RESEARCH ARTICLE

Secure Software Engineering for Industrial IoT: Integrating Threat Modeling into the Development Lifecycle

Misbah Ali[1,*], **Haroon Arif**[2], **Aamir Raza**[3] and **Moomna Nazir**[4]

[1] Department of Computer Science, COMSATS University Islamabad (CUI), Sahiwal Campus, Sahiwal 57000, Pakistan
[2] Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, United States
[3] Department of Information Technology and Management, Illinois Institute of Technology, Chicago, IL 60616, United States
[4] Department of Computer Science, Government Postgraduate College for Women, Sahiwal 57040, Pakistan

Abstract

The Industrial Internet of Things (IIoT) is central to smart manufacturing, enabling real-time automation, data exchange, and system intelligence. However, the convergence of cyber-physical systems with legacy software and heterogeneous architectures introduces significant security challenges. This paper explores how software engineering principles can be strategically employed to enhance IIoT security by integrating threat modeling into the development lifecycle. In this study, we review classic models such as STRIDE, DREAD, and STPA-Sec, and evaluate their effectiveness when applied at various phases of the Secure Software Development Life Cycle (SSDLC). STRIDE focuses on classifying security threats, DREAD helps score the severity of risks, and STPA-Sec provides a safety-oriented approach to identifying unsafe control actions in IIoT environments. Additionally, we propose a secure development process to embed continuous security assurance during IIoT software deployment. This research highlights design-driven security patterns, model-driven engineering strategies, and secure API development best practices. This paper aims to support developers and architects in designing scalable and threat-aware IIoT systems through the alignment of software engineering with IIoT-specific threat vectors.

Keywords: industrial IoT, software engineering, threat modeling, secure software development lifecycle (SSDLC).

1 Introduction

The Industrial Internet of Things (IIoT) is becoming central to industrial operations. It links various devices including smart sensors, actuators, and machinery for process automation, reduction of human effort, and support for intelligent decision-making. IIoT is fundamental to Industry 4.0 as it enables real-time data collection, data-driven maintenance, and adds to

Submitted: 11 July 2025
Accepted: 26 August 2025
Published: 24 October 2025

Vol. 1, **No.** 2, 2025.
10.62762/JSE.2025.729568

***Corresponding author:**
✉ Misbah Ali
talktomisbah.ali@gmail.com

Citation
Ali, M., Arif, H., Raza, A., & Nazir, M. (2025). Secure Software Engineering for Industrial IoT: Integrating Threat Modeling into the Development Lifecycle. *ICCK Journal of Software Engineering*, 1(2), 63–74.

© 2025 by the Authors. Published by Institute of Central Computation and Knowledge. This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

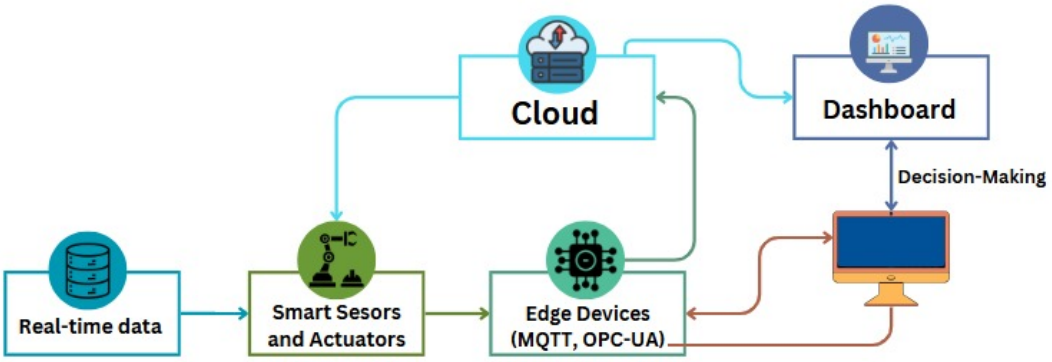

Figure 1. A general High-level view of IIoT architecture.

various sectors such as manufacturing and logistics [1]. This advancement leads to serious security challenges. A significant number of IIoT systems integrate advanced smart devices and legacy systems, which often lack security. The blend of networked machines, embedded controllers, and cloud systems makes IIoT systems more susceptible to threats [2]. Compared to traditional IoT systems, IIoT systems also demand strict operation timings along with availability requirements that amplify the impact of even short-term security failures [3].

