack determines position, and a path planner checks that estimate against a stored boundary.
- **Obstacle avoidance** prevents collision by relying on object-detection or occupancy information produced by the perception stack and passed to the motion planner.
- **Safe-state and degraded-operation logic** shifts the robot to reduced capability when a condition cannot be resolved with confidence. Its effectiveness depends on both the inputs of an anomaly or fault detector and the detector's definition of what constitutes "normal."

- **Sensor cross-validation** is intended to detect a faulty or spoofed sensor by comparing independent channels. Its effectiveness depends on those channels remaining genuinely independent, an assumption a coordinated attack may be designed to undermine.
- **Watchdog and liveness monitoring** detect processes that stop responding. They may not detect a process that continues to operate normally while acting on manipulated input, illustrating the difference between a malfunctioning process and a correctly functioning process working from an incorrect premise.

Across these mechanisms, the recurring issue is not necessarily a failure in the control logic itself, but the trustworthiness of the information that logic relies on. Table 1 makes that dependency visible and provides the unit of analysis used throughout the rest of the paper.

2.3 How trust anchors are implemented in practice

Table 1 identifies each mechanism's trust anchor at the functional level. The underlying implementation matters because the protocol, component, or estimator involved can determine how that trust anchor can be reached or manipulated in practice.

For **emergency-stop systems**, the stop signal may traverse a safety-rated fieldbus segment, alongside or instead of a hardwired safety relay. Robotics middleware configurations may also operate without message-layer authentication enabled by default,^{26,59} which creates a distinction between the safety characteristics of the control path and the authenticity of the messages carried over it.

For **torque and force limiting**, collaborative robots may estimate torque from motor current combined with a kinematic or dynamic model, processed through an observer or Kalman-style estimator, rather than rely on a dedicated torque sensor at every joint. In these implementations, the trust anchor is a computed quantity rather than a single sensor reading. A manipulated but plausible current value introduced into the estimator can affect the resulting torque estimate without necessarily appearing as an obvious outlier, particularly when the estimator is tuned to reject abrupt deviations rather than values that resemble normal current variation.^{11,27}

The human-proximity estimate behind **safe-speed monitoring** may come from a sensor-fusion pipeline combining inputs such as depth cameras, 2D LiDAR, and ultra-wideband (UWB) tags. Each of these input channels has documented cybersecurity weaknesses.^{21,22,29}

Similar dependencies appear in **geofencing and localization**. Outdoor systems may combine GNSS with inertial measurements through an extended Kalman filter, while indoor systems may rely on SLAM-derived pose. GNSS spoofing and SLAM pose drift can stay within a filter's innovation-gating threshold, making them harder to distinguish from normal variation than abrupt changes.^{11,16,18}

For **obstacle avoidance and sensor cross-validation**, fusion architectures can combine sensor information at different stages: early fusion combines raw data before detection, late fusion

combines separate detections, and hybrid approaches combine intermediate features.^{25,30} These architectural differences affect how coordinated manipulation can propagate through the system. A shared intermediate representation, for example, may allow one perturbation to influence a fused output, whereas a late-fusion design may require separate manipulation of multiple detection pipelines.

2.4 Reachability across the five-layer attack surface

While Table 1 names the trust anchor behind each safety mechanism, it does not show how that trust anchor may be reached or manipulated.

VicOne LAB R7's 2025 report organizes these paths across five attack-surface layers:¹

- **Physical:** exposed debug ports, unsigned firmware
- **Perception:** cameras, microphones, radar and other sensing channels susceptible to spoofing or adversarial input
- **AI model:** adversarial examples, data poisoning, model extraction
- **Wireless communication:** weakly authenticated traffic and manipulation of commands or over-the-air (OTA) updates
- **Software/cloud:** middleware, API, and identity and access management weaknesses.

A single trust anchor may be exposed through more than one of these layers. A torque estimate, for example, may be manipulated through a compromised sensor interface, a modified middleware message, or a corrupted state estimator.^{8–11,26,59}

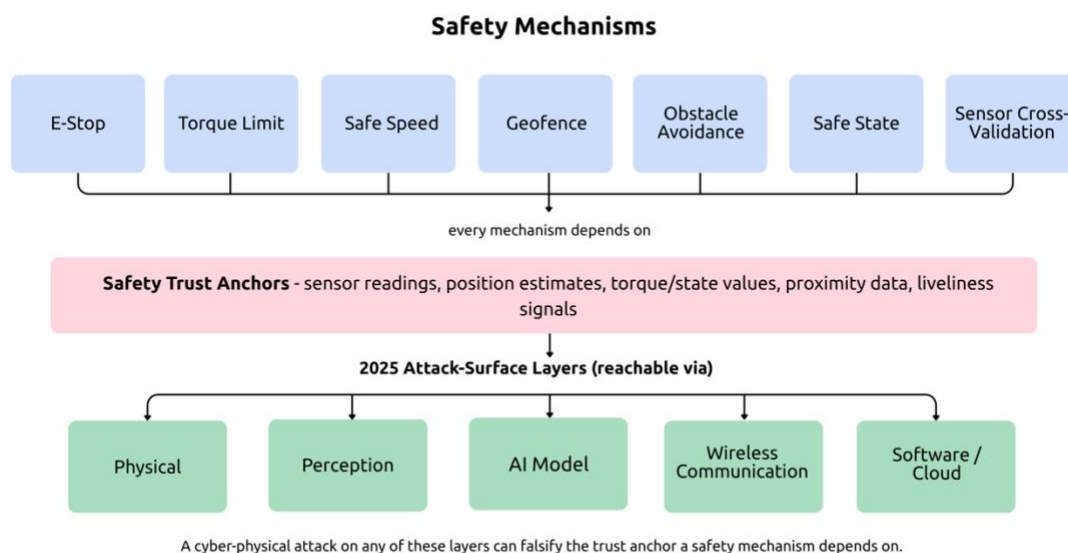


Figure 2. Safety mechanisms depend on trust anchors that may be reachable through multiple layers of the attack surface mapped in VicOne LAB R7's 2025 report.

This has direct implications for defensive prioritization. Securing one layer may leave other paths to the same trust anchor unaddressed. For example, a mitigation focused on the software/cloud layer may not address manipulation originating at the physical, perception, or AI-model layer.

Table 1 and Figure 2 should therefore be read together: Table 1 identifies what each safety mechanism depends on, while Figure 2 shows where those dependencies may be exposed. Together, they provide a practical way to assess coverage across both safety mechanisms and attack-surface layers.

2.5 A common pattern across these mechanisms

Across the mechanisms described above, a common chain emerges: **perception** converts the physical environment into digital observations; **reasoning** interprets those observations through models and control logic; **action** translates the results into commands; and those commands produce a **physical outcome**.

These mechanisms therefore depend on the integrity of information moving through that chain. A torque limiter, for example, may execute exactly as designed while acting on a falsified current reading. The control logic may not have failed; the premise it acts on has changed.

AI-enabled robotic systems can add another layer of complexity. Large language models, vision-language models, and vision-language-action models may combine an operator instruction, text captured by a camera, transcribed audio, or retrieved content within the same reasoning context.^{3,14,15} When the system treats that content as relevant to an executable action, information entering through a perception or content channel may influence behavior without crossing a conventional authentication boundary. Perception models can also be misled by small, deliberately crafted input perturbations (adversarial examples), which change a model's output without any change to the software that runs it.^{57,58}

Content filtering can reduce some of this risk, but it does not by itself establish the origin or trustworthiness of every input. Semantically equivalent content, context-dependent instructions, or legitimate information presented in an unsafe situation may still affect downstream reasoning and action.

2.6 Trust dependencies between models

A single robotic task may involve multiple AI models operating in sequence — a vision-language model interpreting a scene, a language model translating that interpretation into a task, a planner decomposing the task into subgoals, and a low-level policy model converting subgoals into motor commands.

