International Journal of AI, Big Data, Computational and Management Studies
Noble Scholar Research Group | Volume 3, Issue 1, PP. 114-122, January 2022
ISSN: 3050-9416 | https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I1P113

Secure DevSecOps Workflows for Medical IoT Device Integration in Smart Hospitals

Nagarjuna Nellutla
Independent Researcher Eagan, MN, USA .

Abstract: Smart hospitals increasingly depend on interconnected medical IoT devices that collect physiological signals, execute clinical workflows, and support real-time decision making. These devices require continuous software updates and secure firmware delivery to maintain safety and interoperability across complex healthcare ecosystems. Conventional software pipelines are insufficient for this purpose because medical devices operate under strict regulatory constraints, involve persistent wireless connectivity, and remain exposed to both cyber threats and physical misuse. This paper proposes a secure DevSecOps workflow tailored specifically for medical IoT integration within smart hospital environments. The workflow incorporates secure firmware signing, hardware-rooted device identity, continuous vulnerability scanning, compliance-aware release gating, and post-deployment telemetry validation. By embedding security checks directly throughout the update and release lifecycle, medical IoT devices obtain resilient over-the-air provisioning and maintain verifiable trust from manufacturer to bedside. The resulting pipeline shifts device cybersecurity from post-release patching to continuous assurance, enabling safe, traceable, and regulation-conscious delivery of medical device software updates in smart hospitals.

Keywords: DevSecOps, IoMT, Smart Hospitals, OTA Firmware, Medical Device Security, CI/CD, Healthcare Cyber-Physical Systems.

1. Introduction

Smart hospitals rely on large networks of medical IoT devices that collect physiological data, control therapeutic actuators, and interface with clinical record systems. Unlike consumer IoT infrastructure, these medical assets directly influence diagnosis, treatment delivery, and patient monitoring, making them part of a critical cyber-physical safety ecosystem. Devices such as infusion controllers, wearable vital monitors, and bedside sensors must operate continuously, exchange authenticated data with hospital platforms, and remain resistant to malicious tampering. Their software cannot be updated through informal procedures or rapid-release consumer methods, because small failures may propagate into clinical hazards. Therefore, a secure and traceable software lifecycle is central to the safe operation of networked medical technology.

Traditional device maintenance models rely on scheduled service updates, vendor-controlled patches, and occasional onsite upgrades performed by biomedical technicians. These models are increasingly insufficient in smart hospital environments where systems evolve rapidly and connectivity exposes devices to emerging threats that cannot wait for delayed maintenance cycles. Modern deployment patterns require continuous delivery of updates over wireless networks and secure over-the-air firmware provisioning. This transformation expands the attack surface and demands tighter connection between device developers, cybersecurity teams, regulatory constraints, and hospital network operators.

DevSecOps addresses these challenges by embedding secure engineering processes directly into firmware development, build automation, and deployment pipelines. Unlike conventional CI/CD processes, DevSecOps workflows for medical IoT devices operate under explicit safety requirements, formal verification checkpoints, and regulated auditability. Security cannot be treated as an external testing step. Instead, identity management, compliance validation, and image signing mechanisms must be interwoven within every phase of software creation. A trusted chain of custody must extend from source code to physical device execution, ensuring that system compromise cannot occur through manipulated updates or unsafe configurations.

Firmware delivery for medical IoT devices must therefore include rigorous assurances of authenticity, origin attribution, and device-specific permissions. Device trust must be rooted in immutable identity such as hardware-backed cryptographic keys, preventing unauthorized provisioning and binding each software artifact to a verifiable entity. Enforcement mechanisms must ensure that compromised nodes cannot receive privileged updates and that revoked or quarantined devices are automatically barred from receiving firmware images. Secure management of device identity becomes inseparable from update governance, as identity validation protects both networked data and therapeutic control functions.

In addition, medical IoT software pipelines must treat vulnerability scanning and threat modeling as continuous feedback loops rather than event-driven assessments. Automated scanning during build and integration phases identifies code weaknesses before they reach clinical environments. Post deployment telemetry, combined with behavior-based anomaly detection, helps verify that the update behaves safely in production and does not compromise system responsiveness, communication stability, or sensor accuracy. Pipeline revalidation ensures that devices do not merely receive signed binaries, but also continue operating within expected safety parameters once the update is installed.