Existing research in IIoT is often focused on blocking attacks at runtime by employing advanced machine learning-based intrusion detection techniques or secure communication protocols [4–6]. Another study evaluated [7] how different SDLC structures influence the adoption of secure practices within software development teams. It highlighted that security as a core requirement, is more likely to be implemented consistently across all lifecycle stages. Moreover, authors in [8] comprehensively reviewed several IoT security guidelines from industry, government, and academia. They found that about 73% of the actionable advice relates directly to SSDLC phases like planning, implementation, and testing. This underscores the need to embed secure development practices throughout the lifecycle.

However, major system vulnerabilities do not occur at runtime; but are built into the software during Software Development Life Cycle (SDLC). These vulnerabilities comprising weak authentication logic, insecure APIs, and poor data validation can be prevented using improved software design [9]. Compared to traditional security methods that often focus on reactive measures, threat modeling enables early detection of vulnerabilities during design and planning. This makes it especially suitable for Industrial IoT systems, where failures can cause physical damage and downtime. A general high-level view of IIoT architecture, showing the interaction of smart sensors, edge devices, communication protocols, cloud platforms, and decision-making interfaces is presented in Figure 1.

This gap between security and software engineering is one of the key reasons why IIoT systems remain at risk, even when strong protection tools are added later. While several studies have explored security in IIoT, most either focus on specific phases of development or lack integration between threat modeling and secure development practices. There remains a need for a structured approach that combines early risk identification with secure automation across the entire SDLC. This study is focused on embedding security into SDLC in IIoT environments. We explore multiple threat modeling techniques such as STRIDE, DREAD, and STPA-Sec; that aim to support the identification and analysis of security risks during the early phases of SDLC. The study also investigates the application of these techniques in each phase of the Secure Software Development Life Cycle (SSDLC). This analysis covers defining clear security requirements, designing safer architectures, adhering to secure coding protocols, and frequently testing the system vulnerabilities.

This study proposes an advanced framework for SDLC named SecureDev-IIoT, that introduces the integration of security testing and monitoring into the software delivery pipeline. This framework aims to support IIoT developers and engineers in building more secure systems without compromising the performance, scalability, or accessibility. This study makes the following key contributions: 1) It provides a comparative analysis of widely used threat modeling

techniques, including STRIDE, DREAD, and STPA-Sec, mapped across the phases of a secure development lifecycle; 2) It proposes SecureDev-IIoT, a layered framework that combines threat modeling, SSDLC practices, and secure development process automation designed for IIoT systems; and 3) Finally, it offers practical guidelines and illustrations that support secure software design and deployment in industrial environments.

The rest of the paper is organized as follows. Section 2 explores multiple threat modeling techniques. Section 3 is focused on secure software development lifecycle. Section 4 highlights the integration of secure development process principles into IIoT software delivery pipelines. Section 5 proposes the SecureDev-IIoT framework, through the integration of threat modeling, SSDLC, and secure development process into a unified process. Section 6 evaluates the proposed framework through the comparative analysis with traditional IIoT security mechanisms. Section 7 finally concludes the paper along with future directions.

2 Threat Modeling as a Software Engineering Practice

Developing secure IIoT systems requires more than just reactive protection tools—it starts with understanding where threats can appear and how they could impact the system. This is the core idea behind threat modeling, a structured approach to identifying potential vulnerabilities early in the development lifecycle [10]. In software engineering, threat modeling helps teams think like an attacker, which allows them to build stronger defenses during design and coding rather than after deployment. In IIoT systems, threat modeling becomes even more important because of the unique blend of cyber and physical elements [11]. This section will explore several widely adopted threat modeling techniques and their applications within software engineering processes to strengthen IoT security.

STRIDE is a threat modeling method introduced by Microsoft to help developers identify six key security risks. These are spoofing, tampering, repudiation, information leakage, denial of service, and elevation of privilege [10]. It is commonly applied in the early design stage, during the preparation of system architecture diagrams. In IIoT systems, STRIDE helps examine the interaction between system components, including data shared between sensors and control units. Tampering becomes a concern if software updates reach edge devices without any authentication. Spoofing is also possible in cases where device identification lacks proper verification [11, 12]. Although STRIDE is effective in many areas, it may not fully address changing IIoT environments where devices are frequently added or removed [13].