Each stage may introduce its own trust dependency, including the interfaces through which one model's output becomes another model's input. These interfaces may not be visible when the system is assessed only as a collection of individual components.

Changing what a vision-language model says about a scene can change downstream behavior without requiring changes to the downstream software itself. Individual components may continue to process well-formed inputs exactly as designed while the information passed

between them carries an incorrect or manipulated interpretation. What has changed is the meaning carried between models, and cryptographic integrity alone does not establish that this meaning is correct or safe in context.

This has implications for vulnerability management. Conventional frameworks are typically organized around a nameable, versioned component and an applicable patch — the structure a CVE record assumes. A weakness that emerges from the interaction between two models, rather than from either component in isolation, may not map cleanly to that structure or to a single corrective action.

Two related patterns illustrate the issue. **Model chaining** describes how an incorrect or manipulated output can propagate through a multi-model pipeline even when no individual component contains a conventional software vulnerability. The weakness may instead arise from how decisions and representations are passed between components, making it difficult to detect through component-level vulnerability scanning alone.

Emergent vulnerability describes a failure that appears only after models or components are integrated, even when each behaves correctly in isolation. Such behavior may go undetected when validation considers individual models without also testing the composed system. This is why this paper treats trust dependencies, including those between models, as an important unit of safety analysis alongside the individual model.

Chapter 3. Turning the Unlikely into the Repeatable

3.1 How functional safety handles rare events

Functional safety engineering does not promise that hazardous events will never occur. Instead, it seeks to reduce risk to an acceptable level within defined operating conditions and failure assumptions.

For random hardware failures, this can include probabilistic assumptions that faults occur independently and without deliberate intent — for example, a sensor drifting out of calibration or a bearing wearing unevenly. When those assumptions hold, treating such events probabilistically fits the failure model being analyzed.³¹

3.2 The premise cybersecurity breaks

An adversary is not a random fault. A condition that would occur once in ten million operating hours by chance no longer remains purely probabilistic once an attacker can deliberately reproduce the conditions that cause it.

As one perspective from the industry dialogue puts it, an attacker's objective is to turn a low-probability event into a repeatable condition.²

Functional safety probability models remain valid for the failure assumptions they were designed to represent. Cybersecurity does not replace that analysis — it determines whether those assumptions remain valid in the presence of an adversary.

It is worth being specific about which assumption can break. A hazard-rate calculation for a random fault may depend on components failing independently of one another. The probability of two sensors failing at the same moment and in the same direction can therefore be treated differently from the probability of either failing alone.

An adversary, however, may deliberately target multiple channels with the same manipulation. A single crafted physical object, for example, may affect two sensing channels that would otherwise be treated as independent under a random-failure model (Section 4.7). In that case, the independence assumption supporting the redundancy calculation may no longer hold.

The mathematics of functional safety is not wrong; the issue is whether the assumptions behind the calculation still match the conditions being analyzed.

A related consequence is that a hazardous condition considered tolerable partly because it is unlikely to coincide with the moment of greatest consequence may require reassessment if an adversary can deliberately influence when that condition occurs.

This reframes the question a safety case must answer: not only "How likely is this hazardous condition under the assumed failure model?" but also "Can an adversary deliberately create it?"

3.3 A vulnerability does not, by itself, answer the safety question

Identifying a vulnerability is only part of the analysis needed to assess safety impact. It shows that a condition may be deliberately created; it does not show what the robot will actually do once that condition exists — stop safely, continue in a degraded but tolerable way, or cross a boundary it was never validated to cross.

Answering that requires separating three questions:

1. Is the vulnerability reachable?
2. Can it be exploited?
3. If exploited, which specific part of the robot's behavior changes?

Reachability and exploitability establish whether a manipulation path is feasible, but they do not determine the robot's physical response. That response depends on the interaction between the manipulated condition, the control logic, and the robot's physical dynamics.

Chapter 4 applies this distinction to specific safety mechanisms and manipulation scenarios.

3.4 Deployment does not freeze validated conditions

AI robots can continue to change after deployment through software updates, new environments, and fleet-learning pipelines. As these conditions change, assumptions established during design-time validation may no longer hold. An adversary may also seek to exploit changes or gaps that emerge over time.

The probability estimates a safety case relies on are not necessarily fixed for the system's lifetime; they depend on assumptions about operating conditions that may change after deployment.

Chapter 6 examines how continuous verification can help determine whether those assumptions continue to hold over time.

3.5 Why safety margins do not substitute for adversarial analysis

A natural objection is that safety margins already absorb some uncertainty: if a mechanism is engineered with additional tolerance beyond its nominal operating requirement, does the source of a rare triggering condition — random or adversarial — actually matter?

It does, because a safety margin does not necessarily account for deliberate manipulation. Safety margins may be designed around expected variation, such as how far a sensor might drift or how much a component's tolerance may vary. An adversary, however, may deliberately create conditions outside those expected ranges, or strategically within them.

If an attacker understands the assumptions or thresholds behind a safety mechanism, they may use that knowledge to shape a manipulation around them. A margin designed to absorb

expected variation therefore does not establish that the mechanism will remain effective against deliberate manipulation.

This is also relevant to ALARP (“as low as reasonably practicable”)³² and similar risk arguments. If adversarial manipulation introduces credible conditions that were not considered in the original risk assessment, those conditions may need to be incorporated into the validation process described in Chapter 5.

Chapter 4. When Safety Logic Trusts the Wrong Input: Fourteen Cases

4.1 How to read the cases

The cases below follow a common pattern: rather than disabling a safety mechanism directly, the manipulation changes the information or condition on which that mechanism relies. The resulting behavior can then remain consistent with the system's logic while producing an unsafe or unintended outcome.

Each case is examined using the same analytical structure: **Perception** (what is manipulated) → **Reasoning** (how it is interpreted or trusted) → **Action** (the resulting command or response) → Physical Outcome.

4.2 Safe-speed monitoring: inaudible instruction injection

Perception: An inaudible ultrasonic tone, demodulated by a microphone's nonlinear front end, is transcribed by the robot's speech-recognition module as a plausible instruction overriding the human-proximity limit.^{12–15,26,59}

Reasoning: The transcript enters the language-model context without a reliable indication of its origin, allowing the planner to interpret it as an authorized instruction.

Action: The resulting velocity command is accepted if the downstream control path does not independently verify the instruction's source or enforce the relevant safety constraint.

Physical Outcome: The robot operates at full speed inside a zone its safe-speed logic should have restricted, potentially without an obvious fault in the control logic itself.

The individual primitives here — inaudible audio injection and instruction hijacking overriding a stated constraint — are each independently demonstrated. The complete end-to-end chain on a physical robot remains a composed scenario rather than a single published demonstration.

4.3 Torque and force limiting: forged body-state input

Perception: Forged joint-position or torque values are injected on an internal bus, or raw body-state sensors are manipulated electromagnetically.^{6–11,33}

Reasoning: The impedance-control loop computes torque from the corrupted state. Gradual manipulation may remain within estimators tuned to reject abrupt outliers rather than systematic drift.

Action: Actuators apply a torque setpoint computed from the manipulated input.

Physical Outcome: Depending on the system and operating context, the resulting behavior could include loss of balance, a dropped payload, or excessive contact force against a person. In this case, the safety mechanism is affected by manipulating the state information it relies on rather than disabling it directly.

4.4 Geofencing: gradual position drift

Perception: GNSS spoofing or a forged pose message displaces the robot's estimated position along an attacker-chosen path.^{11,16–18}

Reasoning: Gradual, systematic bias may remain within outlier-rejection logic tuned primarily for abrupt deviations.

