Logic Follows Lies: How PLCs and RTUs Execute Adversarial Intent

Norris Cornell · March 2026 · Version: Article · Part 3 of 3

Parts 1 and 2 established why systems trust inputs they shouldn't, and who is exploiting that gap at scale. Part 3 closes the loop: what does failure actually look like when deterministic logic executes on manipulated inputs — and what does detection require at this layer?

The pressure gauge reads normal. The flow meter reads normal. The HMI screen shows green across the board.

The pump has been running dry for eleven minutes.

No alarm has fired. No operator has intervened. The control system is executing exactly as designed — faithfully processing the inputs it was given, making every logical decision correctly, and driving a piece of critical equipment toward destruction with complete confidence.

This is not a malware story. No code was compromised. No vulnerability was exploited in the traditional sense. The attacker manipulated a 4–20mA analog signal on a single sensor line — enough to convince the control loop that pressure was nominal while the actual process condition was catastrophic. The PLC trusted what it was told. The HMI displayed what the PLC reported. The operator watched a screen that had been quietly disconnected from physical reality.

By the time the failure was visible, the forensic window was already closing.

How Deterministic Logic Becomes a Liability

PLCs are not intelligent systems. They are deterministic ones. Every scan cycle — typically every 10 to 100 milliseconds — a PLC reads its inputs, executes its logic, and updates its outputs. It does not question what it reads. It does not cross-reference sensor values against physical possibility. It executes.

This is a feature, not a flaw. Real-time industrial control requires determinism. A PLC that paused to verify the authenticity of a pressure reading before actuating a valve would introduce latency that could itself cause harm. The design assumption is that the inputs are trustworthy — because for most of the last fifty years, in isolated networks with physical security perimeters,

they were.

That assumption no longer holds.

When an adversary manipulates an input — whether through signal injection at the physical layer, a compromised field device, or a spoofed protocol message — the PLC does not detect the manipulation. It executes on the lie. And because PLC logic is designed to handle edge cases by triggering further automated responses, a single manipulated input can cascade through an entire control sequence, each downstream decision correct given what the system was told, each one carrying the process further from safe operating conditions.

RTUs compound this problem. Remote terminal units sit at the edge of OT networks — substations, pipeline monitoring points, water treatment outfalls — often without local operators to catch what the data cannot. When an RTU reports falsified telemetry, the control center sees what the adversary wants it to see. The distance between the sensor and the human who might notice something is wrong is measured in miles, not feet.

What Adversarial Failure Looks Like

The defender's core problem is this: manipulated-input failures are specifically designed to look like normal equipment faults.

A pump that fails because an adversary spoofed its pressure sensor looks, in the log data, like a pump that failed due to cavitation or wear. A protective relay that trips unnecessarily because of a timing manipulation looks like a relay that responded to a transient fault. The failure mode is real. The cause is invisible — or was, until someone thought to look at the sensor data, the signal integrity, the timing relationships between inputs that should have been correlated.

This is not accidental. Advanced adversaries pre-position not just to cause failure but to complicate recovery. In the 2015 Ukraine grid attacks, the attackers didn't simply cut power — they systematically degraded the defenders' ability to respond, disabling backup power systems, corrupting firmware on serial-to-Ethernet converters, and jamming the phone lines operators needed for manual coordination. The attack on the physical process and the attack on the defenders' situational awareness were the same operation.

Most incident response playbooks are not designed for this. They start at the network layer, after the damage is done, looking for the intrusion that preceded the failure. When the manipulation happened at the signals layer — below the network, before the logic — the playbook starts in the wrong place.

There is one detection primitive that traditional tooling consistently undervalues: operator intuition. Experienced process operators develop an internalized sense of how a system

sounds, feels, and behaves under normal conditions. They notice when something is subtly wrong before the data confirms it. This is not noise to be automated away. It is a detection capability that exists nowhere else in the stack — and in environments where signal-layer attacks are designed to produce plausible-looking data, it may be the first line of defense that actually works.

That intuition needs to be institutionalized. It needs to be the thing that triggers a deeper look, not the thing that gets overruled by a green dashboard.

What Detection Actually Requires

You cannot signature-detect a physics-layer attack.

Signature detection works by recognizing known bad patterns — specific malware strings, known exploit payloads, recognized attack tool behaviors. A manipulated analog signal has no signature. A spoofed sensor reading that falls within normal operating range generates no alert. A timing manipulation timed to the PLC scan cycle looks, in the data stream, like a normal process event.

Detection at this layer requires something fundamentally different: behavioral baselines.

A behavioral baseline is a model of what normal looks like — not just normal values, but normal relationships between values. Pressure and flow in a closed loop should correlate in predictable ways. Temperature and power draw in a motor should track together. When independent sensors that should be correlated suddenly disagree, that disagreement is a detection signal — even if neither sensor is individually out of range.

Cross-sensor correlation is not a new idea. What is new is the recognition that it needs to be applied specifically to the inputs that feed control logic, not just to the process outputs that humans traditionally monitor. The attack surface is upstream of where most OT monitoring currently focuses.

Timing integrity monitoring addresses a different dimension of the same problem. Many ICS failures involve timing — GPS-disciplined clocks that can be spoofed, NTP servers that can be manipulated, synchronization signals that critical infrastructure relies on for protection system coordination. A lightweight timing integrity monitor — something that cross-references multiple independent time sources and flags disagreement — is achievable in most environments today. Most environments don't have one.

The forensic problem deserves specific attention. In a physics-layer attack, the evidence of manipulation exists in the signal data — the analog values, the timestamps, the correlation patterns — at the moment the attack occurs. After the failure, that data may be overwritten,

aggregated into summaries that lose the resolution needed for analysis, or simply not collected in the first place. The forensic posture required to investigate a signals-layer attack has to be established before the incident, not after. This means historian configurations, data retention policies, and logging decisions that most OT environments have not made with this threat model in mind.

Closing the Loop

Part 1 of this series established the structural problem: critical infrastructure systems implicitly trust external signals at the physics layer, and that trust is architectural — it cannot be patched away.

Part 2 showed who is exploiting that gap and how: nation-state actors operating below the threshold of traditional ICS security frameworks, moving through the signals layer that existing defenses were never designed to cover.

Part 3 closes the arc: when those attacks execute, the failure is designed to be invisible, the forensic evidence is designed to be ambiguous, and the detection requires capabilities — behavioral baselines, cross-sensor correlation, timing integrity monitoring, and preserved forensic data — that most OT environments do not currently have.

The throughline across all three articles is the same: Trust Nothing, Validate Everything Including Physics.

That framing is not a product recommendation or a framework to certify against. It is a first-principles description of what a mature signals-layer security posture requires. It means treating physical inputs — sensor readings, timing signals, analog values, protocol messages — with the same skepticism that good security practice applies to network traffic and user credentials. It means building detection capabilities that operate at the layer where the manipulation actually occurs, not one layer above where the consequences become visible.

The gap between where most ICS security operates and where these attacks happen is not a gap that better tools alone will close. It requires a different mental model of where the attack surface begins.

It begins at the physics.

Norris Cornell researches critical infrastructure security, signal trust, and physics-layer vulnerabilities at CornellSecurity.com. This is the third and final article in the Inputs Lie series.