The DREAD model is an another advanced technique based on threat modeling that it is focused on five main risk factors. It targets damage potential, reproducibility, exploitability, affected users, and discoverability [12]. Hence, its application assists users in prioritizing threats based on their critical nature. In IIoT systems, DREAD is efficient at making risk-based decisions such as deciding the time investment on input validation logic, or applying additional encryption to certain data flows. Specifically, a vulnerable firmware update mechanism may cause wide-scale disruption; hence, getting high DREAD score, highlighting the requirement of redesign [13]. However, DREAD may struggle in dynamic settings where advanced threats emerge periodically.

The STPA-Sec (System-Theoretic Process Analysis for Security) is a state-of-the-art threat modeling technique [14]. In contrast to STRIDE or DREAD, STPA-Sec examines the system as a whole. It evaluates unsafe control actions including sending commands at the wrong time or missing a safety check. This system-level threat modeling technique is particularly applicable for IIoT systems as devices are tightly coupled leveraging automation logic. STPA-Sec encourages developers to think beyond just protecting devices and to consider how their software interacts with physical processes and human operators [15]. Although STPA-Sec requires more time and expertise to apply, it is a powerful tool for modeling safety-critical software behaviors, and can be used during requirements engineering and architecture design phases in the SSDLC. The mapping of these threat modeling techniques to various phases of the SSDLC for IIoT systems is given in Figure 2.

It illustrates that the STRIDE model is most effective during the early stages, such as requirements gathering and system design, where identifying common threat categories can shape secure architecture. DREAD, on the other hand, is better suited for implementation and testing phases, where scoring and prioritizing risks become critical. STPA-Sec, being a system-level method, is applicable during both requirements and design phases and remains relevant in maintenance

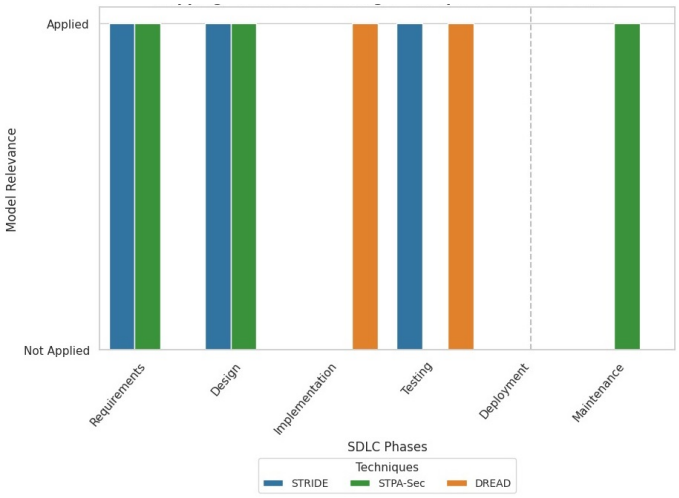

Figure 2. Mapping of threat modeling techniques to SSDLC phases for IIoT systems.

when revisiting system safety and security logic [16].

While these studies offer important insights into IIoT security and secure development practices, most focus on isolated phases of the software lifecycle. Few works integrate threat modeling techniques systematically across all SSDLC phases, and even fewer align these models with development process for secure and automated deployment. In addition, existing approaches often overlook the practical challenges of resource-constrained edge environments, legacy system integration, and post-deployment security monitoring. This highlights a clear need for a unified framework that bridges early risk modeling with continuous security enforcement, as addressed in this paper. A brief comparison of threat modeling techniques in IIOT software engineering is presented in Table 1.

3 Secure Software Development Lifecycle (SSDLC) for IIoT

Secure IIoT systems demand the security embedded at every stage of the SDLC. A Secure Software Development Life Cycle (SSDLC) includes various security activities such as threat modeling, secure coding, and continuous testing [17]. This section explores each phase of SSDLC, transformed to meet various challenges of IIoT systems, including resource constraints and real-time performance requirements.

3.1 Requirements Phase: Capturing Security Needs Early

Requirement engineering is the first phase in SSDLC that involves gathering both functional and security requirements. For IIoT systems, it focuses on various security vulnerabilities in the field. Since IIoT systems often employ remote, autonomous devices, hence, developers must consider unauthorized access, data tampering, and physical sabotage while requirement gathering [17]. Therefore, threat modeling techniques such as STRIDE and STPA-Sec are specifically

Table 1. Comparison of threat modeling techniques in IIoT software engineering.