Action: The path-following controller tracks the manipulated position estimate.

Physical Outcome: The robot may cross its intended boundary into a restricted or hazardous area while individual checks continue to see inputs that appear internally plausible.

4.5 Safe state: poisoning the sensor baseline

Perception: A chemical sensor's calibration baseline is gradually poisoned until "normal" drifts to a level the attacker chose.^{60,61}

Reasoning: The safety logic interprets readings below the configured threshold as safe. Because the manipulated baseline can keep reported values within the expected range, the condition may not trigger the mechanism's normal anomaly criteria.

Action: The system may therefore fail to trigger a shutdown, alarm, or evacuation response.

Physical Outcome: Personnel may remain in a hazardous environment the safety system interprets as normal. This case differs from the preceding examples because the unsafe outcome results from a missing safety response rather than an incorrect physical command. Detection therefore depends on controls that can identify drift or inconsistency independently of the same perception chain, rather than only monitoring for anomalous robot actions.

4.6 Obstacle avoidance: adversarial patches and LiDAR spoofing

Perception: A physical adversarial patch — a printed pattern applied to a wall, floor, or object — causes a vision model to misclassify an obstacle or report a false clear path. A second, independently documented technique achieves a similar result against LiDAR by injecting a spoofed return timed to the sensor's own pulse pattern.^{4,5,19–22}

Reasoning: The manipulated perception output is passed to the motion planner. Without an independent cross-check against a stored map or another modality, the planner may treat that output as a valid representation of the environment.

Action: The controller may issue a continue-at-speed command into an occupied space, or stop unnecessarily for a phantom obstacle.

Physical Outcome: The resulting behavior could include collision or unnecessary operational disruption. Physical adversarial patches that affect vision classification have been demonstrated; their effect on a specific robot's motion planner remains a composed scenario unless demonstrated end-to-end on that platform.

4.7 Sensor cross-validation: when both channels are fooled together

Perception: A single physical object — specifically shaped and textured — is constructed to evade both a camera and a LiDAR simultaneously rather than attacking either sensor independently. Published research has demonstrated attacks in which both modalities miss the same real object or agree on the presence of an object that is not there.^{23,24}

Reasoning: The fusion pipeline relies on agreement between independent channels as a trust criterion. If both channels are manipulated coherently, an inconsistency detector designed primarily to identify disagreement between sensors may have nothing to flag.

Action: The planner may fail to brake for a real object missed by both sensors, or brake unnecessarily for a fabricated one.

Physical Outcome: The resulting behavior could include collision or unnecessary braking despite agreement between the sensor channels. This case shows the limit of redundancy as a defense: cross-validation is effective when failures or manipulations remain independent, but its assurance value can weaken when multiple channels are influenced in a coordinated way.

4.8 Human-in-the-loop approval: false touch injection

Perception: Electromagnetic injection induces a false touch event on a capacitive touchscreen used for operator confirmation of a high-risk action or as a software-based stop control, without physical contact from an operator.^{34,35}

Reasoning: If the interface logic does not independently distinguish a genuine touch from an electromagnetically induced event, the system may process the input as valid human confirmation.

Action: An action gated behind that confirmation, such as releasing an interlock, approving a high-risk motion, or silencing an alarm, may proceed as though an operator had reviewed and approved it.

Physical Outcome: A safety-relevant action that requires human confirmation may proceed without genuine operator approval. Human oversight can also depend on information produced by the same perception and reasoning chain described in Chapter 2. If that information is manipulated, the approval step may inherit the same trust dependency. Human approval should therefore be treated as a safety-relevant input that also requires protection, rather than as an independent backstop by default.

4.9 Proximity estimation: radar and UWB ranging spoofing

Perception: A black-box attack framework can estimate aspects of a radar's internal configuration from observed transmissions and inject fabricated reflections into its point cloud: adding objects that do not exist, deleting objects that do, or shifting a detected object's apparent range and velocity. A related attack against ultra-wideband ranging exploits the correlation logic used to estimate distance, causing a distant device to be reported as closer than it is.^{29,36}

Reasoning: Downstream safety logic may accept manipulated radar or ranging output as valid when it has no independent cross-check against another modality.

Action: A safe-speed controller may relax its limit because a person is reported farther away than they are, or a geofence may accept a falsified proximity reading.

Physical Outcome: The resulting behavior could include unsafe operating speed near an underestimated hazard or an access-control failure based on a spoofed range measurement. Published research has demonstrated both the radar point-cloud attack and the ranging correlation attack against commercial off-the-shelf hardware.

4.10 Balance and attitude control: acoustic resonance on inertial sensors

Perception: Acoustic energy tuned to a MEMS sensor's mechanical resonant frequency induces false accelerometer or gyroscope readings on platforms whose balance or attitude control depends on them. Coordinated multi-axis injection can also reduce the effectiveness of consistency checks designed to identify a single faulty channel because the injected error is not confined to one axis.^{37–40}

Reasoning: The attitude estimator and balance or flight controller may incorporate the manipulated inertial data as though it were genuine.

Action: Stabilization logic may issue motor commands in response to the false attitude being reported.

Physical Outcome: Depending on the platform and control architecture, the resulting behavior could include loss of balance in a legged robot or loss of attitude control in an aerial platform. If the manipulated sensor readings remain internally consistent, the condition may also avoid triggering safety logic that relies on detecting sensor disagreement or implausible state changes.

4.11 Safety parameters: tampered OTA update

Perception: If an over-the-air update channel lacks end-to-end signature verification or equivalent integrity protection, an attacker along the update path may intercept an update request and substitute a modified package.^{7,26,41,42}

Reasoning: If the update client does not independently verify the package's authenticity and integrity, it may accept a well-formed substituted package as a legitimate release.

Action: The modified package may be installed with the privileges available to a legitimate update, potentially changing a torque limit, safe-speed threshold, geofence boundary, or other safety-relevant parameter.

Physical Outcome: The safety mechanism may then continue operating under a policy or threshold that differs from the previously validated one. This differs from manipulating a single real-time reading: the update can change the mechanism's underlying policy and persist across subsequent operating cycles until detection or correction.

4.12 Safety limits: malicious skill module

Perception: A downloadable "skill" package obtained from an open or loosely vetted repository contains logic that changes a safety-relevant parameter alongside its advertised functionality — for example, a "faster picking" skill that also relaxes the safe-speed proximity threshold to achieve its performance gain.^{43,44}

Reasoning: If the skill is installed without sufficient code signing, permission controls, or behavioral review, it may modify safety-relevant settings available to the robot's normal software environment.

Action: The robot may operate under the altered threshold whenever the skill is invoked.

Physical Outcome: The resulting behavior could include an unsafe operating speed or force limit introduced through installation rather than a real-time sensor attack. In this scenario, the user seeking a performance improvement becomes the delivery path for the unsafe change. The complete robot-specific chain has not been demonstrated end-to-end; the scenario draws on documented precedent from malicious or insufficiently vetted software-module ecosystems.

4.13 Sensor fusion: single-channel manipulation of the conflict-resolution rule

Perception: A camera reports a person present in the robot's path while a LiDAR return, degraded by a reflective or absorptive surface, reports the path as clear. An attacker who can influence only one of the two channels can therefore create a disagreement without defeating both sensors.^{24,25,45}

Reasoning: If the fusion architecture resolves conflicting inputs through a fixed priority or confidence rule that does not account for deliberate manipulation, the adjudication process may favor the compromised channel.

Action: The planner may act on whichever signal the adjudication rule gives greater weight to.

Physical Outcome: If the favored signal is the manipulated one, the robot may proceed as though the path were clear when a person is present. This is a narrower, single-channel variant of the coordinated attack described in Section 4.7 and may require influence over only one sensor rather than multiple channels.