Regulated environments demand that every decision made in the update lifecycle be traceable and reversible. Audit logs must capture approvals, build signatures, cryptographic hashes, and test results, enabling forensic review and regulatory compliance during device certification or post-incident investigation. Rollback mechanisms must be engineered with the same rigor as deployment, allowing unsafe firmware to be rapidly withdrawn without leaving devices in a non-functional or unauthenticated state. Hospitals must maintain operational control during remediation actions, ensuring that safety-critical workflows remain uninterrupted.

Finally, secure DevSecOps integration for medical IoT is not merely a cybersecurity exercise but an operational discipline that blends software practices with patient safety engineering. These devices form an extension of the clinical environment, and their software must be treated with the same rigor as medical decision-support tools. A structured lifecycle for firmware creation, delivery, and validation ensures that hospital networks can safely scale connected care without introducing uncontrolled vulnerabilities. This paper details the architectural and operational requirements for implementing secure DevSecOps pipelines that protect medical IoT infrastructure against misuse, malfunction, and evolving cyber threats within smart hospitals.

2. Architecture of a Secure Iomt Devsecops Workflow

A secure DevSecOps workflow for medical IoT devices requires integrating firmware development, cryptographic signing, compliance validation, and controlled over-the-air delivery into a single pipeline that enforces safety and traceability. Unlike software running on standard IT assets, medical firmware interacts with physiological sensing and clinical control functions, which means that every stage of the software lifecycle must be governed by authenticated processes bound to device identity [1]. In a smart hospital ecosystem, this workflow cannot depend solely on vendor-driven release patterns; instead, development pipelines must embed hospital network trust, device authorization policies, and compliance constraints prior to deployment.

A secure IoMT DevSecOps architecture begins at the build stage, where firmware components are compiled from validated source repositories. During this step, static analysis and vulnerability scanning operate continuously to prevent unsafe dependencies, insecure libraries, or outdated cryptographic routines from being introduced. Build outputs are generated as immutable artifacts, ensuring that no modifications occur after scanning. These outputs must be packaged with metadata describing version integrity, risk classification, and device class compatibility.

Following compilation, firmware images undergo cryptographic signing using a hardware-rooted key hierarchy. Signing provides immutable traceability by binding each artifact to a trusted origin. Keys must be protected by secure enclaves or hardware security modules, ensuring that compromised systems cannot forge updates. Image signing prevents rogue intermediaries, network attacks, or device impersonation from injecting malicious versions into the delivery workflow. Once signed, artifacts can be distributed only to devices that possess matching trust anchors.

Compliance enforcement serves as a gating step between signed build artifacts and OTA provisioning. This stage verifies regulatory requirements such as device qualification level, safety classification, and audit policy thresholds. Firmware cannot proceed to deployment unless it meets security scanning scores, passes cryptographic verification, and satisfies rules regarding permitted device scope. Compliance failures trigger automated rollback of the pipeline and revoke build artifacts without human intervention, preventing unsafe firmware from ever reaching smart hospital environments.

OTA delivery requires secure channels that restrict device enrollment, authenticate every session, and verify that devices request only versions they are entitled to receive. Devices authenticate using immutable identities derived from hardware trust anchors, protecting against identity spoofing, unauthorized firmware downgrade, and man-in-the-middle attacks. Update transactions are logged automatically to ensure post deployment accountability. The OTA stage must also enforce throttling policies that prevent mass updates from overwhelming hospital bandwidth or clinical workflows during peak demand.

The final architectural step validates post-deployment behavior using telemetry that checks for safety violations, degraded device responsiveness, wireless communication irregularities, and abnormal sensor patterns [2]. Anomalies trigger either partial rollback or device quarantine, depending on severity. Rollback mechanisms use pre-signed fallback images, ensuring that devices remain operational even after revocation. Telemetry verification, combined with signing and compliance gates, forms a continuous feedback loop that keeps firmware integrity assured throughout its lifecycle.

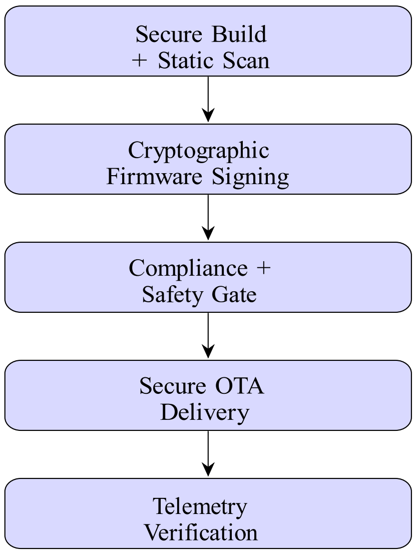

Figure 1: Pipeline-Style Devsecops Architecture for OTA Firmware Delivery to Medical IoT Devices in Smart Hospitals

Fig. 1 depicts the secure DevSecOps pipeline used for medical IoT firmware delivery. It illustrates build-time scanning, signing, compliance gating, secure OTA provisioning, and post-deployment validation, forming a closed feedback loop tailored for smart hospital environments.