Aspect	STRIDE	DREAD	STPA-Sec
Focus Area	Threat categorization	Risk prioritization	Unsafe control actions and system behavior
Primary Use Phase	Design	Implementation / Testing	Requirements / Design
Risk Assessment	Identifies threats only	Scoring-based evaluation	System modeling and causal analysis
Complexity	Low	Medium	High
System-Level View	Limited to components	Focused on individual issues	Covers full system including human interaction
Recommended for IIoT	Good for analyzing interfaces and protocols	Useful but may lack consistency in scoring	Ideal for control logic and safety-critical zones
Tool Support	Strong (e.g., Microsoft TM Tool)	Moderate (used in select platforms)	Limited (requires manual modeling or UML/SysML)
Limitations	May miss system-wide threats	Results can be subjective	Requires detailed system knowledge and effort
Strength in IIoT Context	Helps identify protocol/API risks early	Supports remediation prioritization	Captures unsafe behaviors in cyber-physical systems

appreciated in this phase. STRIDE helps in identifying common categories of threats, while STPA-Sec supports a deeper analysis of unsafe control actions that could potentially result in system-level failures [14]. This phase ensures the integration of security as a core part of the initial design discussion.

3.2 Design Phase: Building Secure Architecture

In the design phase, developers and system architects plan the structure of the software and the roles of different components. IIoT systems are built with several layers, including the device layer, network layer, and application layer. Each layer brings its own set of security risks [18]. To protect these layers, developers use secure design practices such as limiting access rights, encrypting data between modules, and adding input checks in APIs. Some modern protocols like OPC-UA come with built-in security, while older ones like Modbus do not include encryption. If legacy protocols are still in use, they should be separated from other parts of the system to avoid exposing critical assets. Tools like attack trees and attack graphs help to identify possible paths that attackers might use to reach system components. Security in this phase enables proactive planning for risk reduction by the identification of weak sections before writing the code [19].

3.3 Implementation Phase: Writing Secure Code

This phase involves converting the system design to system implementation by writing the secure code. This phase is critical, as coding defects including buffer overflows, hard-coded passwords, or invalidated user inputs may create serious vulnerabilities [14]. Following secure coding standards such as MISRA for embedded C/C++ or CERT guidelines for Java/Python; is highly essential in secure IIoT systems. Developers must consider avoiding direct memory access, unsafe system calls, and open ports without authentication [20]. Additionally, lightweight encryption and security libraries should be integrated early; considering the processing limitations of edge devices.

3.4 Testing Phase: Validating Software Integrity

Security testing is essential along with functional testing during development phase. In IIoT, traditional testing techniques such as unit testing and integration testing must be accompanied using security-specific tests. These tests include: 1) Fuzz testing to identify the system's handling of unexpected or malformed inputs. 2) Static and dynamic analysis tools to detect coding flaws. 3) Penetration testing for identifying real-world vulnerabilities before deployment [21]. In this phase, DREAD can be applied to prioritize the severity of discovered threats and guide fixing decisions. Additionally, testing should also be introduced in this phase for protocol security and authentication flows, particularly for IIoT devices performing crucial tasks.

3.5 Deployment Phase: Securing Release and Configuration

After testing, the software is deployed often on IIoT edge devices, fog nodes, or cloud platforms. Deployment must be done securely to prevent man-in-the-middle (MitM) attacks, unauthorized access, or configuration leaks. This involves secure boot mechanisms to prevent firmware tampering, digital signing of software to ensure authenticity, role-based access control and key management during device onboarding, and encrypted configuration files and disabled debugging interfaces [22]. Cloud and edge environments should also follow Zero Trust principles, ensuring continuous identity verification and policy enforcement even post-deployment.

3.6 Maintenance Phase: Ongoing Protection

Security is not a one-time effort as it requires regular updates and monitoring throughout the system's lifetime. In IIoT, many attacks like botnets, firmware tampering, and supply chain threats emerge after deployment and evolve over time. Key maintenance strategies include Over-the-air (OTA) secure updates with rollback support, Security audits and patch management, Log analysis, anomaly and revisiting threat models (e.g., STPA-Sec) after system changes. This phase also includes end-of-life planning, where decommissioned IIoT devices must be safely retired without leaking data or credentials [23].

Each phase of SSDLC when properly adapted, offers an opportunity to reduce vulnerabilities before they are exploited in the field [24]. By combining established software engineering practices with IIoT-specific constraints and threat models, developers can design, build, and maintain software that is more resilient, secure, and ready for real-world industrial environments. A flowchart presenting the phases of SSDLC based on IIoT-specific security practices is presented in Figure 3.