4.14 Multi-robot collision avoidance: time-reference spoofing

Perception: Multi-robot systems may rely on a shared time reference — often GPS-derived or distributed over the network — to coordinate collision avoidance among robots operating in the same space. Spoofing that reference for one robot in the fleet can desynchronize its estimate of where its peers are relative to their broadcast trajectories.^{46,47}

Reasoning: If the coordination logic assumes a consistent time base across the fleet without independently verifying that assumption, the affected robot may interpret otherwise valid trajectory information against an incorrect timestamp.

Action: The robot may plan a trajectory using a stale or mistimed estimate of a peer's position.

Physical Outcome: The resulting behavior could include a near-miss or collision even when each robot is individually executing collision-avoidance logic based on the information available to it. This case extends the trust-dependency concept developed in this paper beyond single-robot mechanisms to the coordination assumptions shared across a fleet — a category not explicitly enumerated in Table 1 because it belongs to the multi-robot system rather than to one robot's internal safety mechanism.

4.15 Human-presence detection: thermal spoofing

Perception: Thermal or infrared imaging used for human presence detection can be influenced by physical conditions that alter the apparent thermal signature. A targeted heat source or heated decoy may create a false-positive detection, while thermal masking or insulation can reduce a person's apparent signature below a detection threshold.^{48,49}

Reasoning: If the presence-detection logic relies primarily on the thermal signal without an independent cross-check, a manipulated reading may be interpreted as valid evidence that a person is present or absent.

Action: A false positive may cause the robot to enter an unnecessary reduced-speed or safe-state mode. A false negative may prevent the system from reducing speed or stopping for a person whose thermal signature is not detected.

Physical Outcome: The resulting behavior could include unnecessary operational disruption or, more consequentially, unsafe operation near an undetected person. The cited research demonstrates manipulation of thermal pedestrian detection; the resulting robot-safety chain remains a composed scenario unless demonstrated end-to-end on the specific platform. This case also shows that adding another sensing modality does not eliminate trust dependencies: the additional input introduces another set of assumptions that adversarial validation may need to examine.

4.16 Evidence status: what is demonstrated, what is composed

The cases in this chapter do not all carry the same level of empirical support. Some attack primitives have been demonstrated directly on relevant hardware or sensing systems. At the same time, other physical outcomes are composed from separately demonstrated steps or remain conceptual scenarios grounded in documented precedent.

Table 2 summarizes the evidence status of each case and distinguishes the demonstrated attack primitive from the complete physical-outcome chain.

Case	Table 1 mechanism	Attack primitive	Evidence status	Potential physical outcome
Safe speed monitoring	Safe speed monitoring	Ultrasonic instruction injection	Primitives demonstrated; end-to-end robot chain composed	Unsafe speed in a restricted zone
Torque and force limiting	Torque/force limiting	Joint-state or bus injection; sensor manipulation	Primitives demonstrated; robot-safety chain composed	Loss of balance, dropped payload, or excessive force
Geofencing	Geofencing	GNSS or pose spoofing	Spoofing demonstrated; robot-safety chain composed	Boundary violation or entry into a restricted area
Safe state	Safe state/degraded mode	Sensor-baseline poisoning	Primitive demonstrated; safety-response chain composed	Failure to trigger an alarm, shutdown, or evacuation response
Obstacle avoidance	Obstacle avoidance	Adversarial visual patch or LiDAR spoofing	Primitives demonstrated; robot-safety chain composed unless demonstrated on the specific platform	Collision or unnecessary stop
Sensor cross-validation	Sensor cross-validation	Coordinated camera and LiDAR manipulation	Coordinated multimodal manipulation demonstrated; robot-safety outcome composed	Collision or unnecessary braking

Human-in-the-loop approval	Not in Table 1 (operator approval)	Electromagnetic injection into a capacitive touchscreen	False-touch injection demonstrated; robot-approval chain composed	Safety-relevant action proceeds without genuine operator approval
Radar and ranging spoofing	Safe speed monitoring; geofencing (input channel)	Radar point-cloud injection or ranging manipulation	Attack primitives demonstrated on commercial hardware; safety consequence composed	Unsafe speed near a hazard or access-control failure
Acoustic resonance against inertial sensors	Not in Table 1 (balance and attitude control)	Acoustic injection against MEMS accelerometers or gyroscopes	Sensor manipulation demonstrated; platform-specific safety outcome depends on control architecture	Loss of balance or attitude control
Tampered OTA update	Cross-cutting: changes parameters of several mechanisms (e.g. torque, speed, geofence limits)	Update-channel substitution	Update-security weakness and manipulation primitives documented; safety-policy chain composed	Persistent alteration of a safety-relevant parameter
Malicious skill module	Cross-cutting: alters limits (e.g. safe speed monitoring, torque/force limiting)	Malicious or insufficiently vetted downloadable software module	Conceptual robot-specific chain grounded in documented software-supply-chain precedent	Relaxed speed, force, or other safety-relevant limit
Cross-modal adjudication failure	Sensor cross-validation (single-channel variant)	Single-channel manipulation creating disagreement between modalities	Single-modal attacks against fusion demonstrated; robot adjudication chain composed	Robot proceeds based on the compromised channel
Fleet time-synchronization attack	Not in Table 1 (multi-robot coordination)	Shared time-reference spoofing	Time-spoofing primitive demonstrated; multi-robot safety chain composed	Miscoordinated collision avoidance between robots, near-miss, or collision

Thermal spoofing against human-presence detection	Not in Table 1 (human-presence detection; affects safe speed monitoring and safe state)	Thermal decoy, masking, or signature manipulation	Thermal pedestrian-detection manipulation demonstrated; robot-safety chain composed	Failure to detect a person or unnecessary reduced-speed/safe-state response
---	---	---	---	---

Table 2. Evidence status by case

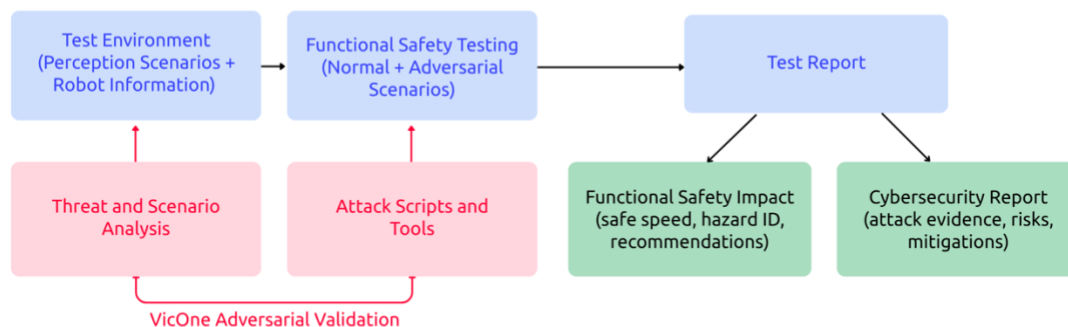
Emergency stop is covered in Section 2.3; watchdog defends a different failure mode (Table 1).

Chapter 5. Validating Safety Under Adversarial Conditions

5.1 Extending the existing safety validation framework

One practical way to integrate cybersecurity and functional safety is to extend the validation process robots already go through, adding test conditions produced by deliberate manipulation, rather than natural variation or component failure.

1. **Threat and Scenario Analysis.** Identify which safety-relevant trust anchors (Table 1) could plausibly be manipulated.
2. **Attack Scripts and Tools.** Convert the manipulation into a reproducible, automated test artifact.
3. **Test Environment.** Introduce the injected condition into the test environment alongside normal inputs.
4. **Functional Safety Testing (Normal + Adversarial).** Exercise the robot under normal and adversarial conditions so its behavior can be compared directly.
5. **Test Report.** Document two complementary outputs: a Functional Safety Impact assessment covering whether safe speed was maintained or violated, identified hazards, and improvement recommendations; and a Cybersecurity Report covering attack results and evidence, vulnerabilities and risks, and mitigation recommendations.



