FIREBALL INDUSTRIES

IGNITE YOUR FACTORY EFFICIENCY

WHITEPAPER

The Purdue Model is Dead

EmberNet and the zero-trust imperative for industrial operations.

By Eric Seme | Fireball Industries | October 2026



EXECUTIVE SUMMARY

The Purdue Model is Dead: EmberNet and the Zero-Trust Imperative for Industrial Operations

For over three decades, the industrial world has operated under a dangerous delusion, clinging to the Purdue Enterprise Reference Architecture as a security framework.[2] This report declares, without reservation, that this practice is no longer a conservative engineering choice but an act of profound professional malpractice. The Purdue Model, a relic from an era of isolated systems, is fundamentally broken, and its continued application in today's hyper-connected landscape is a direct and demonstrable threat to critical infrastructure, corporate solvency, and public safety. Adherence to this obsolete model is the equivalent of a structural engineer insisting on using 19th-century building codes in a modern earthquake zone; it is a willful disregard for evidence that guarantees catastrophic failure. The litany of devastating cyberattacks on industrial targets is not a series of unfortunate events but a predictable pattern of adversaries exploiting the architectural negligence inherent in the Purdue paradigm.

The era of perimeter-based security and implicit trust is over. The only professionally responsible path forward is a complete and uncompromising shift to a Zero Trust architecture. This is not an optional upgrade; it is an existential imperative. A Zero Trust model, founded on the principle of "never trust, always verify," accepts the stark reality that threats can and will exist everywhere, both inside and outside the network. It is the only strategy that provides meaningful resilience against the sophisticated, persistent threats targeting industrial operations today.

This report presents **EmberNet** as the reality-based alternative to the failed Purdue paradigm. EmberNet is a complete, hyper-converged infrastructure (HCI) platform engineered from the ground up for the unique demands of industrial operations. It is not an incremental improvement but a wholesale replacement, providing a secure, resilient, and agile foundation built on the core tenets of Zero Trust. The EmberNet platform consists of four integrated components: **EmberRTOS**, a tamper-proof immutable operating system that makes malware persistence a physical impossibility; **EmberNet Enrollment**, which automates secure hardware enrollment and configuration to eliminate human error; **EmberFLUX**, a Zero Trust Networking fabric that provides granular microsegmentation to contain any breach instantly; and **EmberNet's k3s orchestration**, a self-healing container orchestration engine that ensures operational continuity. This is an urgent call to action. The choice is simple: continue signing the suicide pact of the Purdue Model and wait for the inevitable disaster, or embrace the Zero Trust imperative with EmberNet and choose survival.

New in this edition: Part VI, measured proof from EmberNet; Part VII, how enrollment and zero-trust access work; Part VIII, how a migration runs with Fireball Industries; and a methodology appendix.

INTRODUCTION

The Emperor Has No Clothes

For thirty years, the Purdue Model has been the revered icon of industrial network design, its layered diagram hanging on the walls of engineering departments like a sacred text. It has served as the foundation for cybersecurity standards that are, themselves, dangerously out of date. We are here to state, unequivocally, that the emperor has no clothes. The Purdue Model is dead. Its continued use as a security framework is the architectural equivalent of offering "thoughts and prayers" in the face of a category five hurricane, a well-intentioned but utterly powerless gesture against an overwhelming force. Any CISO, plant manager, or board member who continues to sanction its use is presiding over a ticking time bomb, and when it detonates, ignorance will not be a viable defense.

The model's genesis in the 1990s was innocent enough. It was an industrial engineering framework for organizing data flows in computer-integrated manufacturing, a concept from an era when the internet was a novelty and "cyber warfare" was the stuff of science fiction. Security practitioners, desperate for any kind of blueprint, later latched onto its hierarchical structure. They saw its neat layers and theoretical separation between Information Technology (IT) and Operational Technology (OT) as a convenient way to build a digital fortress. The problem is that the world has changed, the nature of warfare has changed, and the fortress they built is made of cardboard.

The core premise of the Purdue Model (that you can build a trusted, safe "inside" and an untrusted, dangerous "outside") is a dangerous fantasy in the year 2026. The relentless drive for efficiency, remote access, cloud analytics, and the Industrial Internet of Things (IIoT) has not just blurred the lines between IT and OT; it has obliterated them. There is no "inside" anymore. Your network is a porous, dynamic, and constantly changing ecosystem of interconnected devices, many of which were never designed with security in mind. To assume anything within this ecosystem is "trusted" is an act of willful delusion that borders on the criminal.

This report will make the case for professional malpractice. We will demonstrate that the Purdue Model is not merely "showing its age" but is architecturally negligent. We will dissect its fatal flaws, not as theoretical weaknesses, but as the proven attack vectors used to hold national infrastructure hostage, shut down global food supplies, and attempt to poison municipal water systems. We will show how its rigid, brittle structure is fundamentally incompatible with the demands of modern industry and how its core assumptions create a highly flammable core, maximizing the blast radius of any successful breach.

Then, we will present the only viable alternative: a complete and total embrace of a Zero Trust security posture, implemented on a modern, hyper-converged foundation. This is the model the Department of Defense is adopting to protect its most critical assets, from nuclear silos to global logistics networks. If the Purdue Model isn't good enough for the DoD, why in God's name do you think it's good enough for your power plant, your factory, or your water utility? The time for polite debate and incremental change is over. It is time to choose a side: architectural negligence or reality-based security. It is time to introduce **EmberNet**.

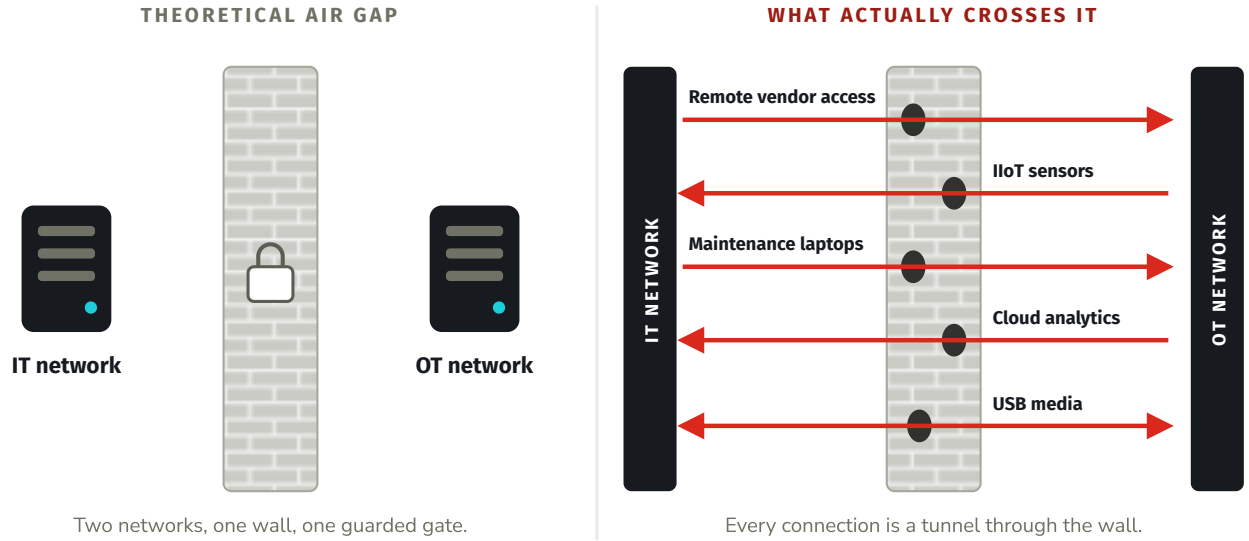
PART I

The Purdue Model: Architectural Negligence

To fully grasp why the continued use of the Purdue Model constitutes professional malpractice, one must look beyond its familiar diagrams and confront the world it was born into. Conceived in the early 1990s by Theodore J. Williams and a university consortium, the Purdue Enterprise Reference Architecture (PERA) was an academic model for organizing data flows in a manufacturing environment.[2][21] It was created before the commercial internet was a household utility, before the rise of global ransomware gangs, and before the concept of a persistent, motivated, and well-funded state-sponsored cyber adversary was a daily reality for every organization on the planet. Its adoption as a security framework was an accident of history, a convenient but ultimately disastrous repurposing of an industrial engineering diagram. The model's neat, hierarchical layers (from the physical process at Level 0 to the enterprise network at Level 5) provided a simple, visual map that was easy for engineers and managers to understand.[1] It gave them a false sense of order and control. It allowed them to believe they could build a digital fortress with a deep moat, high walls, and a single, heavily guarded gate. This belief is now the single greatest threat to industrial operations worldwide. The model that was once a blueprint for order has become a predictable roadmap for attackers, and its foundational principles are now its fatal flaws.