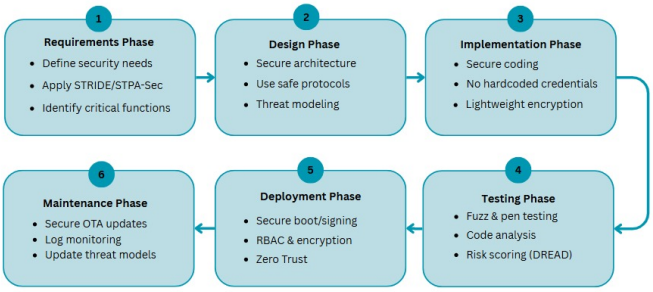

Figure 3. Phases of SSDLC based on IIoT-specific security practices.

4 Secure development process in IIoT Software Delivery

As IIoT systems become increasingly complex and distributed, ensuring their security after deployment is no longer sufficient. Modern industrial software development must embrace secure development process, an approach that integrates security practices directly into the development and delivery pipeline. Secure development process encourages "security as code", enabling faster, safer, and more reliable software releases by automating testing, monitoring, and policy enforcement from the very beginning [25].

In IIoT systems, this integration is especially important due to the safety-critical nature of operations, limited physical access to devices, and heterogeneity of platforms. Unlike traditional IT applications, IIoT deployments must consider not just cloud or mobile environments but also edge and fog nodes, where deployment and updates can be risky and resource-intensive.

4.1 Integrating Security into CI/CD Pipelines

Continuous Integration and Continuous Deployment (CI/CD) is central to any secure development process pipeline. In IIoT development, these pipelines can be extended with tools that automatically scan identified vulnerabilities. These tools support code scanning and help developers follow secure coding standards [26]. The process includes running Static Application Security Testing during code commits and Dynamic Application Security Testing before releasing the software. Builds that contain major security issues or outdated libraries are stopped automatically. This helps detect risks early and ensures that only trusted code is moved forward to production.

4.2 Infrastructure as Code (IaC) and Security-as-Code

Handling infrastructure by hand becomes difficult in IIoT environments, particularly in setups that include many sensors, actuators, edge nodes, and control units. Infrastructure as Code (IaC) helps define configurations, network rules, and security settings through scripts and templates. Security-as-Code allows teams to apply rules for encryption, firewall setup, access permissions, and compliance requirements. This approach helps keep deployments uniform, easier to check, and less likely to include mistakes during setup and updates [27].

4.3 Secure Software Deployment on Edge and Fog Devices

A common issue in IIoT involves installing software on edge and fog devices with limited resources. These devices usually operate on real-time systems or minimal Linux versions and often do not include full operating system security features. To support secure secure development process, in this environment minimal, signed container images (e.g., using Docker or Balena for embedded) should be used to prevent tampering. Moreover, the use of hardware-based root of trust, such as TPM (Trusted Platform Module), can ensure only authenticated software is executed. Additionally, enabling immutable infrastructure techniques—where changes are made via full re-deployments rather than ad hoc patches—can improve auditability and rollback safety. These practices reduce the risk of deploying compromised software or leaving vulnerable components active in the field [26].

4.4 Monitoring, Drift Detection, and Policy Enforcement

Security continues even after the software has been deployed. Secure development process supports ongoing monitoring and policy enforcement to keep IIoT systems protected. One important task in this stage is detecting configuration drift. This helps identify any changes in device settings or network rules that were not approved. Moreover, log aggregation and threat detection, using tools like ELK stack, OSSEC, or cloud-based security monitoring. Additionally, Policy-as-code should be introduced, allowing organizations to define and enforce security rules that are automatically applied and verified during deployments. These strategies make it possible to respond to evolving threats and maintain compliance even in distributed and dynamic

IIoT networks [27].

4.5 Privacy-Aware Automation in Industrial Pipelines

Many IIoT systems collect sensitive operational data, some of which may include employee movement, machine health, or customer behavior. Integrating privacy-aware automation into secure development process practices ensures that personal and confidential data is protected during processing, storage, and transfer. This includes Data masking and pseudonymization before transfer to cloud analytics engines. Moreover, secure key management using vault systems like HashiCorp Vault or AWS KMS should also be focused on. Additionally, ensuring compliance with data protection standards such as GDPR and ISO/IEC 27001 would help multinational industrial deployments [28]. By embedding these capabilities into the pipeline, privacy can be enforced just like functional or performance requirements. The secure development process pipeline adapted for IIoT systems is presented in Figure 4. It illustrates the security integration from the coding phase to monitoring phase.