This workflow transforms firmware provisioning from ad-hoc device management into a controlled lifecycle of traceable artifacts, policy-verified releases, and post-deployment safety monitoring. By ensuring that no firmware can bypass signing, compliance, or telemetry validation, smart hospitals gain a secure foundation for scaling medical IoT integration without compromising patient safety or operational trust.

3. Secure Ota Firmware and Identity Architecture

Over-the-air (OTA) delivery for medical IoT devices requires an identity-anchored approach that binds firmware packages to authenticated targets throughout their lifecycle [3]. Unlike conventional consumer IoT updates, medical firmware influences therapeutic actions, physiological monitoring, and telemetry used for clinical decision support [4]. OTA provisioning is thus inseparable from device identity, safety enforcement, and backward-compatible trust controls. Secure firmware workflows must guarantee that update artifacts originate from a verified source, are delivered to authorized devices, and remain unmodified from signing to installation.

Secure OTA delivery begins with identity issuance rooted in the device manufacturing process. During enrollment, each device receives a hardware-anchored identifier tied to a cryptographic root key that cannot be cloned or reassigned. These keys provide immutable trust that persists even through firmware wipes or network resets. Device registration binds the hardware identity to hospital infrastructure, ensuring that firmware updates are not accepted from external networks or unauthorized vendors. Identity enforcement prevents devices from installing foreign, tampered, or downgrade firmware that could compromise clinical operation.

Once trust anchors are established, OTA firmware packages must be delivered using authenticated sessions and controlled bandwidth. Each update transaction is negotiated with mutual authentication so that both the device and OTA server validate their identity before any software is transferred [5]. Firmware is encrypted in transit, using symmetric keys derived from device-specific secrets, avoiding exposure of key material to intermediaries or third-party gateways. Upon receiving a package, the device verifies signatures, checks version lineage, and ensures the image is compatible with its configuration and hardware revision. OTA sessions are rejected if identity mismatch or version rollback risk is detected.

Fig. 2 shows a trust pipeline tying the firmware signing environment to device identity validation and hospital delivery infrastructure. This identity-centric workflow ensures that OTA deployment is not simply a network transfer task but a controlled distribution of clinically regulated software.

To preserve operational safety, firmware systems must also support rollback, quarantine, and version lineage enforcement. Rollback uses pre-signed fallback images, ensuring that devices are never downgraded to unsigned or unverified builds.

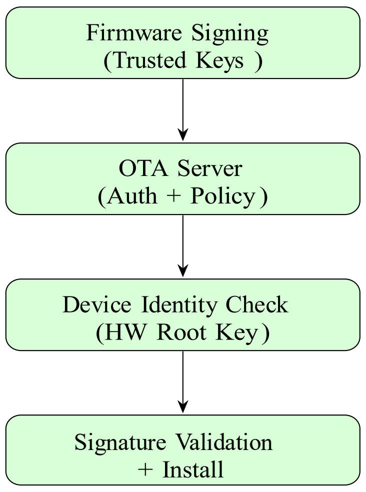

Figure 2: Identity-Anchored OTA Workflow from Signing Through Device Installation

Quarantine isolates devices that repeatedly fail validation, protecting other systems from propagating corrupted firmware. Version lineage associates each image with its predecessors and authorized successors, preventing unsafe branches from reentering the deployment pipeline. This guarantees that the identity chain extends beyond delivery and governs long-term update integrity.

Firmware integrity controls vary in complexity and suitability depending on device role, hardware capabilities, and regulatory class. Table I compares common integrity strategies used in secure OTA systems, including their operational implications in smart hospitals.