Safer robots that can operate safely even when facing attackers.

Figure 3. VicOne's Safety Validation Framework: adversarial conditions injected into the existing functional safety test environment.⁵⁰

5.2 Three questions, asked in order

Within Threat and Scenario Analysis, three questions should be asked in sequence: (1) Is the vulnerability reachable? (2) Can it be exploited? (3) If exploited, which part of the robot's behavior changes, and how?

The third question determines whether a security finding has a safety consequence. Answering it requires testing the manipulated condition within the functional safety environment rather than stopping the analysis once exploitability is confirmed.

These questions can be expanded into a seven-step validation sequence:

1. **Identify the safety trust anchor** for the mechanism under review.
2. **Identify the specific cyber manipulation path** that could falsify it.
3. **Establish reachability.** Can an attacker, with a defined level of access, reach the relevant component, interface, sensor, network, or model?
4. **Establish exploitability.** Can the identified path be used to manipulate the trust anchor?
5. **Measure behavioral change.** Does the manipulation change the robot's perception, state estimation, decision, trajectory, speed, force, or action?
6. **Determine safety impact.** Does the behavioral change invalidate a safety requirement, assumption, or mechanism?
7. **Validate mitigation.** Does the proposed mitigation prevent the manipulation, behavioral change, or physical consequence when tested against the same injected condition?

Steps 1 and 2 build on the trust-anchor and manipulation-path analysis developed in earlier chapters. Steps 3 and 4 establish whether the path is feasible, while Steps 5 through 7 determine its behavioral and safety consequences and whether the mitigation is effective. Section 5.3 describes how software-in-the-loop testing can support those later steps.

5.3 Software-in-the-loop

A robot's control software and behavioral model can be integrated with a digital twin or simulation environment. Platforms such as NVIDIA Isaac Sim⁵¹ or MuJoCo⁵² can support this type of testing. The manipulated condition is then introduced in a form that approximates what the robot's sensors and interfaces would receive.

This allows evaluation of the resulting behavior of safety controls: Does the torque limiter still cap force when its input is falsified (Section 4.3)? Does the geofence still hold once the position estimate is manipulated (Section 4.4)?

This type of testing can be performed before deployment and repeated as new vulnerabilities, attack techniques, or system changes are identified. It can also support evaluation of interim mitigations while a permanent mitigation is being developed.

The fidelity requirement is important. A digital twin adequate for functional safety testing — for example, one modeling nominal sensor noise and expected component tolerances — is not necessarily adequate for adversarial validation. An attack such as the coordinated multi-sensor

case in Section 4.7 may depend on details of a sensor's physical response that a coarse geometric simulation does not model faithfully.

Where the injected condition targets a physical sensing effect rather than a purely logical one — the electromagnetic touch-injection case in Section 4.8, for example — software-in-the-loop testing may need to be complemented by hardware-in-the-loop testing. This retains the relevant sensor or actuator hardware while simulating the surrounding environment.

5.4 A cross-disciplinary assessment by design

Security teams characterize the exploit. Systems and robotics engineers determine how it changes physical behavior. Safety teams evaluate whether that change constitutes a hazard.

Considered separately, each discipline addresses only part of the problem. Evaluating the same injected condition across security, robotics, and safety provides a more complete basis for assessing both exploitability and physical consequence.

5.5 Relationship to existing functional safety standards

This methodology deliberately extends, rather than replaces, existing functional safety frameworks — including the IEC 61508,³¹ and robot-specific standards such as ISO 10218⁵³ and ISO/TS 15066⁵⁴ for collaborative robots.

These standards already address hazard identification, risk assessment, and validation evidence. They do not necessarily address adversarial manipulation of the specific inputs and trust dependencies on which a safety function relies.

Identifying trust anchors and their manipulation paths can complement the hazard-identification activities such standards already require. Similarly, reachability, exploitability, and behavioral-change questions can extend existing risk-assessment and validation activities.

For organizations with established functional safety processes, adversarial validation can therefore build on existing engineering and assurance activities rather than require an entirely separate safety process.^{55,56}

Chapter 6. Security-in-the-Loop: Continuous Verification

6.1 From design-time validation to continuous verification

Section 3.4 already established that the conditions validated in Chapter 5 may change after deployment. Software updates, changes in operating environments, and fleet-learning pipelines can alter the assumptions on which earlier validation was based.

Operational data may also flow back into engineering, where it can inform updated models, policies, or software that are later redeployed to the robot. The same data can support security monitoring by providing evidence of how the system behaves after deployment.

6.2 What runtime monitoring needs to detect

This chapter addresses a specific failure mode: not only a robot that crashes or stops responding, but also one that continues executing its control loop correctly while acting on conditions that no longer match what was validated.

Behavioral baselining across comparable systems can help identify this type of divergence. A unit behaving differently from its peers may warrant investigation even when no specific vulnerability has yet been identified.

Raw sensor values may vary substantially across robots operating in different environments, so monitoring can also focus on patterns in control outputs and their relationship to sensor inputs. Relevant signals may include:

- the distribution of commanded velocities relative to the proximity estimates that should constrain them;
- the rate and magnitude of torque corrections under nominally similar loads;
- the frequency of safe-state transitions per operating hour; and
- the innovation sequence of a state estimator: the difference, at each time step, between what a sensor reports and what the estimator predicts.

Depending on the implementation, changes in the innovation sequence may help reveal inconsistencies that are less apparent from raw sensor values alone.

None of these signals individually proves an attack. Their value is comparative, helping identify behavior that warrants further investigation.^{27,28}

6.3 Context-aware runtime response: bounded, not improvised

A runtime response layer can correlate system state, commands, sensor input, and environmental context, and select from a small set of pre-validated response policies — such as reducing permitted speed near people while a permanent mitigation is prepared.

Its role is bounded: it selects from options security, robotics, and safety teams have already jointly evaluated, rather than improvising a new safety judgment under adversarial conditions.

A response layer that must reason from scratch under adversarial conditions may itself depend on the same kinds of inputs and trust assumptions examined in Chapter 2. Pre-validated response policies help reduce that additional uncertainty.

6.4 Closing the loop back into engineering

Details of the anomaly, the runtime response, and the surrounding context are fed back to engineering teams, who reproduce the condition in simulation, reassess it, and use it to inform an updated, revalidated policy or model.

This creates a recurring assurance cycle: **security research** → **threat intelligence** → **adversarial validation** → **runtime evidence** → **safety reassessment**.

Under this model, the safety case can be reassessed as new evidence, system changes, and operating conditions emerge after deployment.

6.5 Interaction with incident investigation

This loop also has value when investigating an unsafe event. Different causes may produce a similar physical outcome — including a mechanical fault, an ordinary control error, or manipulated perception or reasoning input.

Distinguishing among these possibilities can require **decision provenance**: a record of the robot's relevant system state at the time, the inputs that contributed to it, the channels through which those inputs arrived, and the checks applied before the resulting action was executed.

When this information is not retained, reconstructing the relationship between an input, the robot's internal state, and the resulting action becomes harder. This is particularly relevant in systems that combine information from multiple sensing, software, and AI components.

Incomplete provenance can therefore limit an investigation's ability to determine whether cybersecurity contributed to the event and can reduce the evidence available for subsequent threat modeling and safety reassessment.

An organization that has already implemented the monitoring described in Section 6.2 may be better positioned to support such investigations because some of the same runtime data used for continuous assurance can also provide evidence for post-incident analysis.