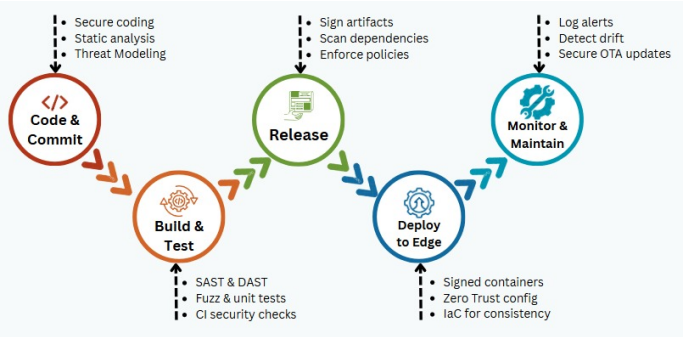

Figure 4. A secure development process workflow customized for Industrial IoT systems.

5 Proposed Framework: SecureDev-IIoT

The complexity of IIoT systems demands a structured approach that combines threat modeling, secure software engineering, and advanced secure development process. We propose a conceptual development framework named SecureDev-IIoT that is designed to guide industrial teams to build secure-by-design IIoT systems. This framework ensures appropriate security techniques across all stages of SDLC. Additionally, it also incorporates automation and continuous validation through secure development process. The following section briefly describes the overview of the proposed framework, its lifecycle flow, and its architecture.

5.1 Framework Overview

The SecureDev-IIoT framework is built on three layers that work together to improve software security in industrial environments. The first layer deals with threat modeling. This step helps teams identify risks early in the development cycle. These methods are applied during the planning and design stages to reveal possible entry points in IIoT systems. The findings guide design decisions, support risk prioritization, and help form a stronger security foundation before coding begins.

The second layer improves development by including activities that strengthen security in areas specific to IIoT. Each phase involves actions such as applying control checks, writing secure code, and performing stress tests. These additions help ensure that security is considered across the full process instead of limiting it to a single stage. The third layer introduces automation using secure development process. It supports the application of security policies during both development and deployment. Tasks in this layer involve scanning source code, identifying vulnerabilities, and validating configurations within the CI/CD workflow. This setup helps teams produce secure software builds. The same process supports secure deployment across edge and fog systems. Continuous monitoring maintains compliance and provides protection against new threats that may appear after release.

The SecureDev-IIoT framework introduces a unique layered integration of three well-established threat modeling techniques—STRIDE, DREAD, and STPA-Sec within the structure of the SDLC. Unlike earlier approaches that apply a single technique at a fixed stage, this method distributes each technique based on its strengths across different SDLC phases. STPA-Sec is employed during the requirement phase to analyze unsafe control actions in industrial control environments. STRIDE supports the design phase by helping identify threats in system architecture, data flows, and interface interactions. During testing and validation, DREAD is used to score and prioritize identified threats based on their potential impact. This multi-technique approach ensures both proactive and reactive security coverage, which is particularly essential in the complex, high-risk environment of IIoT systems. By combining these methods in a coordinated way, the framework offers a more complete and adaptable threat assessment process across the entire software lifecycle—something not addressed in earlier models.

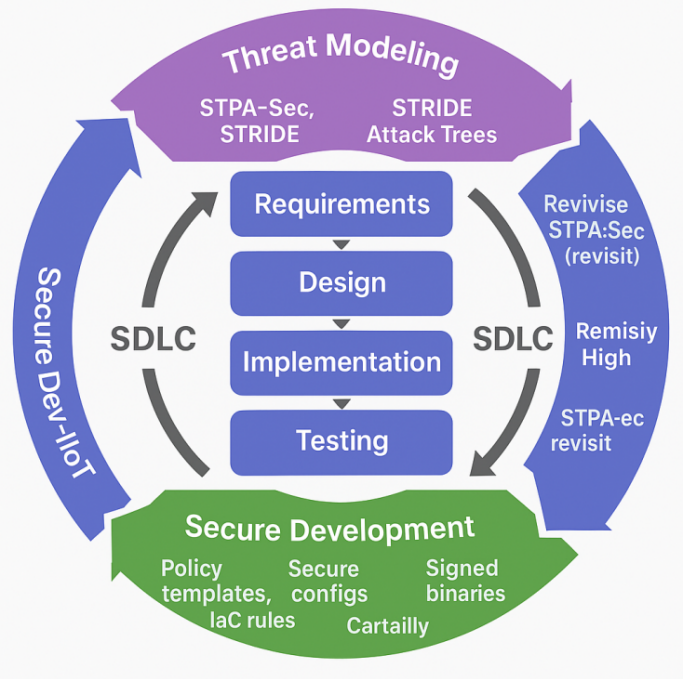

Figure 5. SecureDev-IIoT lifecycle, illustrating threat modeling and secure development activities.