Fatal Flaws

The Purdue Model is built on a foundation of lies. These are not harmless white lies; they are dangerous, foundational delusions that create a security posture so fragile it shatters on first contact with a modern adversary. To continue building upon this foundation is to knowingly and willingly accept catastrophic risk, an act that flies in the face of any reasonable standard of professional care.



Two networks, one wall, one guarded gate.

Every connection is a tunnel through the wall.

Figure 1. The air gap myth in modern OT. The wall on paper, and the paths that cross it in a working plant.

The first and most pervasive lie is the **air gap myth**. The concept of a physical or logical "air gap" separating the pristine OT network from the chaotic IT network is a **fairy tale told by managers** to comfort themselves at night and to placate auditors. It is a security blanket woven from wishful thinking and corporate denial. In the real world of 2026, true air gaps do not exist in any competitive industrial enterprise. The demands of digital transformation (for real-time data analytics, predictive maintenance, remote vendor access, and cloud-based operational dashboards) have riddled this theoretical wall with more holes than a block of Swiss cheese. Every connection, from a maintenance laptop with a cellular modem to an IIoT sensor sending data directly to a cloud platform, is a tunnel that completely bypasses the Purdue Model's neat, hierarchical chokepoints. Every USB drive carried onto the plant floor is a potential Trojan horse, as the architects of Stuxnet so brilliantly demonstrated over a decade and a half ago.[19] Believing you have an air gap is like believing in Santa Claus; it's a charming fantasy for children, but a dangerous, career-ending delusion for a professional responsible for critical infrastructure.

The second fatal flaw is the model's creation of a **highly flammable core**. The Purdue Model operates on a "castle-and-moat" philosophy: a hard, crunchy perimeter with a soft, chewy, and trusting center. It implicitly trusts everything and everyone already inside the OT network. Once an attacker crosses the IT/OT boundary, a boundary we've established is a myth, they find themselves in a flat, unsegmented, and trusting environment. This is not a security zone; it is a blast furnace. Every legacy PLC, every unpatched Windows XP-based HMI, every insecure industrial protocol is kindling, waiting for a spark. In this environment, lateral movement is not just possible; it is trivial. The entire operational environment becomes the blast radius of a single breach. The NotPetya incident in 2017 was a perfect demonstration of this principle.[18] Malware that entered the IT network spread like wildfire into unprepared OT environments, not because the perimeter was weak, but because the core was a tinderbox of implicit trust. The Purdue Model doesn't contain threats; it incubates them until they consume the entire operation in a digital inferno.

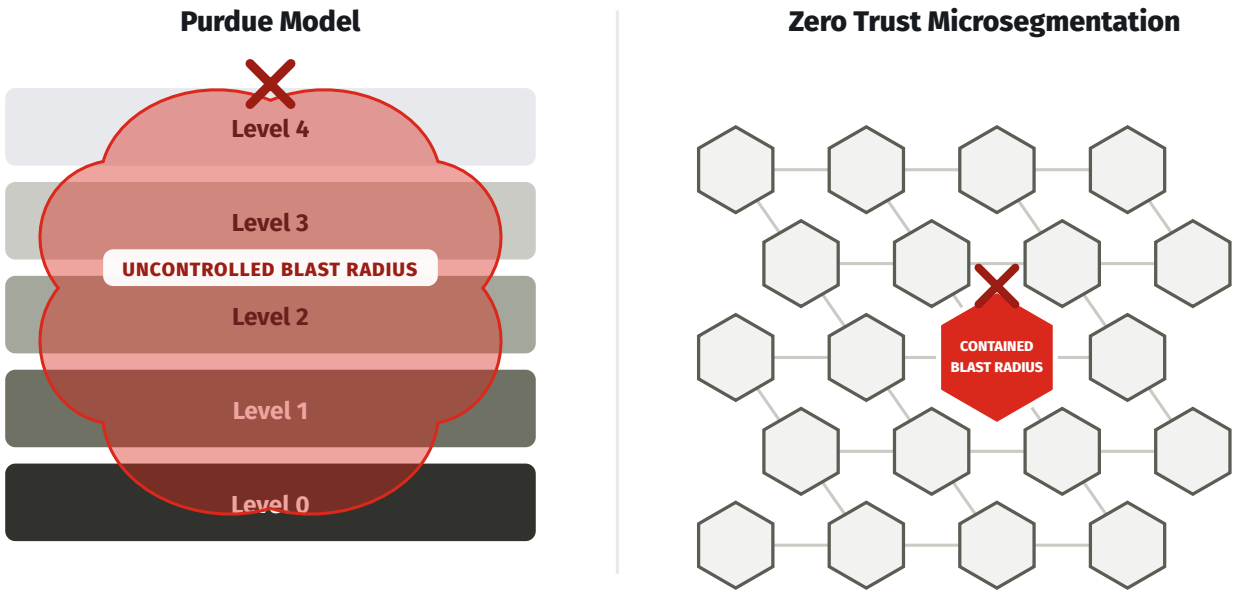


Figure 2. One breach, two architectures. In the Purdue stack it spreads through every level; in a microsegmented network it stays in one segment.

Third, the reliance on Virtual Private Networks (VPNs) for remote access has created a massive, indefensible **VPN attack surface**. VPNs, the traditional tool for bridging the IT/OT divide for remote work, are nothing short of an engraved **invitation for state-sponsored actors** and ransomware gangs. A VPN, by its very nature, extends the trusted corporate network directly into the heart of the industrial control environment. It grants broad, network-level access, effectively connecting an employee's potentially compromised home network to the PLC that controls a chemical reactor. When an attacker compromises a user's VPN credentials through a simple phishing email or buys them on the dark web, they don't just get access to a single application; they get a foothold on the entire network. They are inside the castle walls, free to move laterally and discover the vulnerable systems that the Purdue Model so conveniently groups together. The Colonial Pipeline attack, which brought the East Coast's fuel supply to its knees, was initiated through a compromised VPN account.[13] It wasn't a sophisticated zero-day exploit; it was an attacker walking through the front door using a key that was left under the mat. Relying on VPNs for OT access is not a calculated risk; it is a security death wish.

Finally, the Purdue Model imposes a **rigid, brittle architecture** that is fundamentally incompatible with the agility required for modern industry. The model's strict, top-down, "north-south" data flow is an artifact of 1990s thinking. Today's operations require dynamic, "east-west" communication. A sensor on one production line needs to share data with a control system on another. An analytics platform at the edge needs to communicate directly with a cloud service. The Purdue Model forbids this. It forces all communication through slow, cumbersome, and often insecure vertical pathways, hindering innovation and creating a false sense of security. When engineers, faced with impossible architectural constraints, inevitably create workarounds to get their jobs done, they introduce unsanctioned and unmonitored

communication paths that completely subvert the model's intended security. The architecture is not just insecure; it is an active impediment to business progress. It is a framework that forces a choice between being secure and being competitive, and it fails at delivering security anyway.

Brutal Real-World Examples

The architectural negligence of the Purdue Model is not a theoretical concern. It is a documented reality, written in the headlines of global news and the catastrophic losses on corporate balance sheets. These are not isolated incidents; they are the predictable, inevitable consequences of a failed security paradigm that has been allowed to persist for far too long.

Consider the May 2021 ransomware attack on **Colonial Pipeline**. This was not just a ransomware incident; it was a **hostage situation** involving the critical fuel supply for the American East Coast. Attackers gained entry through a single compromised VPN account that lacked multi-factor authentication, a direct indictment of the Purdue Model's reliance on a fragile perimeter.[13][14] Once inside the IT network, they moved laterally, and the company was forced to proactively shut down the entire 5,500-mile pipeline.[13] Why? Not because the OT control systems themselves were compromised, but out of sheer terror that the malware could cross the poorly segmented IT/OT boundary. The billing systems on the IT side were encrypted, and without them, the company could not effectively track fuel distribution. The rigid coupling of IT and OT, a hallmark of poorly implemented Purdue-style architectures, meant that a failure in the "untrusted" zone caused total **operational paralysis** in the "trusted" one. The result was panic buying, fuel shortages, a reported $4.4 million ransom payment, and a national security crisis.[13][15] This was a direct, foreseeable failure of architectural design.

Look at **JBS Foods**, the world's largest meat supplier, which was forced to halt operations across North America and Australia after a ransomware attack in the same year. The attack crippled their production, leading to immediate supply chain disruptions and the threat of food shortages. The company paid an $11 million ransom to regain control,[16] a sum that pales in comparison to the losses from operational paralysis and the permanent damage to its reputation. Again, the pattern was the same: an intrusion into the corporate network that bled into the operational side, demonstrating a catastrophic failure of segmentation and containment, the very things the Purdue Model purports to provide. The "chewy center" of their operations was devoured because the "crunchy outside" was breached.