6.6 A worked example: from trust anchor to test report

The preceding chapters have developed several parts of the framework: the trust-anchor concept, the perception–reasoning–action chain, the reachability/exploitability/behavioral-change sequence, and the continuous-assurance loop. The following composite example traces these elements through a single scenario to show how they can work together in practice.

Consider a warehouse AMR whose safe-speed mechanism scales velocity based on the estimated distance to nearby warehouse staff, using a fused depth camera and 2D-LiDAR proximity estimate (Table 1, row 3). The scenario draws on the coordinated multi-sensor deception discussed in Section 4.7. Its evidence status remains a demonstrated primitive rather than a full end-to-end hardware demonstration on this specific AMR platform.

Step	Validation question	Application to the AMR example
1. Identify the safety trust anchor	What information does the safety mechanism rely on?	The fused human-proximity estimate consumed by the motion controller.
2. Identify the cyber manipulation path	How could that trust anchor be falsified?	A physical decoy designed to affect both the depth camera and LiDAR, targeting the agreement on which the fusion logic relies.
3. Establish reachability	Can the attacker reach the relevant input or interface?	Determine whether the decoy can plausibly be introduced into the AMR's operating environment under conditions in which both sensing channels process it.
4. Establish exploitability	Can the manipulation actually change the trust anchor?	Test whether the adversarial condition materially alters the fused human-proximity estimate. If simulation cannot reproduce the relevant physical sensing effects with sufficient fidelity, hardware-in-the-loop testing may be needed.
5. Measure behavioral change	What changes in the robot's behavior?	Observe whether the altered proximity estimate changes commanded velocity, trajectory, or another relevant control output.
6. Determine safety impact	Does the behavioral change invalidate a safety requirement or assumption?	Determine whether the AMR maintains a commanded velocity above the safe-speed threshold

		that should apply at the actual distance from a person.
7. Validate mitigation	Does the mitigation prevent the manipulation, behavioral change, or consequence?	Test changes to the fusion logic, additional consistency checks, or an interim operational control against the same adversarial condition.

Table 3. Applying the seven-step validation sequence to a warehouse AMR scenario

The resulting Functional Safety Impact assessment records the affected safety requirement, observed behavioral change, and resulting hazard. The Cybersecurity Report records the manipulation path, supporting evidence, identified weakness, and recommended mitigations.

The process does not end with the test report. If a permanent mitigation cannot be deployed immediately, the bounded runtime response described in Section 6.3 can provide an interim control. Fleet-level monitoring can then help identify behavioral divergence or sensor-fusion inconsistencies while the mitigation is developed and validated.

Once the mitigation has been validated against the same adversarial condition, it can move through the organization's controlled release process. Runtime evidence can then feed back into subsequent safety reassessment, completing the continuous-assurance loop described in this chapter.

This example shows how adversarial validation can build on existing functional safety and engineering activities by adding deliberate manipulation to the conditions those processes evaluate.

Chapter 7. Recommendations

The recommendations below translate the paper’s analysis into actions for three groups: system builders and integrators, operators and fleet managers, and regulators, standards bodies, and researchers. Each recommendation points back to the analysis that supports it.

Recommendations marked “**do immediately**” can generally begin within existing safety, security, monitoring, or change-control processes. “**Near-term**” actions may require additional testing, tooling, or engineering work. “**Longer-horizon**” actions are more likely to require broader coordination, new standards, or changes to existing assurance and vulnerability-management practices.

System builders and integrators

Primary action: Document the trust anchor behind each safety mechanism before extending testing to adversarial conditions.

1. Identify and document the safety trust anchor for each safety mechanism (**do immediately**).
2. Where supported, enable authentication and integrity protections for safety-relevant messages and interfaces, and identify paths where equivalent protections are absent or not enforced (**do immediately**).
3. Add relevant adversarial scenarios from Chapter 4 to existing functional safety tests (**do immediately**).
4. Evaluate fusion logic against coordinated, multi-channel manipulation, not only single-sensor spoofing (Section 4.7) (**near-term**).
5. Extend component and provenance tracking to AI models, model versions, fine-tuning data, and other artifacts that can affect safety-relevant behavior (**near-term**).
6. Require signature verification and integrity protection for over-the-air updates, and treat safety-relevant parameters such as speed, force, and geofence limits as protected configuration whose changes are authenticated and logged (Section 4.11) (**do immediately**).
7. Require code signing, permission controls, and behavioral review for downloadable skill or plug-in modules, so that a module cannot modify safety-relevant parameters as a side effect of its advertised function (Section 4.12) (**near-term**).
8. Treat operator-approval inputs, such as touchscreen confirmations and software stop controls, as safety-relevant inputs that need their own protection, rather than as an independent backstop by default (Section 4.8) (**near-term**).
9. Add independent cross-checks for safety inputs that depend on a single sensing or ranging method, such as radar or UWB distance and thermal human-presence detection, validating them against another modality or a stored map (Sections 4.9 and 4.15) (**near-term**).
10. Limit how far and how fast detector baselines and calibration values can drift, and check safe-state inputs against a reference that does not share the same sensing chain (Section 4.5) (**near-term**).

11. Validate inertial-sensor-derived attitude and balance estimates against independent state sources, and do not rely on cross-axis consistency checks alone, since coordinated multi-axis acoustic injection can defeat them; where the platform is exposed, also consider acoustic isolation or resonance-tolerant sensor selection (Section 4.10) (**near-term**).

Operators and fleet managers

Primary action: Monitor for behavioral divergence, not only unit-level health signals.

1. Establish fleet-level behavioral baselines using the signals in Section 6.2 — including innovation sequences, torque-correction patterns, and safe-state transition frequency as a starting instrument set where applicable (**do immediately**).
2. Include calibration updates and model-update pipelines in security monitoring and change-control processes, particularly where they can alter safety-relevant behavior (**do immediately**).
3. Maintain sufficient decision provenance to help distinguish mechanical faults, control errors, and manipulated perception or reasoning inputs after an incident (**longer-horizon**).
4. Track temporary runtime responses until the underlying mitigation has been validated and the temporary control can be reassessed (**do immediately**).
5. Validate the time source used for fleet coordination against an independent reference, and flag clock offsets between robots (Section 4.14) (**near-term**).

Regulators, standards bodies, and researchers

Primary action: Strengthen the evidence connecting adversarial conditions to safety behavior.

1. Consider how documented adversarial validation evidence can be incorporated into applicable certification and assurance frameworks, building on the relationship to IEC 61508-family standards discussed in Section 5.5 (**longer-horizon**).
2. Support the development of decision-provenance approaches for AI-driven robotic systems (**longer-horizon**).
3. Expand end-to-end demonstration research that connects a documented manipulation to measurable changes in robot behavior and safety outcomes (**longer-horizon**).
4. Support vulnerability-management approaches that can represent weaknesses arising from interactions between models or components, including cases that do not map cleanly to a single vulnerable component (Section 2.6) (**longer-horizon**).^{55,56}

Section 6.6's worked example provides a practical template for piloting these recommendations on a single mechanism and platform before extending them across a fleet or product line: identify one trust anchor, construct one adversarial test, evaluate it through the existing validation environment, and use the resulting evidence to inform subsequent testing.

Conclusion. Why Robot Safety Depends on Cybersecurity

This paper opened with a scenario that may seem contradictory at first: a robot receives an authorized command, runs genuine software, and executes exactly as instructed — yet still produces an unsafe outcome. Authorization and integrity verification confirm that a command came from an authorized source and was not altered. Neither establishes that, in the current situation, its physical consequence will not cause harm.

For many modern robotic safety mechanisms, that gap reflects an architectural dependency: safety decisions rely on software, sensors, communications, and computed state that may themselves be exposed to manipulation. An adversary's ability to deliberately reproduce a condition that functional safety correctly treats as rare can change the assumptions behind that safety analysis. A mechanism may continue to execute as designed while acting on a manipulated premise.