Figure 5 shows the SecureDev-IIoT lifecycle, illustrating the threat modeling and secure development activities aligning with each phase of the SSDLC in a continuous, circular process.

While the framework is methodology-focused, parts of it can be implemented using widely available tools. For example, Microsoft Threat Modeling Tool supports STRIDE analysis, while OWASP Threat Dragon can assist in modeling attack surfaces. Secure coding and static analysis are supported through tools like SonarQube and CodeQL, and DevOps automation can be configured using platforms like GitHub Actions or Jenkins for CI/CD integration [29–31].

5.2 SecureDev-IIoT Lifecycle Flow

The SecureDev-IIoT framework is divided into six phases. Each phase includes specific security activities designed for IIoT systems. In the first phase, STPA-Sec is used to examine system behaviors that may cause unsafe conditions. At the same time, STRIDE is applied to identify security concerns related to data exchange, system interfaces, and communication protocols. These methods help teams address both safety and security from the early stages.

The following phase focuses on system design. Developers build the structure of the system and use tools such as attack trees and threat graphs to explore possible intrusion paths. Secure protocols are applied, and legacy components are kept separate from critical functions to reduce potential exposure. The third phase is about writing secure code. Developers follow industry rules, such as MISRA for embedded software, and scan their code to catch issues early. Simple encryption is also added to keep data safe, even on devices with limited resources.

The fourth phase involves testing. Teams run unit and integration tests along with security checks such as fuzzing and penetration testing. DREAD scoring helps them decide which issues are more serious and need attention first. The fifth phase focuses on putting the software into real environments. Signed files are used to ensure software has not been changed. Access is controlled through RBAC, and every connection is checked using Zero Trust methods.

In the final phase, the system continues to stay protected after it goes live. Secure OTA updates are used to fix issues quickly. Logs are reviewed, and unusual behavior is flagged. Threat models are also updated over time, so security stays strong even as the system grows and changes.

5.3 Architecture of SecureDev-IIoT

The architecture of SecureDev-IIoT is aimed at applying specific security methodologies in each phase of SDLC. This framework integrates security techniques across all the stages of SDLC. It supports security from initial requirements analysis to final maintenance phase. The main goal is to incorporate proactive and reactive security approaches considering the dynamic nature of attack surface and IIoT limitations.

In the initial phase of requirement gathering, various techniques such as STRIDE and STPA-Sec identify threats and unsafe control actions ahead of software development phase. In the design phase, developers use tools such as attack trees and secure API guidelines to create systems that are both safe and scalable. The next phase begins with implementation. At this stage, developers apply secure coding practices and run static analysis tools to find possible security issues in the code.

The testing phase includes techniques such as fuzzing and penetration testing. DREAD scoring is used to help teams focus on fixing the most serious issues first. In the deployment phase, security involves checking the build using signed files and applying Infrastructure as Code to enforce system settings. During the final phase, maintenance, secure Over The Air (OTA) updates are used to keep the system reliable. These