Table 1: Integrity Mechanisms Supporting Secure OTA Firmware Provisioning in Medical IoT Devices

Integrity Method	Security Guarantees	Operational Implications
Symmetric MAC	Prevents tampering in transit	Requires secure key distribution; hospital must manage shared secrets
Public Key Signature	Verifies origin + authenticity	Enables vendor traceability; modest CPU cost on constrained devices
Hardware-Rooted Boot	Validates entire boot chain	Requires secure elements; guarantees safe startup even after OTA failure

Together, cryptographic identity controls, authenticated OTA sessions, and lineage-aware firmware governance form a full trust loop. Smart hospitals gain the ability to deploy updates confidently while preventing tampering, impersonation, and unsafe downgrades. OTA becomes a regulated, measurable extension of DevSecOps rather than a maintenance task, aligning device software with clinical safety expectations across its operational lifespan.

4. Automated Compliance, Policy Gating, and Regulatory Alignment

Secure deployment of medical IoT firmware requires not only encryption and device identity but also enforceable compliance rules that block unsafe builds before they reach smart hospital environments [6]. These rules originate from regulatory and safety requirements that govern clinical software, embedded medical devices, and protected health data. In DevSecOps workflows, compliance becomes an executable component of the pipeline rather than a post-release audit [7]. Security behaviors are converted into machine-verifiable gates that determine whether firmware is eligible for delivery, ensuring that all updates meet regulatory and safety expectations before they are signed or distributed.

Policy gates combine static rules, risk scoring, and mandatory test outcomes into executable compliance checks. Firmware cannot progress through the pipeline unless it satisfies cryptographic standards, safety signatures, vulnerability score thresholds, vendor traceability requirements, and device class compatibility [8]. This turns compliance into a binary decision point rather than a subjective assessment. When a build fails policy validation, the workflow automatically revokes its eligibility and triggers a remediation ticket for engineering or biomedical cybersecurity teams [9]. No manual override is permitted unless a controlled exception pathway is triggered by hospital risk authorities. This prevents hurried releases and avoids un reviewed firmware escaping into clinical settings.

Regulatory alignment must be enforced at the same level of precision. Medical devices are developed under lifecycle standards that define software safety classifications, risk management processes, and post-market security responsibilities. Smart hospitals cannot rely on vendor claims alone; they must verify that each firmware update remains compliant at the moment of deployment. Policy gates, therefore, incorporate regulatory static rules linked to software classification, risk control measures, software provenance, documentation completeness, and device-specific operational profiles. Updates are rejected if they violate classification constraints or if their intended deployment conflicts with hospital usage policies or clinical restrictions.

Policy enforcement also applies to post-deployment requirements. Firmware must include traceable version metadata, rollback protections, and verifiable controls for cybersecurity risk management. Updates that fail to provide audit-ready identifiers or lack documented remedial behavior cannot be released because they would interfere with incident investigation, device forensics, or quality assurance. This transforms firmware from a functional artifact into a governed clinical component, with lifecycle guarantees that persist beyond installation. DevSecOps pipelines therefore extend regulatory enforcement from engineering teams to hospital operations, eliminating blind spots that traditionally occurred between vendor development and clinical usage.

Fig. 3 illustrates a policy-driven gating system for medical IoT DevSecOps workflows. It uses a **split-decision diamond** model, representing how firmware is evaluated against regulatory standards, cryptographic requirements, vulnerability thresholds, and device-classification rules. Only firmware passing all requirements proceeds to OTA signing and delivery.

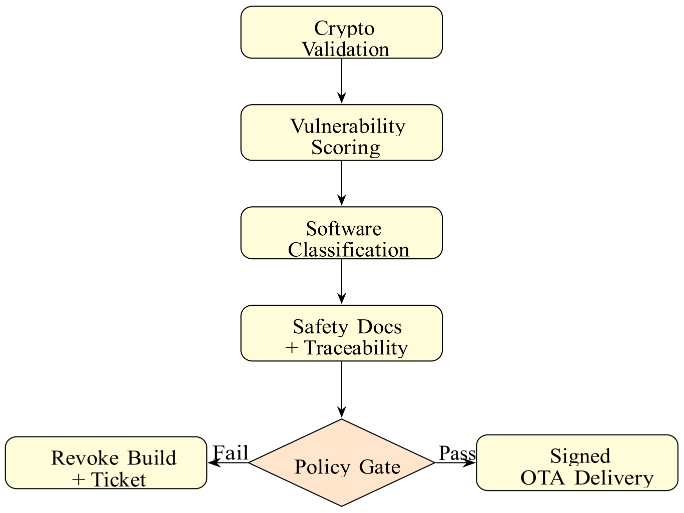

Figure 3: Compliance Policy Gate for Devsecops Enforcement in Medical IoT OTA Workflows

Automating compliance transforms firmware delivery from a reactive legal checkpoint into a continuous safety mechanism. Smart hospitals no longer wait for audits to discover problems. Instead, unsafe software is eliminated early and automatically, and only versions proven compliant at deployment time reach clinical environments. This results in firmware that is safe by design, traceable by default, and aligned with regulatory intent throughout its lifecycle, not just during initial certification.