The response is not to abandon functional safety but to extend it: a validation methodology that injects adversarial conditions into a robot's existing safety test environment, producing evidence of what a robot actually does under those conditions; and a continuous discipline — fleet-level monitoring, bounded runtime response, and a feedback loop back into engineering — because deployment does not freeze the conditions a robot was validated against. These practices can build on methods already used across cybersecurity and functional safety, including reachability analysis, hazard assessment, validation, and risk analysis. The additional task is to identify the safety-relevant trust dependencies those methods need to examine under adversarial conditions.

Cybersecurity, understood this way, is not an additional layer of protection stacked on top of functional safety. It helps determine whether the assumptions a safety case already depends on — that a sensor reading is accurate, that a position estimate is reliable, or that a human-presence signal is genuine — still hold once deliberate manipulation is considered.

Robot safety therefore depends on cybersecurity wherever a safety mechanism relies on digital information that an adversary could manipulate.

For robot safety to remain meaningful in increasingly connected, AI-enabled systems, the integrity of safety-relevant information should be treated as part of the safety case itself. That means identifying the information a safety mechanism depends on, testing whether it can be deliberately manipulated, and reassessing the resulting safety claim as systems and operating conditions change.

Robot safety depends not only on what a system is designed to do, but also on whether the information guiding that behavior can still be trusted.

References

- ¹ VicOne LAB R7. *Securing the Rise of AI Robots: Cyber Risks, Real-World Threats, and Defense Strategies*. White paper, 2025. <https://info.vicone.com/ai-robotics-security-risk-whitepaper>
- ² Chu, J., and J. Yueh. "Fireside Chat." *Robotic Hacking Community, DEF CON 34*, Las Vegas, August 7, 2026. <https://robotichackingcommunity.com/program.html>
- ³ Xiao, S., et al. "SoK: Understanding the Fundamentals and Implications of Sensor Out-of-band Vulnerabilities." *Network and Distributed System Security (NDSS) Symposium*, 2026. <https://dx.doi.org/10.14722/ndss.2026.230450>
- ⁴ Cao, Y., et al. "You Can't See Me: Physical Removal Attacks on LiDAR-Based Autonomous Vehicles Driving Frameworks." *32nd USENIX Security Symposium*, 2023, 2993–3010. <https://www.usenix.org/conference/usenixsecurity23/presentation/cao>
- ⁵ Shin, H., et al. "Illusion and Dazzle: Adversarial Optical Channel Exploits against Lidars for Automotive Applications." *Cryptographic Hardware and Embedded Systems (CHES)*, 2017, 445–467. https://doi.org/10.1007/978-3-319-66787-4_22 <https://eprint.iacr.org/2017/613>
- ⁶ Quarta, D., et al. "An Experimental Security Analysis of an Industrial Robot Controller." *IEEE Symposium on Security and Privacy (S&P)*, 2017, 268–286.
- ⁷ Maggi, F., et al. *Rogue Robots: Testing the Limits of an Industrial Robot's Security*. Trend Micro Research, 2017. <https://documents.trendmicro.com/assets/wp/wp-industrial-robot-security.pdf>
- ⁸ Kune, D. F., et al. "Ghost Talk: Mitigating EMI Signal Injection Attacks against Analog Sensors." *IEEE Symposium on Security and Privacy (S&P)*, 2013, 145–159.
- ⁹ Kong, Z., Y. Zhang, and S. Y. Chau. "Phantom Force: Injecting Adversarial Tactile Perceptions into Embodied Intelligence via EMI." *ACM Asia CCS*, 2026. <https://doi.org/10.1145/3779208.3804894> <https://arxiv.org/abs/2605.13492>
- ¹⁰ Wang, K., et al. "GhostTac: Manipulating Tactile Sensors without Physical Contact." *arXiv preprint*, August 2026. <https://arxiv.org/abs/2608.20817>
- ¹¹ Nashimoto, S., et al. "Sensor CON-FUSION: Defeating Kalman Filter in Signal Injection Attack." *ACM AsiaCCS*, 2018, 511–524.
- ¹² Zhang, G., et al. "DolphinAttack: Inaudible Voice Commands." *ACM CCS*, 2017, 103–117. <https://arxiv.org/abs/1708.09537>
- ¹³ Xia, Q., Q. Chen, and S. Xu. "Near-Ultrasound Inaudible Trojan (NUIT): Exploiting Your Speaker to Attack Your Microphone." *32nd USENIX Security Symposium*, 2023, 4589–4606.
- ¹⁴ Greshake, K., et al. "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection." *ACM Workshop on Artificial Intelligence and Security (AISec)*, 2023. <https://arxiv.org/abs/2302.12173>
- ¹⁵ Robey, A., et al. "Jailbreaking LLM-Controlled Robots." *arXiv preprint*, 2024. <https://arxiv.org/abs/2410.13691>
- ¹⁶ Shen, J., et al. "Drift with Devil: Security of Multi-Sensor Fusion Based Localization in High-Level Autonomous Driving under GPS Spoofing." *29th USENIX Security Symposium*, 2020, 931–948. <https://www.usenix.org/conference/usenixsecurity20/presentation/shen>
- ¹⁷ Humphreys, T. E., et al. "Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer." *ION GNSS*, 2008.
- ¹⁸ Kerns, A. J., et al. "Unmanned Aircraft Capture and Control via GPS Spoofing." *Journal of Field Robotics* 31, no. 4, 2014, 617–636.
- ¹⁹ Brown, T. B., et al. "Adversarial Patch." *arXiv preprint*, 2017. <https://arxiv.org/abs/1712.09665>