Or recall the chilling 2021 incident in **Oldsmar, Florida**, where an attacker remotely accessed a water treatment plant's control system and attempted to increase the level of sodium hydroxide (lye) in the water supply to poisonous levels. The only thing that prevented a mass poisoning event was a single, alert operator who noticed the mouse cursor moving on its own and manually reversed the change. The attacker gained access through insecure remote access software, bypassing whatever perimeter defenses were in place and directly manipulating systems at Level 2 of the Purdue hierarchy.[17] This is the ultimate nightmare scenario: a digital intrusion causing direct physical harm to a civilian population. It is

the starkest possible proof that relying on perimeter segmentation while leaving control systems exposed is an abdication of professional responsibility. In 2023, Oldsmar's former city manager said the incident was employee error, and the FBI said it found no evidence of a cyberattack.[36]

These are not edge cases. The 2017 NotPetya attack caused over $10 billion in damages globally, shutting down production for giants like the shipping company Maersk and the pharmaceutical manufacturer Merck[18] because their networks were flat, with no meaningful separation between IT and OT. The Stuxnet worm, discovered in 2010, masterfully navigated the layers of a supposedly air-gapped facility to physically destroy centrifuges.[19][20] It proved, more than fifteen years ago, that the Purdue Model's core assumptions were flawed. The fact that we are still having this conversation today is a testament to the dangerous inertia within the industrial sector. These events are not warnings; they are verdicts. They are the brutal judgment passed on an obsolete architecture. The financial ruin is real. The operational paralysis is real. The threat to public safety is real.

Why Continuing is Malpractice

Let us be perfectly clear. In light of these repeated, catastrophic failures, continuing to advocate for, design, or operate an industrial network based on the Purdue Model is professional malpractice. It is a willful disregard for established evidence and a conscious acceptance of unacceptable risk. It is a breach of the duty of care owed to shareholders, customers, and the public.

Imagine a **structural engineer** designing a skyscraper in San Francisco today using building codes from 1920. They ignore decades of research on seismic activity, reinforced concrete, and base isolation. They insist that the old ways are sufficient because that's how they've always done it. When the earthquake hits and the building collapses, killing hundreds, would we call it an unfortunate accident? Or would we call it gross negligence? Would the engineer be allowed to continue practicing? Of course not. They would be stripped of their license, face civil lawsuits, and likely criminal charges. There is no difference here. The threat landscape has changed. The technology has changed. The evidence of failure is overwhelming. To ignore it is to be complicit in the resulting disaster.

This is no longer a technical issue for the engineering department; it is a **board-level liability**. In the wake of Colonial Pipeline, the SEC and other regulatory bodies are making it clear that cybersecurity is a matter of corporate governance and risk management.[22] When your company is on the front page for causing a national fuel shortage or a food supply crisis, the board of directors will be held accountable. They will ask the CISO, the CIO, and the VP of Operations a simple question: "What were you doing to prevent this?" Answering "We were following a security model from the 1990s that has been repeatedly proven to fail" is not a defense. It is a confession. It is an admission of negligence that will open the floodgates to shareholder lawsuits, regulatory fines, and personal liability.

Continuing to follow the Purdue Model is a **suicide pact**. It is an implicit agreement between you and your adversary that you will build your defenses in a predictable, easily circumvented way, and they will exploit them at their leisure. You are handing them the architectural blueprints to your own destruction,

gift-wrapped. It is a guarantee of failure, signed by every executive who refuses to acknowledge the reality of the modern threat environment.

The final, damning piece of evidence comes from the most security-conscious organization on the planet: the United States Department of Defense. The DoD is actively moving its critical infrastructure and control systems away from perimeter-based models like Purdue and toward a comprehensive Zero Trust Architecture. Their official guidance explicitly states that Zero Trust principles must be applied to Operational Technology.[9]

> **If the Purdue Model is not good enough to protect nuclear silos, fighter jets, and global military logistics, why on earth do you think it is good enough to protect your power grid, your manufacturing plant, or your city's water supply?**

The DoD has accepted reality. It is time for the industrial world to do the same. The Purdue Model is dead. Mourning is optional; moving on is mandatory.

PART II

Zero Trust: Not Optional, Essential

PURDUE MODEL	ZERO TRUST WITH MICROSEGMENTATION
Perimeter-based security	Identity-based, resource-centric security
Implicit trust on the internal network	Never trust, always verify
Flat or macro-segmented zones	Fine-grained microsegmentation down to individual workloads
Large lateral movement and blast radius	Tightly contained blast radius
VPN-based remote access	Zero-trust remote access, per app and per device
Brittle and rigid architecture	Resilient and adaptable architecture

Figure 3. Purdue versus Zero Trust for industrial security.

The wholesale failure of the Purdue Model and its perimeter-based security philosophy leaves a terrifying void. If the castle-and-moat is a lie, what is the alternative? The answer is not to build higher walls or a wider moat. The answer is to abandon the fortress entirely and adopt a new paradigm: **Zero Trust Architecture (ZTA)**. This is not a product, a feature, or a buzzword. It is a fundamental, non-negotiable shift in security strategy, and it is the only viable path forward for any organization serious about protecting its industrial operations. The core principle of Zero Trust is as simple as it is profound: **never trust, always verify**. It is the architectural embodiment of professional paranoia, and in the current threat environment, paranoia is just another word for realism.

Zero Trust obliterates the archaic and dangerous distinction between a "trusted" internal network and an "untrusted" external one. It assumes that the network is always hostile. It assumes that attackers are already inside. It assumes that every user, every device, and every application is a potential threat until proven otherwise, every single time they request access. This is not cynicism; it is realism. In a world of IT/OT convergence, ubiquitous remote access, and compromised supply chains, it is the only sane assumption to make. Security is no longer about defending a perimeter; it is about protecting each individual resource, making data and critical assets the new, defensible boundary.

This philosophy is built on three non-negotiable pillars. The first is **continuous verification**. Identity is not a one-time password. It is a dynamic, contextual assessment that includes not just the user's credentials, but the health of their device, their location, the time of day, and the specific resource they are trying to access. Access is granted on a per-session basis and is constantly re-evaluated. If any aspect of that context changes or becomes suspicious, access can be revoked in real time. The second pillar is the ruthless enforcement of **least-privilege access**. This dictates that any user, device, or application is granted only the absolute minimum set of permissions required to perform a specific, authorized task, for the shortest possible time. A maintenance engineer needing to update a single PLC is not given broad network access; they are given a secure, temporary, and fully logged tunnel to that one device and

nothing else. This dramatically shrinks the attack surface and minimizes the potential damage from a compromised account. The third core pillar is to **assume breach**. This shifts the security posture from a fragile, prevention-only model to one that combines prevention with aggressive detection and rapid response. It acknowledges the reality that breaches are not a matter of *if*, but *when*, and it designs the architecture to contain and neutralize them instantly.

The endorsement for this model comes from the highest levels of the cybersecurity and defense communities. The **Carnegie Mellon University Software Engineering Institute (CMU SEI)**, a federally funded research and development center, has examined applying Zero Trust to industrial control systems and holds that zero-trust principles matter for ICS, while laying out the obstacles: legacy devices that are hard to upgrade and were never designed with security in mind, a low tolerance for communication interruptions, shared accounts, and too little collaboration between IT and OT teams.[6] The **U.S. Department of Defense** has made Zero Trust a strategic imperative, issuing a comprehensive strategy and roadmap for its implementation across all its networks, including the sensitive OT systems that control its most critical assets.[9][23] These organizations are not chasing trends; they are responding to the stark reality of the modern threat landscape.

A foundational technology for implementing Zero Trust in an OT environment is **microsegmentation**. If the Purdue Model's macro-segmentation is like building a few large, easily breached fire zones in a building, microsegmentation is like installing an advanced **fire suppression system** in every single room. It is a security technique that divides the network into tiny, granular, and isolated zones, often down to the level of a single PLC, HMI, or even an individual workload. By default, all communication between these microsegments is denied. Only explicitly allowed, legitimate traffic based on a strict, least-privilege policy is permitted to pass. This is the ultimate containment strategy. If a device within one microsegment is compromised by ransomware or a malicious actor, the breach is trapped. The fire is contained to a single room. The attacker cannot move laterally to other parts of the plant floor, cannot scan for other vulnerable systems, and cannot reach their ultimate target. Microsegmentation transforms the highly flammable core of the Purdue Model into a series of fireproof, blast-proof vaults. It is the practical application of the "assume breach" principle, ensuring that when one component fails, it does not trigger a catastrophic, cascading failure of the entire operation. For modern OT, Zero Trust is not optional; it is the essential, logical, and only professionally defensible successor to the failed paradigm of the past.