5. Runtime Telemetry, Threat Monitoring, and Behavioral Rollback for Iomt

Secure DevSecOps workflows do not end once firmware has been deployed. In the operational phase, medical IoT devices must continuously validate their behavior to ensure that neither malicious tampering nor unexpected failure conditions threaten clinical functionality [10]. Runtime security therefore becomes a parallel safeguard alongside OTA delivery, providing continuous monitoring and enforceable rollback logic. Smart hospitals rely on telemetry from deployed devices to detect instability, unauthorized modifications, and abnormal execution patterns, all of which signal the need for remediation. Unlike traditional consumer IoT monitoring, behavioral verification in hospitals must align with clinical safety priorities, device classification, and predictable physiological workloads.

Telemetry collection focuses on two data domains: system behavior and clinical interaction. System behavior includes CPU load, memory saturation, sensor error rates, firmware integrity measurements, wireless signal anomalies, and transient reboot frequency. Clinical interaction captures data freshness, physiological reading cadence, connectivity to clinical record systems, and responsiveness to control commands. Combining these views ensures that threat monitoring does not simply detect technical anomalies but identifies deviations that could affect patient diagnosis, monitoring reliability, or therapy delivery. Behavioral monitoring must therefore differentiate noisy device conditions from genuine safety failures and cybersecurity compromise.

Threat detection relies on classic anomaly modeling using deterministic thresholds, trend deviation metrics, statistical baselines, and cluster grouping of telemetry profiles. For medical IoT, deterministic checks remain essential, such as verifying that physiological value frequency remains within expected sampling intervals, that update rollbacks do not exceed safety counters, or that firmware hash checks persist across reboots. Trend deviation monitors detect delayed drifts such as power irregularities, wireless degradation, and excessive dropped packets. Cluster analytics classify persistent abnormal communication as potential spoofing or impersonation attempts, particularly if identity-bound keys fail repeatedly during OTA checks.

To ensure that monitoring does not produce excessive false positives, telemetry decisions are weighted using clinical priority scoring. Devices that capture critical physiological functions trigger immediate rollback or quarantine when anomalies are detected. Devices that support non-critical tasks, such as asset tagging or inventory sensing, generate lower-priority audits or delayed remediation cycles. This prioritization ensures that automation honors clinical risk without requiring manual interpretation for every deviation. The pipeline therefore orchestrates remediation with responsiveness proportional to safety relevance.

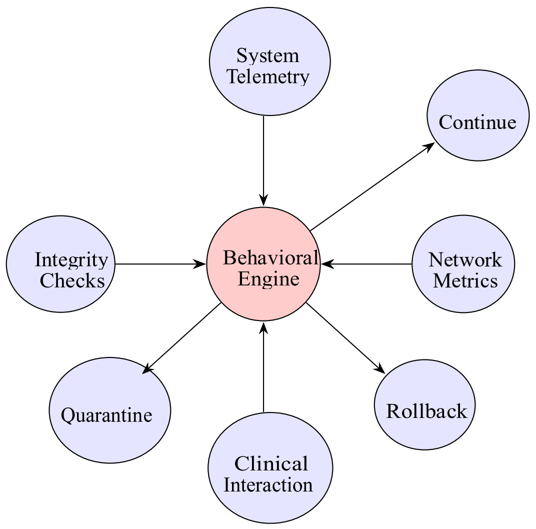

Figure 4: Radial Telemetry Model Enabling Runtime Behavior Monitoring and Rollback Decisions for Deployed Medical IoT Devices

The radial layout in Fig. 4 emphasizes that runtime protection for medical IoT devices is not a linear workflow but a closed behavioral loop in which multiple telemetry sources converge to a central decision engine. Unlike traditional monitoring that evaluates only communication or CPU metrics, this model treats firmware integrity, clinical interaction, and network behavior as coequal indicators of device safety. The execution outcome is therefore determined by the combined weight of these telemetry signals rather than any single trigger, ensuring that remediation actions such as rollback or quarantine are only activated when the overall behavioral profile indicates unacceptable clinical or operational risk.