- ²⁰ Eykholt, K., et al. "Robust Physical-World Attacks on Deep Learning Visual Classification." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, 1625–1634.
<https://arxiv.org/abs/1707.08945>
- ²¹ Cao, Y., et al. "Adversarial Sensor Attack on LiDAR-Based Perception in Autonomous Driving." *ACM CCS*, 2019, 2267–2281.
- ²² Sun, J., et al. "Towards Robust LiDAR-Based Perception in Autonomous Driving: General Black-Box Adversarial Sensor Attack and Countermeasures." *29th USENIX Security Symposium*, 2020, 877–894.
<https://www.usenix.org/conference/usenixsecurity20/presentation/sun>
- ²³ Cao, Y., et al. "Invisible for Both Camera and LiDAR: Security of Multi-Sensor Fusion Based Perception in Autonomous Driving under Physical-World Attacks." *IEEE Symposium on Security and Privacy (S&P)*, 2021, 176–194. <https://arxiv.org/abs/2106.09249>
- ²⁴ Hallyburton, R. S., et al. "Security Analysis of Camera-LiDAR Fusion against Black-Box Attacks on Autonomous Vehicles." *31st USENIX Security Symposium*, 2022, 1903–1920.
<https://www.usenix.org/conference/usenixsecurity22/presentation/hallyburton>
- ²⁵ Jin, Z., et al. "Unity Is Strength? Benchmarking the Robustness of Fusion-Based 3D Object Detection against Physical Sensor Attack." *ACM Web Conference (WWW)*, 2024, 3031–3042.
- ²⁶ Mayoral-Vilches, V., et al. "SROS2: Usable Cyber Security Tools for ROS 2." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.
<https://doi.org/10.1109/IROS47612.2022.9982129>
- ²⁷ Choi, H., et al. "Detecting Attacks against Robotic Vehicles: A Control Invariant Approach." *ACM CCS*, 2018, 801–816.
- ²⁸ Quinonez, R., et al. "SAVIOR: Securing Autonomous Vehicles with Robust Physical Invariants." *29th USENIX Security Symposium*, 2020, 895–912.
<https://www.usenix.org/conference/usenixsecurity20/presentation/quinonez>
- ²⁹ Leu, P., et al. "Ghost Peak: Practical Distance Reduction Attacks Against HRP UWB Ranging." *31st USENIX Security Symposium*, 2022. <https://www.usenix.org/conference/usenixsecurity22/presentation/leu>
- ³⁰ Wang, Z., Y. Wu, and Q. Niu. "Multi-Sensor Fusion in Automated Driving: A Survey." *IEEE Access* 8, 2020, 2847–2868.
- ³¹ International Electrotechnical Commission. *IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*. 2010.
- ³² Health and Safety Executive. *Reducing Risks, Protecting People: HSE's Decision-Making Process*. HSE Books, 2001.
- ³³ Dayanikli, G. Y., et al. "Physical-Layer Attacks Against Pulse Width Modulation-Controlled Actuators." *31st USENIX Security Symposium*, 2022, 953–970.
<https://www.usenix.org/conference/usenixsecurity22/presentation/dayanikli>
- ³⁴ Wang, K., et al. "GhostTouch: Targeted Attacks on Touchscreens without Physical Touch." *31st USENIX Security Symposium*, 2022, 1543–1559.
<https://www.usenix.org/conference/usenixsecurity22/presentation/wang-kai>
- ³⁵ Shan, H., et al. "Invisible Finger: Practical Electromagnetic Interference Attack on Touchscreen-Based Electronic Devices." *IEEE Symposium on Security and Privacy (S&P)*, 2022, 1246–1262.
- ³⁶ Hunt, D., et al. "MadRadar: A Black-Box Physical Layer Attack Framework on mmWave Automotive FMCW Radars." *NDSS Symposium*, 2023. <https://arxiv.org/abs/2311.16024>
- ³⁷ Son, Y., et al. "Rocking Drones with Intentional Sound Noise on Gyroscopic Sensors." *24th USENIX Security Symposium*, 2015, 881–896.
- ³⁸ Trippel, T., et al. "WALNUT: Waging Doubt on the Integrity of MEMS Accelerometers with Acoustic Injection Attacks." *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2017.
- ³⁹ Tu, Y., et al. "Injected and Delivered: Fabricating Implicit Control over Actuation Systems by Spoofing Inertial Sensors." *27th USENIX Security Symposium*, 2018, 1545–1562.

- ⁴⁰ Jeong, J., et al. "Un-Rocking Drones: Foundations of Acoustic Injection Attacks and Recovery Thereof." *NDSS Symposium*, 2023. <https://www.ndss-symposium.org/ndss-paper/un-rocking-drones-foundations-of-acoustic-injection-attacks-and-recovery-thereof>
- ⁴¹ Mayoral-Vilches, V., et al. "Introducing the Robot Security Framework (RSF), a Standardized Methodology to Perform Security Assessments in Robotics." *arXiv preprint*, 2018. <https://arxiv.org/abs/1806.04042>
- ⁴² National Vulnerability Database. *CVE-2025-45467: Unitree Go1 Firmware Update Relies Solely on MD5 Checksums*. 2025. <https://nvd.nist.gov/vuln/detail/CVE-2025-45467>
- ⁴³ Palo Alto Networks Unit 42. "OpenClaw's Skill Marketplace and the Emerging AI Supply Chain Threat." 2026. <https://unit42.paloaltonetworks.com/openclaw-ai-supply-chain-risk/>
- ⁴⁴ The Hacker News. "Researchers Find 341 Malicious ClawHub Skills Stealing Data from OpenClaw Users." February 2026. <https://thehackernews.com/2026/02/researchers-find-341-malicious-clawhub.html>
- ⁴⁵ Cheng, Z., et al. "Fusion Is Not Enough: Single Modal Attacks on Fusion Models for 3D Object Detection." *International Conference on Learning Representations (ICLR)*, 2024. <https://arxiv.org/abs/2304.14614>
- ⁴⁶ Spanghero, M., and P. Papadimitratos. "Time-Based GNSS Attack Detection." *arXiv preprint*, 2025. <https://arxiv.org/abs/2502.03868>
- ⁴⁷ Tibaldo, C., et al. "GNSS-WASP: GNSS Wide Area SPoofting." *34th USENIX Security Symposium*, 2025. <https://www.usenix.org/system/files/usenixsecurity25-tibaldo.pdf>
- ⁴⁸ Zhu, X., et al. "Fooling Thermal Infrared Pedestrian Detectors in Real World Using Small Bulbs." *AAAI Conference on Artificial Intelligence* 35, no. 4, 2021, 3616–3624. <https://doi.org/10.1609/aaai.v35i4.16477>
- ⁴⁹ Zhu, X., et al. "Infrared Invisible Clothing: Hiding from Infrared Detectors at Multiple Angles in Real World." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. <https://doi.org/10.1109/CVPR52688.2022.01296>
- ⁵⁰ VicOne. "VicOne Brings Physical AI Safety Stress Test to DEF CON 34 to Test How Cyber Risk Can Change Robot Behavior." Press release, 2026. <https://lab-r7.vicone.com/press-releases/vicone-brings-physical-ai-safety-stress-test-to-def-con-34-to-test-how-cyber-risk-can-change-robot-behavior/>
- ⁵¹ NVIDIA. *Isaac Sim*. <https://developer.nvidia.com/isaac/sim>
- ⁵² Todorov, E., T. Erez, and Y. Tassa. "MuJoCo: A Physics Engine for Model-Based Control." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, 5026–5033. <https://doi.org/10.1109/IROS.2012.6386109>
- ⁵³ International Organization for Standardization. *ISO 10218-1:2025 and ISO 10218-2:2025, Robotics — Safety Requirements*. 2025. <https://www.iso.org/standard/73933.html>
- ⁵⁴ International Organization for Standardization. *ISO/TS 15066:2016, Robots and Robotic Devices — Collaborative Robots*. 2016. <https://www.iso.org/standard/62996.html>
- ⁵⁵ IEC 62443-3-3 and IEC 62443-4-2. *Industrial communication networks — Network and system security*. International Electrotechnical Commission.
- ⁵⁶ National Institute of Standards and Technology. *NIST SP 800-82 Rev. 3: Guide to Operational Technology (OT) Security*. 2023. <https://doi.org/10.6028/NIST.SP.800-82r3>
- ⁵⁷ Carlini, N., and D. Wagner. "Towards Evaluating the Robustness of Neural Networks." *IEEE Symposium on Security and Privacy (S&P)*, 2017. <https://arxiv.org/abs/1608.04644>
- ⁵⁸ Goodfellow, I. J., J. Shlens, and C. Szegedy. "Explaining and Harnessing Adversarial Examples." *International Conference on Learning Representations (ICLR)*, 2015. <https://arxiv.org/abs/1412.6572>
- ⁵⁹ Open Robotics. "About Security." *ROS 2 Documentation: Rolling*. <https://docs.ros.org/en/rolling/Concepts/Intermediate/About-Security.html>
- ⁶⁰ Kravchik, M., B. Biggio, and A. Shabtai. "Poisoning Attacks on Cyber Attack Detectors for Industrial Control Systems." *ACM/SIGAPP Symposium on Applied Computing (SAC)*, 2021. <https://doi.org/10.1145/3412841.3441892> <https://arxiv.org/abs/2012.15740>

⁶¹ Urbina, D. I., et al. "Limiting the Impact of Stealthy Attacks on Industrial Control Systems." ACM CCS, 2016, 1092–1105. <https://csrc.nist.gov/pubs/conference/2016/10/24/limiting-the-impact-of-stealthy-attacks-on-ics/final>



Why Robot Safety Depends on Cybersecurity: Protecting the information that safety decisions rely on
Copyright © 2026 VicOne Inc.
All Rights Reserved.

Learn more about VicOne
by visiting VicOne.com
or scanning this QR code:

