Blueskytec America Inc.
Michael A. Strauss | Clinton Groves

Deterministic Protective Systems: Execution Legitimacy in Control Environments

ABSTRACT

Maritime operations increasingly depend on machines acting faster than humans can observe or intervene. Propulsion, navigation, sensors, logistics, and decision aids exchange data and commands autonomously across platforms, domains, and classification boundaries. Yet the trust model underpinning these machine-to-machine interactions remains fundamentally unchanged.[1]

Existing security standards (IEC 62443[2], NIST SP 800-82[3]) and architectures (Purdue zoned networks[4]) focus on user access, network segmentation, and software policies. These controls determine who can send commands and where they are routed, but not whether a received command should be executed by the machinery at that moment. In effect, current defenses do not enforce machine-execution legitimacy – the requirement that any machine-level command be valid in its operational context before action.

This paper argues that legitimacy must be enforced deterministically at the execution boundary as the point where a control signal becomes a physical action. We define a Deterministic Protective System (DPS) as a class of mechanisms that implement a fixed, pre-validated operational envelope at this boundary. DPSs, admit only authorized state transitions. We illustrate this concept with the Stuxnet attack in which well-formed Programmable Logic Controller (PLC) instructions drove centrifuges out of safe range while falsifying sensor displays, showing that structural validity is not enough. In contrast, a DPS would have blocked those abnormal commands outright. We discuss how deterministic boundary enforcement complements traditional probabilistic defenses and show via comparison (Table 1) that DPS greatly reduces the exploitable attack surface in Operational Technology (OT)/shipboard systems. We assume attackers cannot physically breach the DPS reconfiguration path.

This enforcement primitive fills the gap left by current Industrial and Control System (ICS)/OT cyber frameworks, ensuring that only mission-appropriate commands ever reach machinery.

INTRODUCTION

ICS aboard naval vessels have very different priorities (Figure 1) than enterprise Information Technology (IT) systems. ICS/OT networks require real-time determinism, continuous availability, and safety-critical integrity. For example, automated commands to valve opens, change power level, or execute navigation maneuvers often cannot be delayed or reversed without disrupting the physical processes.

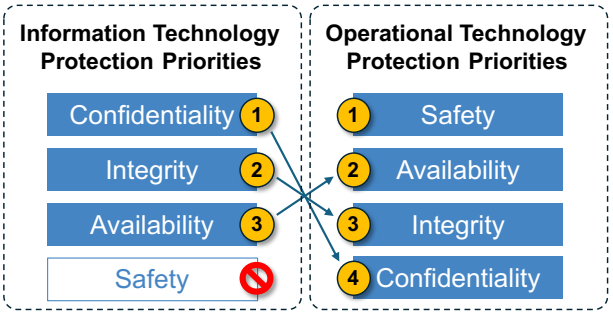

Figure 1. Conflicting Priorities of IT and OT

Conventional cybersecurity practices, i.e., user/password authentication, firewalls, network segmentation (the Purdue/ISA-99 model), malware scanning, Virtual Private Networks (VPN), and anomaly detection, were developed for IT environments where delays or reboots are tolerable. In typical IT, packets may be dropped or queries blocked without catastrophic consequence, and throughput is king. In OT, however, an

unauthorized command can immediately drive equipment to unsafe states.

Security standards for industrial control (ISA/IEC 62443) and guidelines (NIST SP 800-82) aim to secure ICS networks using risk-based policies. These frameworks prescribe zones/segments, role-based access, configuration management, and monitoring. For example, the Purdue model segments control networks from enterprise networks. Such measures decide who can speak to devices and through which channel, but they do not verify what those devices do once they receive a command. In short, after the network and access controls are satisfied, the delivered command is treated as legitimate by default.

The central question of this paper is, "Should this machine-executable signal even exist under these circumstances?" In our view, this question must be answered at the instant of execution. Current standard controls do not require enforcement of real time command validity. As a result, threats can hide as legitimate commands. We argue that adding a control primitive for execution legitimacy is both necessary and feasible. The enforcement locus for this primitive is the execution boundary, which is the interface where controller outputs turn into actuator inputs. A DPS placed at this boundary inspects each command against a pre-defined, deterministic rule set, permitting only safe operations.

This approach does not replace existing defenses, it augments them. By intercepting illicit commands at the last gate, deterministic enforcement reduces the load on traditional monitoring and detection tools. We will define DPS precisely below, illustrate why probabilistic methods alone cannot provide this safety, and outline how DPSs can be implemented. We also note that this paper assumes threat models typical of naval OT where attackers may penetrate networks or compromise software, but do not have direct physical access to the DPS logic as the reprogramming interface is protected.

THE FALSE ASSUMPTIONS

Existing ICS security approaches rest on assumptions that break down in a cyber-physical context. A common assumption is that preventing unauthorized access and monitoring execution is "good enough." In practice, once a malicious command passes through firewalls or gain network access, many common defenses become moot.[5] For instance, there is no dialogue between the signal and a turbine; it will shut down immediately regardless of whether the command was intentional. In other words, industrial actuators have no built-in veto for dangerous commands. Even if an intrusion detection system (IDS) later flags the event, it is too late, the system has already executed the instructions.

The Stuxnet incident exemplifies this gap. The "worm" infiltrated the Natanz facility's PLC network and executed precisely crafted instructions to spin uranium centrifuges at damaging speeds (as low as 2 Hz or as high as 1,410 Hz) during brief intervals. These commands were syntactically valid for the Siemens PLCs and thus passed normal input checks. At the same time, Stuxnet falsified the operators' SCADA display by feeding back recorded "normal" speed values. From the operator's point of view, every executed command was legitimate. In reality, the centrifuges were being destroyed.[6] This attack did not exploit protocol bugs or buffer overflows in the traditional sense but rather abused the fact that nothing in the system refused to execute these extreme commands that were injected onto the network.

This provides a clear example in the fact that structurally valid instructions can remain operationally illegitimate and still execute. By the time any alarm sounds, if at all, critical damage may have occurred. Conventional "detect after fact" measures therefore offer no protection at the point of system action. For naval systems, where safety and mission continuity are paramount, relying on post-execution alerts is unacceptable. To prevent catastrophes, we must enforce legitimacy before commands affect machinery. In

other words, dangerous commands must be stopped before they reach the actuator at all.

THE EXECUTION BOUNDARY

We define the execution boundary as the conceptual line where a control signal transitions to physical effect. At this boundary, the system's authority shifts from "command planning" to "physical action." To prevent illegitimate actions, this is where enforcement must occur.

Importantly, the execution boundary itself is an abstract interface. In practice, enforcement happens in hardware that sits at this interface. We break down the problem as follows:

- Execution Boundary (concept) – The point at which a digital command becomes an analog or machine action.
- Operational Envelope – The set of all permissible states and commands at that point.
- DPS Mechanism – The deterministic device enforcing the envelope.

In effect, the DPS defines the envelope and discards any signal not in it. This ensures that nothing forbidden ever gets beyond the boundary. For example, the envelope must include base controls features authorized devices, message/frame types, source-designation pairings, rate, size, timing/frequency, and per-message replay protections. Thes features may also extend to state conditions like "valve-open commands are allowed only when the pump is running and pressure is below X," or "speed-up commands cannot exceed this ramp rate." If a command violates any of these constraints, it is dropped immediately.

Crucially, the enforcement at the execution boundary must be deterministic and tamper-resistant. Software or policy engines running on CPUs are insufficient because they inherit the CPU's complexity and vulnerability.[7] CPUs are essentially universal Turing machines; a flaw can let an attacker run arbitrary code. Instead, enforcement should use the simplest automaton that can accomplish the task, since bounded logic cannot be manipulated into executing arbitrary programs.

In practice, this suggests using fixed-function hardware to perform the check. One successful pattern is the DPS placed in-line at the controller, with a physically separate management path. Only this protected management interface can reprogram DPS; normal data inputs cannot alter its logic. By physically preventing any attacker from sending changes to the DPS, the execution boundary device remains static at runtime. Any command that does not match the preloaded envelope is simply rejected.

Enforcement at the execution boundary differs fundamentally from segmentation or monitoring. Firewalls, VPNs, and network zoning place barriers upstream and isolate subsystems, but once a segment is breached, they offer no second check on each control command. Similarly, IDS/SCADA monitors observe activity but cannot stop a command that a controller blindly executes. Only a boundary check can ensure that every action is vetted. In effect, this treats the execution boundary as a final safeguard, ensuring that nothing unauthorized is allowed to pass through.

WHAT IS A DPS AND WHY ARE THEY NECESSARY

A Deterministic Protective System (DPS) is a device or component whose sole job is to enforce the legitimate operational envelope at the execution boundary. Key attributes of a DPS include:

- Execution-Point Enforcement – Validation happens exactly where commands become actions. The DPS inspects or computes all relevant bits of a command in real time and decides allow/deny before it is passed to controllers and physical output changes. This is in contrast to network firewalls or application-layer checks that occur off the physical control loop.
- Fixed Deterministic Logic – The DPS uses only pre-defined logic (no learning or

randomness). For any given input state, its output is always the same and thoroughly verifiable. This determinism means an attacker cannot exploit subtle timing or learning behavior, and the scope of possible outcomes from any input is strictly bounded.

- Non-Reliance on Heuristics – A DPS operates without anomaly scores, signatures, or statistical inference. It does not "look for" bad behavior; it only knows what good behavior is. Everything outside that set is rejected. In practice, this means DPS logic must be specified in advance (often through formal rules or code) and tested exhaustively.
- Minimal Attack Surface – Because its functionality is fixed, the DPS has virtually no attack surface in the deployed system. There is no software stack or network interface for attackers to subvert. The only critical surface is the physical reprogramming interface, which is assumed secured. This contrasts with conventional devices (PLC CPUs, networked controllers) whose rich interfaces give attackers many vectors for attack.
- Longevity and Scalability – DPS units can be engineered for the long lifetimes of naval systems. Their logic can be updated safely via the protected path using extensive safeguards if operational requirements change (for example, new legitimate commands are needed). Once deployed, the DPS remains unchanged and does not require periodic patching. This deterministic stability is ideal for ships that may operate for decades.

Because a DPS enforces a static, narrow set of behaviors, it makes advanced threats far less effective. Even if attackers develop AI-driven exploits or use quantum techniques, their ability to inject new classes of commands is futile. The DPS simply won't allow them. In effect, a DPS turns the execution boundary into a zero-trust, zero-error checkpoint.[8] This radically alters the threat model as adversaries now must find a way to compromise the DPS hardware or bypass it entirely which is precluded by our physical protection assumption.

Implementations of DPSs have been explored in industry. One example is a reactor protection system, where the critical safety logic is coded in rather than in software.[9] Another is the Hardsec guidance employed by several manufacturers, which advocates exactly this approach. In all such cases, the DPS provides a form of hardware-enforced whitelisting of control signals.[10]

WHAT DETERMINISTIC DEFENSES DO TO DETECTION

Deterministic enforcement at the boundary fundamentally changes the job of all other security mechanisms. With a DPS in place, the stream of commands and data coming out of protected subsystems is guaranteed to conform to the pre-approved envelope. From the perspective of an external observer (e.g., an IDS, a historian, or a network monitor), the system appears inert or at least fully compliant.

In practical terms, DPS protection means that any "noise," illicit command attempts, are absent by design. For example, a network IDS monitoring traffic to protected PLCs would observe only legitimate protocol transactions on schedule. Any attacker's attempt to send an unauthorized SCADA command would be blocked silently at the boundary, leaving no trace on the plant network. This greatly improves the signal-to-noise ratio for detection as alarms triggered would almost certainly indicate real intrusions or misconfigurations outside the DPS scope such as unauthorized access to the control network itself. In short, the environment becomes cleaner.

Detectors can then focus on anomalies outside the boundary. This includes human and process factors such as unexpected engineering workstations on the network, unusual traffic patterns in enterprise links, or deviations in sensor data indicative of equipment faults. These remain important. NIST SP 800-82 emphasizes that "monitoring and auditing activities are imperative to understanding the current state of the ICS". DPSs fully support this, but their presence means those monitoring tools no longer need to guard the low-level control

path, which they were never designed to do. They can be tuned to the higher-level needs of operational situational awareness and incident response.

As a result, false positives drop dramatically. Most intrusion alarms in OT networks come from benign anomalies or known deviations. If all control commands are validated by a DPS, those benign anomalies vanish, and genuine malicious activity stands out clearly. In other words, deterministic enforcement gives probabilistic defenses a nearly noise-free lens. It is still prudent to run IDS/IPS, vulnerability scans, and behavior analytics elsewhere, but their alerts will now be primarily about human or network events, not machine commands. This division of labor aligns with best practices. DPS handles the "must not fail" enforcement on the controller path, while probabilistic methods cover the rest.

ROLE OF PROBABILISTIC DEFENSE

A common question is whether adding DPS means we can "throw away cyber." The answer is no. Probabilistic defense (firewalls, IDS, anomaly detection, integrity checks, etc.) remains essential, but its scope shifts. DPS does not eliminate the need for cyber defense, it frees it to address threats outside the execution boundary.

Even with a DPS in place, attackers may still target upstream systems. For example, an attacker could compromise a development workstation and insert malicious logic into PLC code prior to deployment. Alternatively, an attacker could exploit ancillary systems such as HMI consoles, historians, or VPN servers that interface with OT networks, manipulating sensor inputs to inject false data into controllers upstream of the DPS.

Defenses against these threats are still needed.[11] ICS frameworks like NIST 800-82 and IEC 62443 provide layers of controls for these scenarios across access control, patch management, network DMZs, etc. We do not propose abandoning them. Rather, we propose that machine-command legitimacy is the gap those frameworks leave open. By filling that gap with deterministic boundary enforcement, we reduce the cumulative risk.

In summary, DPSs complement probabilistic security mechanisms by removing the need to detect malicious actuator commands, which the DPSs prevent by design. This allows IDS and anomaly detectors to focus on higher-level concerns such as data anomalies, configuration changes, unauthorized devices, etc. The result is a more resilient security posture in which deterministic enforcement eliminates entire classes of attacks, while probabilistic defenses address the remaining threats, including lateral movement, insider misuse, and attempts to alter the DPS configuration.

COMPARISON OF ENFORCEMENT APPROACHES

The table below (Table 1) compares several classes of control-path enforcement for OT/shipboard systems. Each row highlights the point of enforcement, how it can fail, its attack surface, and its suitability for control system contexts.

Table 1. Comparison of enforcement approaches for OT/ICS. A DPS enforces at the point of action, unlike network-based controls, and thus dramatically reduces exploitable paths.

Approach	Enforcement	Failure Modes	Attack Surface	OT Suitability
Segmentation (Purdue Zoning)	Network or VLAN boundaries, firewalls/routers isolate zones.	Misconfiguration or hardware compromise allows lateral movement; does not vet commands themselves.	Network devices (firewalls, switches); inter-zone links.	Widely used and recommended for ICS, but if internal traffic is trusted, an adversary can bypass.

Approach	Enforcement	Failure Modes	Attack Surface	OT Suitability
Monitoring/IDS	Network taps, host agents observe traffic/behavior.	Passive; cannot prevent a command from executing. Alerts only after fact. May miss slow or encrypted attacks.	IDS sensors, logs, management console interfaces.	Standard practice; improves situational awareness but produces alerts only post-execution.
Software policy (whitelisting, RBAC)	Controller/OS level (authentication, code-signing, input validation).	Relies on correct software and CPU; vulnerable to zero-days or privileged exploits. Mis-emulation possible.	Host operating system, libraries, network stack (e.g., Windows / SCADA OS vulnerabilities).	Common in OT (auth servers, patch policies) but often impractical on legacy devices and suffers platform flaws.
DPS (Deterministic Protective System)	Execution boundary itself (fixed logic in hardware or microcode).	Only if logic is flawed or attacker breaches the protected path. Otherwise fails safe (drops commands).	Minimal, essentially the (protected) reconfiguration interface. Input channels cannot subvert it.	Highly suitable for critical OT, enforces safety in real time; assumes physical security of DPS.

CONCLUSIONS

Naval control systems require guarantees that every executed command is valid in its physical context, a guarantee that current security models do not explicitly provide. Conventional controls of user authentication, segmentation, monitoring focus on preventing unauthorized commands from entering the system, but they stop short of ensuring that a received command should ever be carried out by the actuators. This gap is execution legitimacy the requirement that no command ever "slips through" that would put the system in an unsafe or unintended state.

We argue that this missing control must be enforced deterministically at the execution boundary. A Deterministic Protective System (DPS) embodies this idea. By rejecting any command not in a pre-defined safe set, the DPS ensures that only appropriate signals reach machinery. In effect, DPS separates "who can talk to the controller" (handled by existing frameworks) from "what the controller is allowed to do" (handled by the DPS).

The insights offered here are grounded in both theory and practice.

The IEC/ISA 62443 series and NIST 800-82 describe layered defenses (users, networks, systems) but do not prescribe a deterministic boundary check. Incidents like Stuxnet show that until we add one, attackers can always craft apparently valid commands that cause damage. By making the execution boundary the enforcement point, we fill the gap without disrupting the rest of the architecture.

In summary, execution legitimacy is the new primitive at the instant of action, the system must ask, "Is this command truly allowed by design, policy, and safety constraints?" Deterministic enforcement (DPS) at the boundary provides this decision mechanism. Operating in fixed logic, with no reliance on post-hoc detection or complex software. Naval systems especially benefit from this approach, as they demand high assurance over long lifetimes. By hardening the execution boundary, we reduce the liability on other security

layers and ensure that even a sophisticated cyber-attack cannot directly produce unsafe behavior.

DPS effectively eliminates a wide class of ICS attacks. Future work will quantify how DPS requirements map onto specific naval threat models and explore the integration with existing command architectures.

Enforcing machine-execution legitimacy at the execution boundary is the missing control primitive in naval cyber defense. When this primitive is in place, deterministic protective systems provide the assurance that conventional methods cannot, guaranteeing that only approved commands ever drive critical equipment.

AUTHOR BIOGRAPHIES

Michael A. Strauss is the Chief Technology Officer for Blueskytec America Inc., where he has more than 25 years of leadership in advanced technology research and delivery of products for commercial and defense applications.

Clinton Groves is the Chief Executive Officer for Blueskytec America Inc., where he brings more than 25 years operational and product leadership in innovation and technology solutions to commercial and defense applications.

Disclosure: Both authors are employed by Blueskytec America, Inc., which develops and commercializes Deterministic Protection Systems (DPS) technology. The views presented in this paper reflect the authors' professional judgment and are informed by their work in the development of DPS.

REFERENCES

[1] Cybersecurity and Infrastructure Security Agency. (2026, February 10). Barriers to secure OT communication: Why Johnny can't authenticate. https://www.cisa.gov/resources-tools/resources/barriers-secure-ot-communication-why-johnny-cant-authenticate

[2] International Electrotechnical Commission. (n.d.). *IEC 62443 series: Security for industrial automation and control systems*. IEC.

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

[4] Williams, T. J. (1994). The Purdue enterprise reference architecture. *Computers in Industry*, *24*(2), 141–158.

[5] MITRE Corporation. (2025). *ATT&CK for ICS*. https://attack.mitre.org/techniques/ics/

[6] Mueller, P., & Yadegari, B. (2012). The Stuxnet worm. University of Arizona, Department of Computer Science. https://www2.cs.arizona.edu/~collberg/Teaching/466-566/2012/Resources/presentations/topic9-final/report.pdf

[7] Why Is Hardware More Secure than Software? (2024, May 15). *Cyber Defense Magazine*. https://www.cyberdefensemagazine.com/why-is-hardware-more-secure-than-software/

[8] Syed, N.F., Shah, S.W., Shaghaghi, A., Anwar, A., Baig, Z.A., & Doss, R.R. (2022). Zero Trust Architecture (ZTA): A Comprehensive Survey. IEEE Access, 10, 57143-57179. https://doi.org/10.1109/ACCESS.2022.3174679

[9] Karch, B., Gray, T.A., & Rowland, M. (2024). Field Programmable Gate Array-Based Reactor Protection Systems and Potential for Inclusion of Secure Elements to Improve Cybersecurity.

[10] Sandy Macadam (2021); Hardsec: Practical Non-Turing-Machine Security for Threat Elimination; https://www.hardsec.com

[11] Zhu, B., Joseph, A.D., & Sastry, S. (2011). A Taxonomy of Cyber Attacks on SCADA Systems. *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, 380-388.