Remediation decisions activate rollback or quarantine paths only when telemetry exceeds defined risk thresholds. Rollback restores the last signed and validated firmware image, retaining device availability while removing untrusted effects. Quarantine isolates devices from the network, preventing propagation of malicious code or compromised telemetry. Both responses remain reversible after forensic review, ensuring operational control. Runtime telemetry thus becomes a dynamic extension of the DevSecOps workflow, enforcing safety throughout the device's operational lifecycle rather than confining security to the design and distribution stages.

6. Interoperability, Lifecycle Coordination, And Multi-Vendor Trust For Iomt Devsecops

Smart hospitals depend on interconnected medical devices sourced from multiple vendors, each of which maintains distinct software lifecycles, security practices, and firmware provisioning mechanisms. Without structured interoperability controls, hospitals risk inconsistent update timing, incompatible cryptographic standards, or conflicting remediation policies that compromise safety [11]. A unified DevSecOps model must therefore coordinate lifecycle activities across heterogeneous device ecosystems, ensuring that trust extends beyond individual vendors and remains enforceable throughout the operational lifespan of each device on the network.

Interoperability begins with harmonized identity management. Vendor-specific credentials cannot operate in isolation, as they risk fragmenting device authentication across hospital systems. Device identity must instead be anchored to hardware-backed trust primitives that remain uniform regardless of manufacturer. This ensures that all firmware verification and OTA authorization are rooted in consistent validation rules. Such an approach prevents rogue devices from masquerading as legitimate endpoints and reduces the need for vendor-unique authentication logic, which complicates integration and increases maintenance risks [12].

Lifecycle coordination further requires standardized release timing and compatibility meta-data. Firmware images must declare supported device classes, safety criticality, cryptographic credentialing requirements, and rollback lineage. Hospitals must be able to correlate vendor updates with clinical workflow schedules, reducing disruption to dependent systems such as infusion therapy, physiological telemetry dashboards, and bedside decision-support tools. Release scheduling allows hospitals to stage updates across critical and non-critical device categories, maintaining operational continuity without delaying security patches that address ongoing exploitation risks.

Multi-vendor trust must also extend into remediation processes. Rollback rules, quarantine enforcement, and telemetry driven monitoring cannot differ arbitrarily between device types. If remediation logic is inconsistent across vendors, compromised devices could propagate harmful behavior into hospital networks simply because they are governed by weaker remediation criteria. A unified remediation policy binds all devices to the same thresholds for behavioral rollback, network quarantine, and firmware revocation. This approach ensures that trust is not defined by vendor compliance alone but by a shared safety standard enforced throughout the DevSecOps pipeline.

Fig. 5 illustrates a tri-domain trust architecture defined by hospital governance, vendor assurance, and device-level identity. Each domain enforces specific guarantees, and firmware is permitted to operate only when all three trust vectors align. Vendor assurance governs development and signing practices, hospital governance enforces deployment policy and monitoring controls, and device identity guarantees authenticated execution at runtime. None of these domains is sufficient in isolation; they form a closed ecosystem of origin, policy, and execution security.

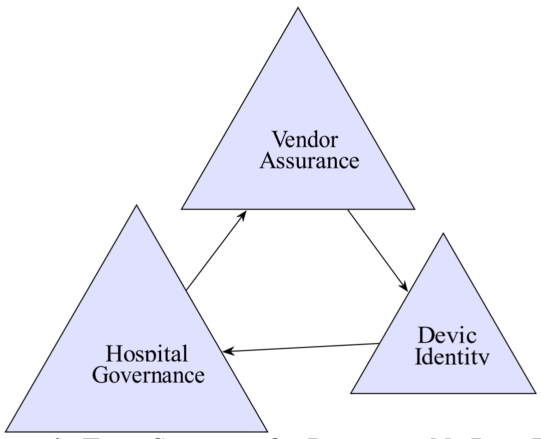

Figure 5: Tri-Domain Trust Structure for Interoperable Iomt Devsecops in Smart Hospitals

The trust model in Fig. 5 highlights that no single stakeholder controls the security state of a connected medical device. Vendors prove software origin and integrity, hospitals enforce clinical and cybersecurity policy, and devices validate execution using strong identity primitives [13]. Updates fail if any party cannot meet its responsibility. This cohesive interaction transforms DevSecOps from a development pipeline into a regulated ecosystem of responsibilities that collectively safeguard patient-centric connected care. When interoperability and lifecycle coordination are enforced through shared trust anchors, smart hospitals gain scalable device ecosystems that remain secure even as products evolve and supply chains diversify.

7. Operational Cost, Scalability, and Safety Trade-Offs

Secure DevSecOps workflows for medical IoT devices inevitably introduce trade-offs among operational cost, scalability, and safety. Smart hospitals must weigh engineering investment, infrastructure requirements, and compliance overhead against the benefits of rapid firmware delivery and strong security assurances. Every design decision—whether around pipeline depth, testing rigor, or rollout strategy—shifts the balance among these dimensions. Understanding these tradeoffs is essential to avoid either over-engineering that slows clinical innovation or under-engineering that exposes patients and infrastructure to unnecessary risk.

From a cost perspective, highly structured pipelines that enforce extensive static analysis, multiple compliance gates, and layered telemetry checks require more engineering effort and operational tooling. They incur higher compute consumption for continuous scanning, artifact storage, and observability infrastructure. However, these costs are offset by reduced manual interventions and fewer emergency remediation efforts following field failures. Less rigorous pipelines have lower immediate expense but risk unplanned downtime, forensic investigation overhead, and reactive patching, which can be more disruptive for hospital operations.

Scalability considerations arise as the number of connected devices and firmware variants grows. Rapid OTA strategies that minimize gating and testing can scale quickly across thousands of devices, but they rely on downstream remediation to correct issues.

Balanced DevSecOps architectures with automated validation and staged rollouts scale more predictably: they allow controlled expansion while preserving safety guarantees. Strict regulation-driven OTA strategies, which require manual approvals or extended testing windows, may struggle to scale if each firmware variant and device class demands separate review cycles.

Safety is the dimension with the least tolerance for compromise. Pipelines that prioritize rapid deployment at the expense of thorough validation increase the risk of propagating faulty firmware or misconfigurations to life-critical devices. Balanced strategies integrate safety into automated gates, ensuring that rapid deployment does not come at the cost of uncontrolled behavior. Regulation-driven approaches maximize safety through conservative release practices, but they can delay important security patches if review processes are slow or fragmented across different stakeholders.

To illustrate these trade-offs, Fig. 6 compares three OTA strategies: rapid deployment with minimal gating, balanced DevSecOps, and heavy regulation-driven OTA. Each strategy is rated across relative cost, scalability, and safety. The chart is not intended to prescribe a single universal choice; rather, it highlights that different hospitals may select different positions depending on their device mix, regulatory risk posture, and resource constraints.

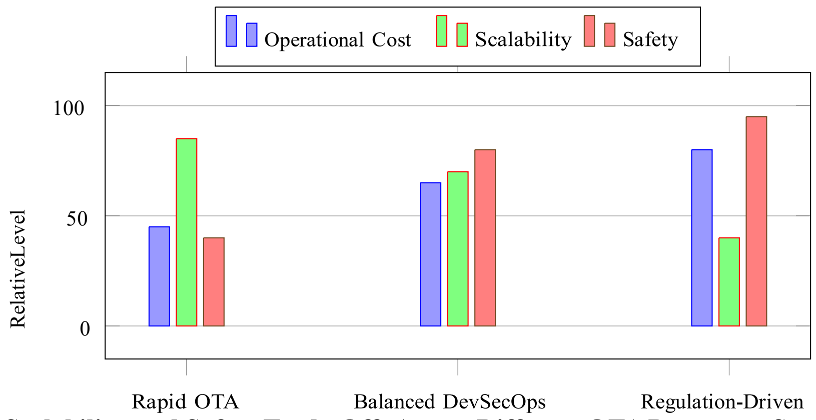

Figure 6: Relative Cost, Scalability, and Safety Trade-Offs Across Different OTA Devsecops Strategies for Medical IoT Devices

As shown in Fig. 6, rapid OTA offers attractive scalability at relatively low short-term operational cost but provides the weakest safety guarantees. Regulation-driven OTA maximizes safety at the expense of scalability and sustained investment in compliance processes. Balanced DevSecOps aims to situate hospitals between these extremes by using automation to reduce manual overhead while keeping safety gates and telemetry-driven validation in place. The appropriate strategy for any smart hospital depends on its device criticality, regulatory exposure, and willingness to invest in long-lived automation rather than reactive incident handling.

8. Conclusion

Secure DevSecOps workflows provide the structural foundation required to manage medical IoT devices within smart hospitals, where firmware changes directly affect patient safety, clinical workflow reliability, and biomedical infrastructure integrity. Unlike consumer IoT ecosystems, medical environments must integrate software delivery into a continuous safety discipline that addresses identity management, traceability, compliance enforcement, telemetry validation, and lifecycle interoperability across multiple vendors. By embedding cryptographic trust anchors, continuous vulnerability evaluation, and automated policy gates into OTA provisioning, DevSecOps elevates firmware delivery from a reactive maintenance task into a persistent assurance mechanism.

The architectural patterns examined in this work show that runtime behavior monitoring and telemetry-driven rollback are essential components of a safety-conscious IoMT platform. These mechanisms ensure that medical devices remain trustworthy even after deployment, preserving both data accuracy and functional correctness under evolving threat conditions. Furthermore, interoperability and multi-vendor trust coordination prevent fragmented pipelines that could weaken the device ecosystem through inconsistent security or delayed patch delivery. Smart hospitals benefit most when the DevSecOps model spans manufacturers, governance authorities, and runtime device execution, forming a closed loop of safety regulated software execution.

Ultimately, secure DevSecOps enables hospitals to scale their connected medical infrastructure without sacrificing patient protections. By treating software integrity as a continuous operational responsibility rather than a discrete engineering milestone, smart hospitals gain resilient IoMT ecosystems capable of adapting securely to technology growth and emerging cybersecurity risks. Robust DevSecOps execution therefore becomes a prerequisite for trusted digital care, supporting long-term innovation in connected medicine while maintaining uncompromised clinical safety.

References

1. Y. Sun, F. P.-W. Lo, and B. Lo, “Security and privacy for the internet of medical things enabled healthcare systems: A survey,” *IEEE Access*, vol. 7, pp. 183339–183355, 2019.
2. R. C. Moioli, P. H. J. Nardelli, M. T. Barros, W. Saad, A. Hekmatmanesh, P. E. G. Silva, A. S. de Sena, M. Dzaferagic, H. Siljak, W. Van Leekwijck, D. C. Melgarejo, and S. Latre, “Neurosciences and´ wireless networks: The potential of brain-type communications and their applications,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1599–1621, 2021.
3. K. Nomikos, A. Papadimitriou, G. Stergiopoulos, D. Koutras, M. Psarakis, and P. Kotzanikolaou, “On a security-oriented design framework for medical iot devices: The hardware security perspective,” in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, 2020, pp. 301–308.
4. A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, “Detecting behavioral change of iot devices using clustering-based network traffic modeling,” *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7295– 7309, 2020.
5. N. Asokan, T. Nyman, N. Rattanavipanon, A.-R. Sadeghi, and G. Tsudik, “Assured: Architecture for secure software update of realistic embedded devices,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2290–2300, 2018.
6. E. Raj, D. Buffoni, M. Westerlund, and K. Ahola, “Edge mlops: An automation framework for aiot applications,” in *2021 IEEE International Conference on Cloud Engineering (IC2E)*, 2021, pp. 191–200.
7. D. Spychalski, O. Rode, M. Ritthaler, and G. Raptis, “Conceptual design and analysis of a mobile digital identity for ehealth applications,” in *2021 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*, 2021, pp. 1–4.
8. T. Yaqoob, H. Abbas, and M. Atiquzzaman, “Security vulnerabilities, attacks, countermeasures, and regulations of networked medical devices—a review,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3723–3768, 2019.
9. Z. Kazemi, A. Papadimitriou, D. Hely, M. Fazcli, and V. Beroulle, “Hardware security evaluation platform for mcu-based connected devices: Application to healthcare iot,” in *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*, 2018, pp. 87–92.
10. K. Kuru and W. Khan, “A framework for the synergistic integration of fully autonomous ground vehicles with smart city,” *IEEE Access*, vol. 9, pp. 923–948, 2021.
11. G. Hatzivasilis, O. Soultatos, S. Ioannidis, C. Verikoukis, G. Demetriou, and C. Tsatsoulis, “Review of security and privacy for the internet of medical things (iomt),” in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2019, pp. 457–464.
12. A. Sobecki, J. Szymanski, D. Gil, and H. Mora, “Framework for in-´ tegration decentralized and untrusted multi-vendor iomt environments,” *IEEE Access*, vol. 8, pp. 108102–108112, 2020.
13. S. Cong, M. Jianfeng, and Y. Qingsong, “On the architecture and development life cycle of secure cyber-physical systems,” *Journal of Communications and Information Networks*, vol. 1, no. 4, pp. 1–21, 2016.