PART III

Hyper-Converged Infrastructure: The Modern Foundation

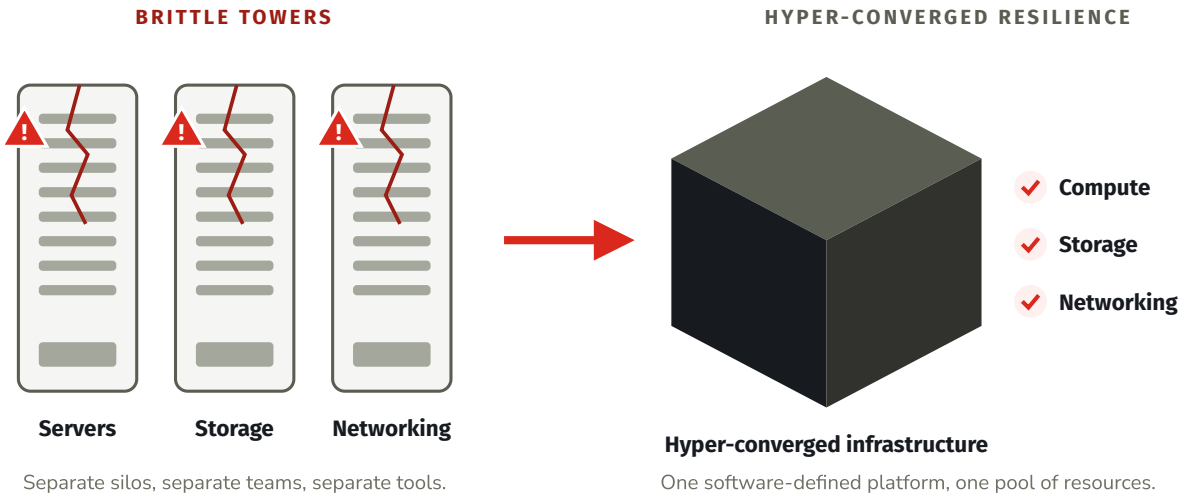


Figure 4. From brittle towers to hyper-converged resilience.

The architectural negligence of the Purdue Model extends beyond its flawed security concepts; its very physical and logical structure is a relic, fundamentally unsuited for the demands of modern industrial operations. The traditional three-tier architecture it encourages (separate, siloed towers of compute, storage, and networking hardware, each managed by different teams with different tools) is the definition of rigid and brittle. This distributed chaos is not just an operational headache; it is a security liability. It creates seams, gaps, and inconsistent policies for attackers to exploit, and makes rapid, unified response to an incident nearly impossible. To build a resilient, Zero Trust architecture, you must first replace this crumbling foundation with a modern one: **Hyper-Converged Infrastructure (HCI)**.

HCI is a radical simplification and consolidation of the industrial stack. It is a software-defined approach that collapses the entire infrastructure (compute, storage, and networking) into a single, unified platform running on cost-effective, commercial off-the-shelf servers. This is not merely bundling hardware together; it is a deep, software-based integration where all resources are virtualized and managed as a single, cohesive pool. This approach directly solves the brittleness of the Purdue-era architecture. Instead of a collection of disparate, single-purpose servers, each a potential point of failure, you have a single, resilient cluster. Data and workloads are distributed across all nodes. If a hardware node fails, its workloads are automatically restarted on healthy nodes elsewhere in the cluster. There is no single point of failure. This inherent resilience is the bedrock of operational continuity, a concept alien to the fragile, hierarchical dependencies of the Purdue Model.

Layered on top of this resilient foundation is the next critical component for reality-based security: the **immutable operating system**. Traditional operating systems are mutable; their core files and configurations can be changed while the system is running. This makes them a playground for malware. An attacker who gains access can modify system files, install rootkits, and establish persistence that is

incredibly difficult to detect and remove. An immutable OS eliminates this entire class of threats. Its core principle is that the base operating system, once deployed, is read-only and cannot be modified. Updates are not applied via risky, one-off patches; the entire OS image is replaced atomically with a new, verified version. If the update causes a problem, the system can be instantly rolled back to the previous known-good state.[10][11] This ensures predictable, deterministic behavior across the entire fleet, a non-negotiable requirement for industrial control.

The final piece of this modern foundation is a powerful **container orchestration** engine. In a distributed industrial environment with hundreds or thousands of devices at the edge, manual configuration is a recipe for disaster. It leads to configuration drift, inconsistent security policies, and human error. A modern orchestration platform automates the deployment, management, and lifecycle of all industrial applications. It is the central brain that ensures every node in the distributed infrastructure is running the correct software, the correct configuration, and the correct security policy. This is the difference between managing a thousand separate, vulnerable systems and commanding a single, unified, and resilient platform. The orchestration engine can automatically deploy updates, scale applications based on demand, and even automatically isolate and remediate a compromised node without human intervention. This combination of HCI, an immutable OS, and centralized orchestration creates a secure, resilient, and agile platform that is the polar opposite of the Purdue Model's brittle, static, and insecure architecture. It is the necessary foundation upon which a true Zero Trust security posture can be built.

PART IV

EmberNet: Reality-Based Industrial Security

The Purdue Model is a fantasy. It is an architecture built for a world that no longer exists. **EmberNet** is the reality-based alternative. It is not another layer of security paint slapped onto a crumbling wall. It is a complete, integrated Hyper-Converged Infrastructure (HCI) platform designed from first principles for the brutal realities of the modern industrial threat landscape. It is built on the Zero Trust imperative, founded on hyper-converged resilience, and engineered to provide the visibility, control, and containment that the Purdue Model so catastrophically fails to deliver. EmberNet is the comprehensive solution for organizations ready to abandon architectural negligence and adopt a reality-based security posture. Its architecture is composed of four tightly integrated components, each designed to address specific weaknesses of traditional industrial systems and build upon the layer below it to create a holistic, defense-in-depth environment.

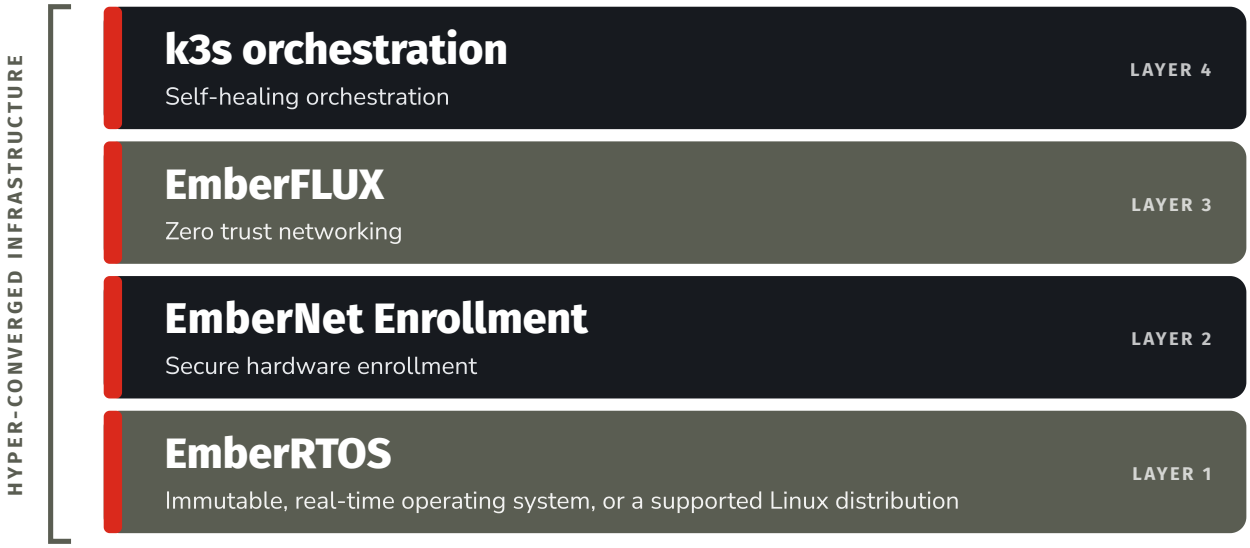


Figure 5. The EmberNet platform stack. Each layer builds on the one below it.

Platform Architecture

The EmberNet platform is a complete stack that replaces the fragmented and insecure hardware and software landscape typical of Purdue-based environments. It provides a single, unified solution that is secure by design and built for the unique demands of OT.

The foundation of the platform is **EmberRTOS** or a supported Linux distribution, with EmberRTOS as the immutable, real-time option. EmberRTOS is not just another hardened Linux distribution; it is a purpose-built, immutable, real-time operating system designed for absolute security and stability. **If your OS is mutable, your security is a suggestion. Malware doesn't ask for permission to persist; it just writes to the disk. EmberRTOS's read-only filesystem makes persistence a physical impossibility.** Any attacker who gains temporary execution on an EmberRTOS node will find their malware, their configuration

changes, and their persistence mechanisms wiped clean upon the next reboot. This fundamental principle of immutability ensures the integrity of the entire system from the moment it boots, providing a tamper-proof foundation that eliminates an entire class of common attack vectors.

The next layer is **EmberNet Enrollment**, the secure hardware enrollment and configuration engine. In a traditional environment, provisioning new hardware is a manual, error-prone process that often introduces security vulnerabilities. EmberNet Enrollment automates this entire lifecycle. New hardware nodes are securely enrolled into the EmberNet cluster using a cryptographic identity process, ensuring that no rogue or unauthorized hardware can join the trusted fabric. All configuration is managed centrally and pushed declaratively to the nodes. This eliminates manual configuration on individual devices, preventing configuration drift and ensuring that every node in the fleet adheres to a consistent, verified, and secure baseline. EmberNet Enrollment transforms infrastructure management from distributed chaos into centralized, automated control.

The heart of EmberNet's security model is **EmberFLUX**, the Zero Trust Networking (ZTN) fabric. This is a revolutionary, software-defined networking layer that implements identity-based microsegmentation for every workload running on the platform. By default, no application, container, or device can communicate with any other. All communication must be explicitly authorized by a centrally managed policy that defines precisely which services can talk to each other, over which protocols, and on which ports. **EmberFLUX's microsegmentation is the fire suppression system that Purdue lacks.** It transforms the highly flammable core of a traditional OT network into a series of blast-proof, isolated vaults. Even if an attacker were to compromise a single application, they would be trapped within that microsegment, unable to move laterally, scan the network, or escalate their attack. EmberFLUX contains the fire instantly, reducing the blast radius from the entire plant to a single, manageable point.

Finally, **EmberNet's k3s orchestration** provides the resilient container orchestration that ensures the mission continues, no matter what. EmberNet's k3s orchestration is a purpose-built orchestration engine that manages the lifecycle of all industrial applications, which run as isolated, containerized workloads. It automates their deployment, scaling, and health management across the entire EmberNet cluster. This provides unprecedented resilience. **EmberNet's k3s orchestration is a fortress that heals itself. If a node dies, the mission continues. In Purdue, if the SCADA server dies, the plant goes dark.** With k3s orchestration, if a hardware node fails or is taken offline for maintenance, its critical application containers are automatically and instantly rescheduled onto healthy nodes in the cluster,[34] often with zero downtime. This self-healing capability ensures the high availability of critical industrial services, a level of resilience that is simply unattainable with the fragile, monolithic systems of the past.

How EmberNet Addresses Each Purdue Failure

Each component of the EmberNet stack is purposefully designed to solve a specific, critical failure of the Purdue Model, providing a direct and comprehensive solution to decades of architectural negligence.

Air gap myth → EmberNet accepts reality with zero trust.

EmberNet does not rely on the fantasy of a perfect perimeter. It assumes the network is hostile and builds security from the inside out. EmberFLUX's ZTN capabilities ensure that even if an attacker gets onto the physical network, they cannot move, communicate, or cause harm because every connection requires cryptographic identity and policy authorization.

Flat zones → EmberFLUX microsegmentation.

The dangerous, "chewy center" of the Purdue Model is eliminated. EmberFLUX creates granular, software-defined segments around every single workload. There are no flat zones. An attacker who compromises one asset is trapped in a digital cage, unable to move laterally to infect other systems.

Brittle architecture → k3s orchestration self-healing.

The single points of failure inherent in Purdue's rigid, server-per-application model are replaced by a resilient, self-healing cluster. EmberNet's k3s orchestration ensures that the failure of a single piece of hardware does not lead to a catastrophic operational shutdown. The mission continues.

Manual configuration → EmberNet Enrollment automation.

The chaos of manual server builds, inconsistent patching, and configuration drift is replaced by a centralized, automated, and declarative configuration management system. EmberNet Enrollment ensures every node in the fleet is identical, secure, and compliant, eliminating the vulnerabilities introduced by human error.

Mutable systems → EmberRTOS immutability.

On EmberRTOS nodes, the vulnerable, patch-and-pray model of traditional operating systems is replaced by a tamper-proof, read-only foundation. EmberRTOS makes malware persistence a physical impossibility, ensuring the integrity of the platform and providing a trusted base for all industrial applications.

PART V

The Migration Imperative

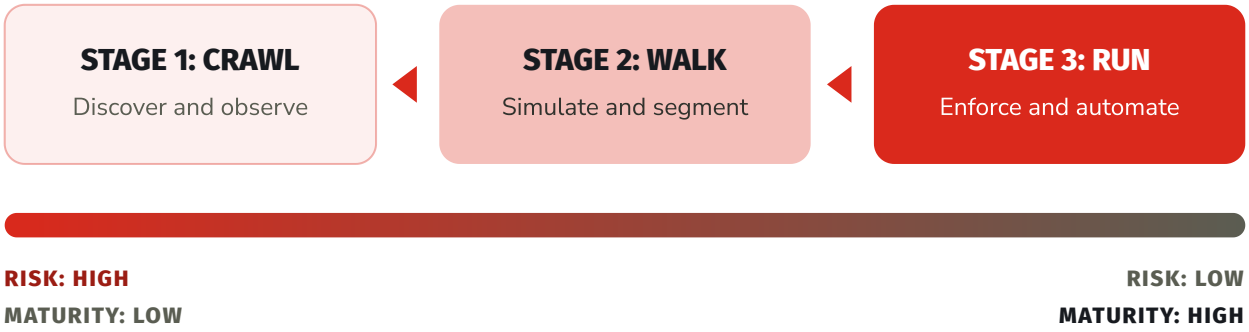


Figure 6. The zero trust migration roadmap for industrial operations.

The question is no longer *if* your Purdue-based architecture will be breached, but *when*, *how catastrophically*, and *how fast* you can migrate away from it to prevent that inevitability. The migration to a Zero Trust architecture with EmberNet is not an academic exercise or a long-term strategic goal. It is an urgent operational imperative, a race against an adversary who is already inside your network or actively probing your weak perimeter. Every day spent operating on a legacy architecture is another day you are willingly playing Russian roulette with your production lines, your balance sheet, and your professional reputation.

The **ROI of avoiding the next Colonial Pipeline** is the only metric that matters. The return on investment for this migration is not measured in incremental efficiency gains. It is measured in catastrophe avoidance. What is the value of preventing a complete shutdown of your global operations, avoiding a nine-figure ransomware payment, and staying off the front page of the Wall Street Journal? The cost of inaction is existential. The cost of migrating to EmberNet is a rounding error by comparison. This is not an expense; it is the most critical insurance policy your organization will ever purchase, and it is one that actually pays out by preventing the disaster in the first place.

The migration path is not a disruptive, rip-and-replace nightmare. It is a phased, logical, and non-disruptive process that begins with visibility. You cannot protect what you cannot see. The first step is to deploy EmberNet's read-only collectors and asset registry. The collectors make no writes, send no discovery broadcasts, and do no network scanning. Data leaves the plant outbound only, so no inbound port is opened. You see the floor without opening the network.

From this foundation of visibility, the **implementation roadmap** proceeds as a crawl, walk, run approach. You start by implementing EmberFLUX's microsegmentation in a monitoring-only mode. The platform will learn the normal communication patterns of your operation and automatically generate recommended, least-privilege policies. You can simulate the impact of these policies without enforcing

them, ensuring that security will not disrupt operations. Once validated, you move to enforcement, starting with your most critical assets. You place your Safety Instrumented Systems, your primary PLCs, and your SCADA servers into their own isolated microsegments using EmberNet Enrollment and EmberFLUX. With each step, you are shrinking the attack surface and building resilience. Next, you tackle the perimeter by replacing your high-risk VPNs with EmberNet's secure, identity-based access. Finally, you leverage EmberNet's k3s orchestration to begin containerizing and modernizing your industrial applications, deploying them on EmberRTOS or a supported Linux distribution, with EmberRTOS providing the secure, immutable foundation.

Success metrics are clear and tangible. A 99% reduction in the lateral movement attack surface. The containment of a simulated breach to a single device. The elimination of all VPN-based OT access. The ability to patch and update your entire distributed infrastructure in minutes, not months, using EmberNet Enrollment. The ultimate success metric is simple: when the next global ransomware campaign hits, your operations continue uninterrupted while your competitors, still clinging to their Purdue Model security blankets, grind to a halt. This is not a question of "if," but "when" and "how fast." The migration imperative is clear. The time to act was yesterday. The next best time is now.

PART VI

Proof from EmberNet

Everything up to here is the argument. This part is the evidence: what EmberNet has done on real hardware and on real projects, with the conditions stated so you can judge it for yourself.

EmberRTOS: the worst case, measured for a full day

Part IV calls EmberRTOS a real-time operating system. Real time is a promise about the worst case, not the average, so the worst case is the number we measured. With the real-time kernel switched on, a Raspberry Pi 5 (BCM2712) running EmberRTOS held a worst-case wake-up latency of **23 µs** over **86,423 seconds (24 hours)** and about **627 million samples**.[27]

23 µs
Worst-case wake-up latency across every thread over the 24 hour run.

627 M
Wake-ups measured, about 627 million samples in 86,423 seconds.

4 to 9 times
Inside the 100 to 200 µs band that industrial control loops target.

The measurement used `cyclictest` with three SCHED_FIFO priority 99 threads, waking at 200, 700, and 1200 µs intervals. Their worst cases were 17, 20, and 23 µs, and the average on every thread was 1 µs. Over the day the board ran from about 30 °C to about 60 °C with no throttling. Industrial control loops target a worst case somewhere in the 100 to 200 µs band, which puts 23 µs 4 to 9 times inside it.[27]

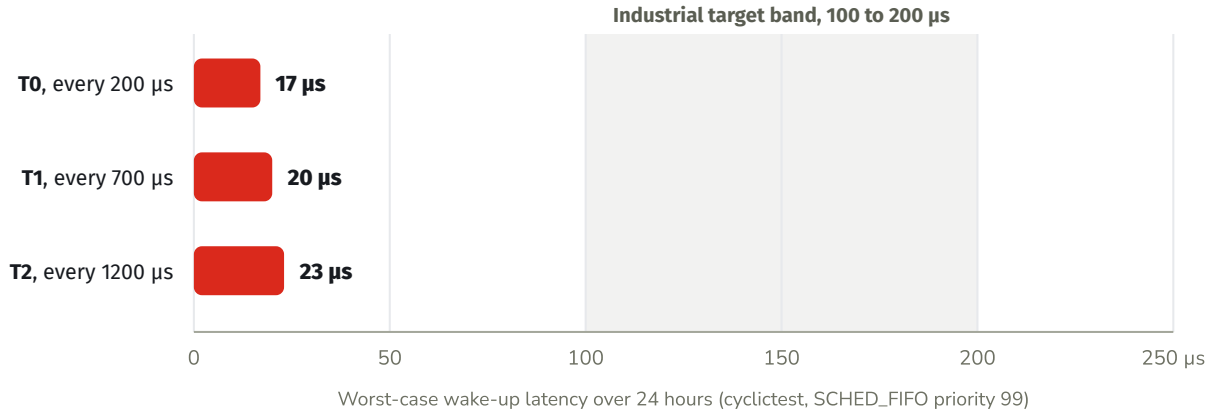


Figure 7. EmberRTOS on a Raspberry Pi 5 (BCM2712), real-time kernel on, 24 hour soak. Average latency was 1 µs on every thread. Source: EmberRTOS hardware results.[27]

WHAT THIS TEST IS, AND WHAT IT IS NOT

The stress load (hackbench, two CPU-burn loops, and a memory streamer) was pinned to the housekeeping core, so the real-time cores stayed isolated. It is not an all-core adversarial test, and the figures are for this board.

Remote PLC work with nothing exposed

Part II describes the maintenance engineer who gets a tunnel to one PLC and nothing else. On EmberNet, an engineer in Austin downloaded a PLC application to a virtual PLC in Cleveland, 1,184 miles away, with no VPN, no port forwarding, and nothing exposed on the plant network.

Austin to Cleveland

On September 29, 2026, an engineer in Austin, Texas downloaded the PLC application from the CODESYS IDE (CODESYS V3.5 SP22 Patch 2) to a virtual PLC (vPLC) in Cleveland, Ohio, over EmberNet.[28] The Cleveland node is an Industrial PC (arm64, running k3s) with CODESYS Control for Linux ARM64 SL 4.22 in a container, and the IDE reached it over the node's EmberNet overlay address on port 11740. There was no VPN, no port forwarding, and nothing exposed on the plant network. The project is deployed with the client. That is about 1,200 miles (1,184 miles straight-line).

Fireball Industries is a CODESYS System Partner.

Cleveland to Mumbai

An engineer in Cleveland, Ohio downloaded a CODESYS application to a virtual PLC at a plant in Mumbai, India, over EmberNet. That is about 7,900 miles (7,946 miles great-circle). The client is anonymous.

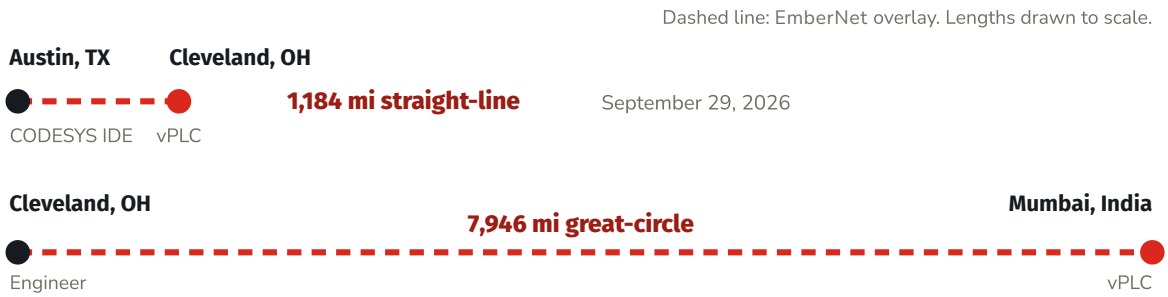


Figure 8. Two CODESYS downloads to virtual PLCs over EmberNet, with distances drawn to the same scale.

PART VII

How Enrollment and Zero-Trust Access Work

Here is how EmberNet puts the Part II pillars into practice, taken from the product documentation.

Identity, not address

Every node, service, and user on the mesh has a cryptographic identity, and access is granted between identities, not between IP addresses.[30] Routing follows identity too. That is the property that shuts down lateral movement: there is no address space to scan, because reachability is not a function of addressing. Connections are mutually authenticated with TLS, using certificates the platform issues when a node enrolls, rotates before they expire, and can revoke immediately if a host is compromised. Validity periods are deliberately short, which limits how useful a stolen certificate is.[29]

Outbound only, on port 443

Nodes dial out to the EmberFLUX fabric; nothing dials in. There is no inbound port to open at the plant, no port forward, and no public IP on your equipment. Everything rides port 443, so it passes through restrictive networks without special handling, works through carrier-grade NAT with no configuration, and runs over cellular, satellite, Wi-Fi, or Ethernet.[29] Traffic is encrypted end to end between identities across the whole path, not just to a gateway.[30]

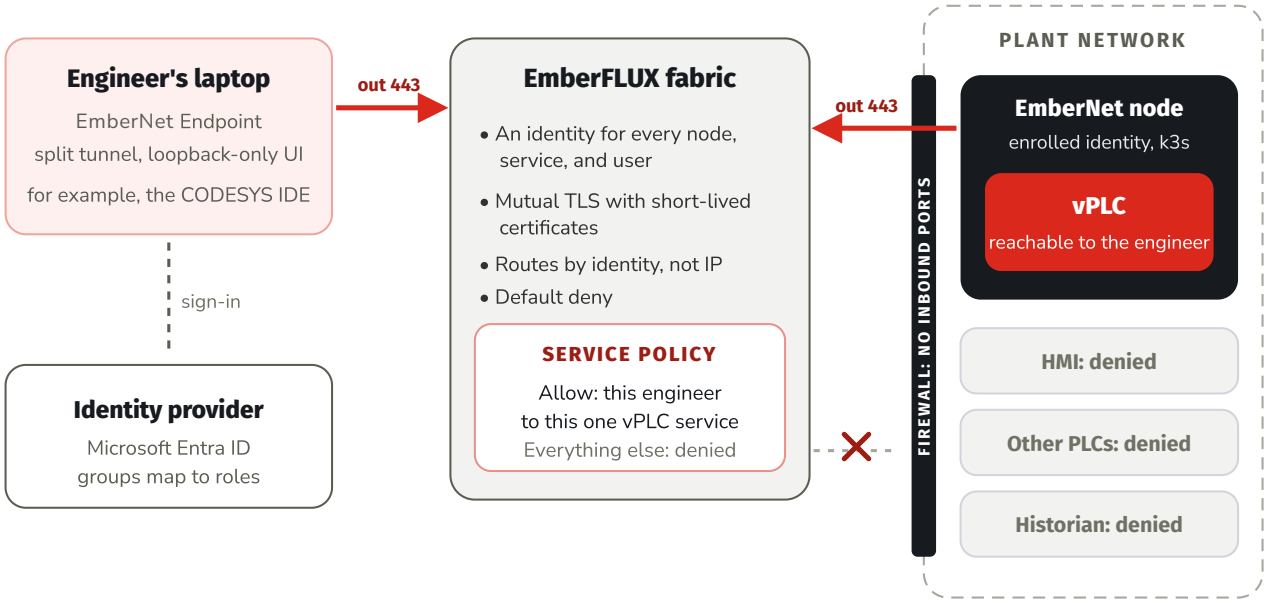


Figure 9. How an engineer reaches one PLC and nothing else. Both ends dial out on 443; the plant opens no inbound port; policy grants one service and denies the rest. The HMI, PLC, and historian boxes are illustrative.

How a node enrolls

Fireball Industries provisions nodes with EmberRTOS or a supported Linux distribution, with EmberRTOS as the immutable, real-time option.[31]

1. **Provisioning.** The node is imaged with EmberRTOS or a supported Linux distribution. On EmberRTOS, first boot applies the site's settings (hostname, networking, and keys) from a configuration medium.
2. **Cluster formation.** A single-node site is a cluster of one. A multi-node site joins additional nodes to the first.
3. **Platform services.** These come up as containerized workloads on the node, the mesh agent among them. There is no manual orchestration step.
4. **Mesh enrollment.** The node generates an identity, dials out to the mesh, receives its certificates and policy, and becomes reachable to authorized services. Nothing dials in.
5. **Tenant setup.** The tenant is created and bound to directory groups, so its users resolve to the right roles and scope.[31]

The mesh itself is managed from the Flux Console in EmberCORE. Creating an identity there issues an enrollment token that the endpoint uses to join, and posture checks can require an endpoint to meet conditions (operating system, process, MFA, and more) before it is trusted.[30]

How an engineer reaches one PLC, and nothing else

There is no separate EmberNet password. Sign-in is delegated to your identity provider (the documented one is Microsoft Entra ID), and the engineer's directory groups decide both their role and which tenants they can see.[33] On the engineer's laptop, EmberNet Endpoint enrolls once with the engineer's Microsoft account through the device-code flow. It is split-tunnel by default, claiming routes only for EmberNet-assigned subnets, and its management interface listens on loopback only, never on the network.[32]

Then policy decides. A policy grants a source identity access to a target service, and everything not granted is denied.[29] The documentation's own example reads:

```
Allow:
  Source:   @engineering-team
  Target:   @edge-dashboard
  Protocol: HTTPS
  Ports:    443
```

Grant an engineer's identity the one vPLC they are commissioning, and that is the one thing they can reach; the HMI next to it and every other controller stay denied. When access is denied, it is usually the system working correctly, and EmberCORE shows the policy evaluation logs and connection attempt history.[29]

PART VIII

Getting There with Fireball

Part V says the migration is phased, not rip-and-replace: visibility first, then crawl, walk, run. This is how that runs as a Fireball Industries project, and what you do in each phase.

How we work

Every Fireball Industries project runs through the same six steps.[35]

1. **Conversation.** We start with a conversation about your operations and goals.
2. **Scope and quote.** We scope a solution that fits your timeline and budget, then send back a quote you can act on.
3. **Design and build.** Controls design and system architecture, robotics and motion integration, panel builds, and prototype and pilot-line development.
4. **Digital commissioning.** We validate systems before they reach your floor, for faster, safer rollouts with less downtime and fewer surprises at startup.
5. **Installation and startup.** Project management and site startups, with on-site installation and SCADA support.
6. **Support.** Remote and on-site troubleshooting, preventive and predictive maintenance, retrofit and modernization, and the Manufacturing Excellence Program: monthly support for continuous automation, AI, and process improvement, without the overhead of full-time staff.

The migration, phase by phase

The three stages are the ones in Figure 6. Each checklist is work your team does, with Fireball alongside. None of it is exotic: NIST defines zero trust architecture in SP 800-207 and covers OT security in SP 800-82 Rev. 3,[24][25] and in April 2026 CISA, with the Department of War, the Department of Energy, the FBI, and the Department of State, published joint guidance on adapting zero trust principles to operational technology.[26]

Stage 1, Crawl: discover and observe

- Inventory every device, every user, and every communication pathway on the plant network.
- Inventory every remote access path: VPN profiles, vendor accounts, remote desktop tools, and the cellular modems on maintenance laptops. Colonial Pipeline's attacker came in through a legacy VPN profile that was not intended to be in use.[13]
- Write down which systems actually talk to which, and why.
- Confirm outbound HTTPS on 443, DNS, and NTP from the machines that will become nodes, tested from the machines themselves rather than read off the firewall policy.[31]
- Create the directory security groups that will map to roles, so access is ready before anyone needs it.[33]

Stage 2, Walk: simulate and segment

- Run microsegmentation in monitoring-only mode and let the normal communication patterns show themselves.
- Review the recommended least-privilege policies with operations, line by line, and simulate them before you enforce anything.
- Decide which nodes need EmberRTOS with the real-time kernel before they are provisioned, not after.[31]
- List who needs to reach which controller, and for what task. That list becomes your policies.

Stage 3, Run: enforce and automate

- Enforce first on your most critical assets: Safety Instrumented Systems, primary PLCs, and SCADA servers, each in its own microsegment.
- Move remote OT access from VPNs to identity-based access one group at a time, then shut the VPN profiles off.
- Containerize and modernize industrial applications on k3s, running on EmberRTOS or a supported Linux distribution.
- Measure against the Part V success metrics: the lateral movement attack surface, a simulated breach contained to a single device, VPN-based OT access remaining, and the time it takes to patch the fleet.

Start with a conversation

Tell us what runs on your floor, how people reach it today, and what you need it to do next. Email **info@fireballz.ai**.

EmberNet is priced per control plane, never per node. Pilot is free, one per organization. Starter is $2,500 a month and Pro is $5,000 a month. Enterprise is contact sales.

CONCLUSION

Choose Your Side

We have laid the case bare. The Purdue Model is an artifact of a bygone era, a framework of architectural negligence that has been proven, repeatedly and catastrophically, to fail. Its foundational principles (the mythical air gap, the trusted internal network, the rigid hierarchy) are the very vulnerabilities that modern adversaries exploit to cause financial ruin and operational paralysis. To continue relying on this model is not a conservative choice; it is a reckless gamble against overwhelming odds. It is a professional malpractice that invites disaster and assigns liability directly to those who refuse to see the writing on the wall.

The path forward is not an incremental adjustment. It is a complete and total paradigm shift to a Zero Trust architecture. This is the consensus of the world's leading cybersecurity experts and the strategic direction of the U.S. Department of Defense.[23] It is the only approach that accepts the reality of today's converged, porous, and hostile network environments. It is the only strategy that provides meaningful containment and resilience in the face of a breach that you must assume has already happened.

This is a moment of decision. There are two sides. On one side is the Purdue Model: a **suicide pact** with the past, a comforting delusion that will inevitably lead to a rude and brutal awakening. It is the path of inertia, of hoping for the best while building for the worst-case scenario.

On the other side is the Zero Trust imperative, made real by **EmberNet**. This is the path of reality-based security. It is an architecture of resilience, containment, and agility, designed to protect critical infrastructure not in the world we wish we had, but in the world we actually live in. It is the choice to build a fortress that can withstand the modern siege, a system that can contain the fire and heal itself.

For every CISO, every plant manager, every executive, and every board member, the choice is yours. You can continue to endorse a failed architecture and hope you are not the next headline, or you can take decisive action to protect your operations, your shareholders, and your career. There is no middle ground. There is no more time for debate. Choose your side.

APPENDIX

Methodology

The original argument

Each figure was checked against a public source, and a superscript marks every figure a source supports. The original paper argues from incidents, not formulas, and has no calculations of its own. The success metrics in Part V (for example, a 99% reduction in the lateral movement attack surface) are targets for a migration, not measured results, and the paper defines no formula for them; this edition adds none.

The new figures

FIGURE	HOW IT WAS DERIVED
4 to 9 times inside the band	Target band divided by the measured worst case: 100 ÷ 23 ≈ 4.3 and 200 ÷ 23 ≈ 8.7.
About 627 million samples	Each thread wakes once per interval for 86,423 seconds: about 432 million at 200 µs, 123 million at 700 µs, and 72 million at 1200 µs, for about 627 million in total.
1,184 and 7,946 miles	Great-circle (straight-line) distance between city centers, by the haversine formula with an Earth radius of 3,958.8 miles. It is not the network path length.

What each source is

SOURCE TYPE	REFERENCES
Congressional testimony	Colonial Pipeline CEO before the U.S. Senate [13]
Government advisories, strategy, and rules	CISA [17][26], DoD CIO [9][23], NIST [24][25], SEC [22]
Company statements as reported	Colonial Pipeline ransom [15], JBS ransom [16]
Journalism and research	Dark Reading [14], Wired [18], Symantec [19], Institute for Science and International Security [20], the PERA paper itself [21], Tampa Bay Times [36]
Original paper's references	[1] to [12], kept as published and rechecked
EmberNet's own results and documentation	EmberRTOS hardware results [27], CODESYS on EmberNet [28], EmberNet documentation [29] to [34], Fireball Industries [35]. The Cleveland to Mumbai download is reported by Fireball Industries; no public page describes it.

References

All links were checked on October 5, 2026.

1. SANS Institute. "The Purdue Model: Introduction to ICS Security Part 2." 2021. https://www.sans.org/blog/introduction-to-ics-security-part-2
2. Wikipedia. "Purdue Enterprise Reference Architecture." https://en.wikipedia.org/wiki/Purdue_Enterprise_Reference_Architecture
3. Gurvinder Pal Singh, The CyberChef. "Is the Purdue Model Obsolete in the Era of IoT and Cloud Industrial Control Systems (ICS)?" Medium. https://medium.com/@GurvinderPalSingh-TheCyberChef/is-the-purdue-model-obsolete-in-the-era-of-iot-and-cloud-industrial-control-systems-ics-5805d022ac4e
4. Help Net Security. "How hackers exploit critical infrastructure." 2018. https://www.helpnetsecurity.com/2018/07/19/hackers-exploit-critical-infrastructure/
5. TechTarget. "Industroyer2: How Ukraine avoided another blackout attack." 2022. https://www.techtarget.com/searchsecurity/news/252523694/Industroyer2-How-Ukraine-avoided-another-blackout-attack
6. Carnegie Mellon University Software Engineering Institute. "IT, OT, and ZT: Implementing Zero Trust in Industrial Control Systems." 2022. https://www.sei.cmu.edu/blog/it-ot-and-zt-implementing-zero-trust-in-industrial-control-systems/
7. Industrial Cyber. "Zero Trust: The Essential Guide." https://industrialcyber.co/zero-trust/zero-trust-the-essential-guide/
8. Expert Insights. "Zero Trust Adoption Statistics and Trends in 2025." https://expertinsights.com/zero-trust/zero-trust-adoption-statistics-and-trends
9. U.S. Department of Defense, Chief Information Officer. "Zero Trust for Operational Technology Activities and Outcomes." 2025. https://dodcio.defense.gov/Portals/0/Documents/Library/ZT-OperationalTechnologyActivitiesOutcomes.pdf
10. Worldline Tech Blog. "Immutable OS: a new paradigm for more secure and resilient systems." 2023. https://blog.worldline.tech/2023/03/29/immutable_os.html
11. The New Stack. "3 Immutable Operating Systems: Bottlerocket, Flatcar and Talos Linux." https://thenewstack.io/3-immutable-operating-systems-bottlerocket-flatcar-and-talos-linux/
12. Control Engineering. "Edge to cloud: Understanding new industrial architectures." https://www.controleng.com/edge-to-cloud-understanding-new-industrial-architectures/
13. Joseph Blount, President and CEO, Colonial Pipeline Company. Testimony before the U.S. Senate Committee on Homeland Security and Governmental Affairs. June 8, 2021. https://www.hsgac.senate.gov/imo/media/doc/Testimony-Blount-2021-06-08.pdf
14. Dark Reading. "Colonial Pipeline CEO: Ransomware Attack Started via Pilfered 'Legacy' VPN Account." June 2021. https://www.darkreading.com/cyberattacks-data-breaches/colonial-pipeline-ceo-ransomware-attack-started-via-pilfered-legacy-vpn-account
15. Minnesota Reformer (States Newsroom). "Colonial Pipeline CEO: 'One of the toughest decisions I have had to make' to pay a $4.4M ransom." June 8, 2021. https://minnesotareformer.com/briefs/colonial-pipeline-ceo-one-of-the-toughest-decisions-i-have-had-to-make-to-pay-a-4-4m-ransom/
16. IndustryWeek (Agence France-Presse). "Meatpacking Giant JBS Says Paid $11 Million to Hackers in Ransomware Attack." June 2021. https://www.industryweek.com/technology-and-iiot/article/21166663/meatpacking-giant-jbs-says-paid-11-million-to-hackers-in-ransomware-attack
17. CISA, FBI, EPA, and MS-ISAC. "Compromise of U.S. Water Treatment Facility." Joint Cybersecurity Advisory AA21-042A. February 2021. https://www.cisa.gov/news-events/cybersecurity-advisories/aa21-042a
18. Andy Greenberg. "The Untold Story of NotPetya, the Most Devastating Cyberattack in History." Wired. August 2018. https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/
19. Symantec Threat Hunter Team. "W32.Stuxnet Dossier." https://www.security.com/threat-intelligence/stuxnet-dossier-espionage

20. David Albright, Paul Brannan, and Christina Walrond. "Did Stuxnet Take Out 1,000 Centrifuges at the Natanz Enrichment Plant?" Institute for Science and International Security. December 22, 2010. https://isis-online.org/isis-reports/did-stuxnet-take-out-1000-centrifuges-at-the-natanz-enrichment-plant/
21. Theodore J. Williams. "The Purdue enterprise reference architecture." Computers in Industry 24, no. 2-3 (1994): 141-158. https://doi.org/10.1016/0166-3615(94)90017-5
22. U.S. Securities and Exchange Commission. "SEC Adopts Rules on Cybersecurity Risk Management, Strategy, Governance, and Incident Disclosure by Public Companies." Press release 2023-139. July 26, 2023. https://www.sec.gov/news/press-release/2023-139
23. U.S. Department of Defense, Chief Information Officer. "DoD Zero Trust Strategy." October 21, 2022. https://dodcio.defense.gov/Portals/0/Documents/Library/DoD-ZTStrategy.pdf
24. National Institute of Standards and Technology. "Zero Trust Architecture." SP 800-207. August 2020. https://csrc.nist.gov/pubs/sp/800/207/final
25. National Institute of Standards and Technology. "Guide to Operational Technology (OT) Security." SP 800-82 Rev. 3. September 2023. https://csrc.nist.gov/pubs/sp/800/82/r3/final
26. CISA. "Adapting Zero Trust Principles to Operational Technology." April 29, 2026. https://www.cisa.gov/resources-tools/resources/adapting-zero-trust-principles-operational-technology
27. EmberNet, Fireball Industries. "EmberRTOS: the real-time edge OS" (hardware results). 2026. https://embernet.ai/emberrtos.html
28. EmberNet, Fireball Industries. "CODESYS on EmberNet." 2026. https://embernet.ai/codesys.html
29. EmberNet Docs. "Zero-Trust Networking." 2026. https://docs.embernet.ai/docs/networking/zero-trust
30. EmberNet Docs. "Flux zero-trust networking." 2026. https://docs.embernet.ai/docs/services/flux
31. EmberNet Docs. "Installation Guide." 2026. https://docs.embernet.ai/docs/deployment/installation
32. EmberNet Docs. "EmberNet Endpoint." 2026. https://docs.embernet.ai/docs/embernet-endpoint/overview
33. EmberNet Docs. "SSO Integration." 2026. https://docs.embernet.ai/docs/platform/sso
34. EmberNet Docs. "Architecture." 2026. https://docs.embernet.ai/docs/platform/architecture
35. Fireball Industries. "About" (how a project runs). 2026. https://www.fireballz.ai/about
36. Jack Evans, Tampa Bay Times. "Cyberattack on Oldsmar's water supply never happened, official says." April 11, 2023 (via Bay News 9). https://baynews9.com/fl/tampa/news/2023/04/11/cyberattack-on-oldsmar-s-water-supply-never-happened--official-says

WORK WITH FIREBALL INDUSTRIES

Ready to retire the Purdue Model?

Fireball Industries designs, builds, commissions, and supports industrial automation, and builds EmberNet, the platform this paper describes. Every engagement starts the same way: a conversation about your operations and goals, then a scope that fits your timeline and budget.

Email
info@fireballz.ai

Fireball Industries
fireballz.ai

EmberNet
embernet.ai



© 2026 Fireball Industries LLC. All rights reserved. EmberNet is a product of Fireball Industries.