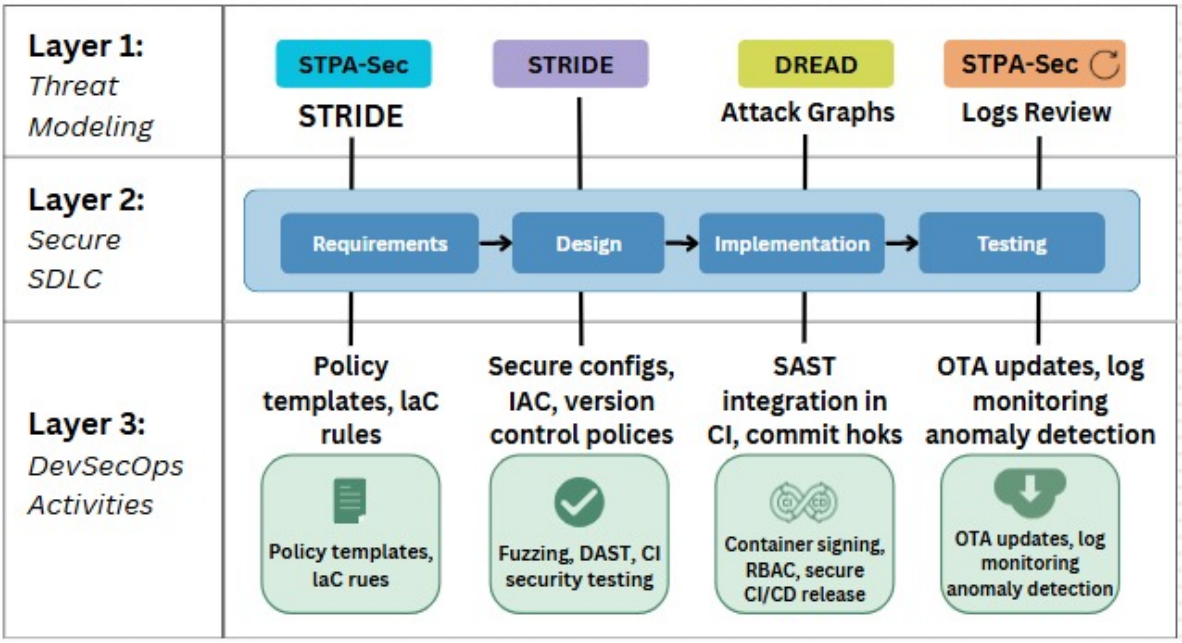

Figure 6. SecureDev-IIoT: A layered framework combining threat modeling, SSDLC, and secure development process.

updates also support clear and traceable actions in systems where safety is a priority. The full framework, including threat modeling and secure development activities is shown in Figure 6.

The phase-wise mapping of SDLC to the applied techniques and security goals is summarized in Table 2.

Table 2. SDLC phases alongside the techniques employed and their respective security goals.

Phase	Technique applied	Security goal
Requirements	STRIDE, STPA-Sec	Identify early risks, define secure goals
Design	Attack Trees, Secure APIs	Architect resilient systems
Implementation	Secure Coding, Static Analysis	Eliminate common vulnerabilities
Testing	DREAD, Fuzzing, SAST/DAST	Validate security robustness
Deployment	Signed Containers, RBAC, IaC	Secure and consistent releases
Maintenance	OTA Updates, Drift Detection	Ensure long-term system integrity

The SecureDev-IIoT framework is adaptable across various industrial domains. In energy systems, it helps protect SCADA and grid communication. In healthcare, it supports secure updates for connected devices. In manufacturing, it ensures control logic integrity and protects Programmable Logic Controllers (PLCs). This flexibility allows the framework to be reused across sectors without major changes.

6 Evaluation and Discussion

The SecureDev-IIoT framework addresses the specific requirements and constraints of IIoT systems while supporting secure software engineering practices along with threat modeling. The following section will examine its advantages and limitations by comparing it to existing security mechanisms.

Existing IIoT security approaches often focus on runtime protection by employing intrusion detection systems, encryption protocols, or access control mechanisms [32]. These techniques are reactive and may fail in preventing vulnerabilities that were introduced during initial stages of SDLC. While SecureDev-IIoT is proactive as it promotes the early identification of risks and mitigates them during the requirements and design phases. It also reduces the likelihood of crucial errors reaching to production stage. The key differences between SecureDev-IIoT and traditional runtime-based security approaches are highlighted in Table 3.

The key strengths of SecureDev-IIoT lie in its

Table 3. Comparison of secureDev-IIoT with runtime security approaches.

Aspect	Traditional IIoT Security	SecureDev-IIoT Framework
Focus	Detection and mitigation at runtime	Prevention and secure-by-design development
Lifecycle Coverage	Mostly post-deployment	Full SSDLC integration
Threat Modeling	Rarely applied	Applied early and throughout
Automation (CI/CD, IaC)	Limited	Fully integrated via secure development process
Customization for IIoT	Often generic	Tailored to IIoT constraints
Maintenance Support	Patch-driven, reactive	OTA updates, drift detection, secure logging

comprehensive coverage, lightweight applicability, and process adaptability. Specifically, it can be applied in both large-scale industrial environments and small, embedded IIoT projects. It also aims to support compliance with international standards such as IEC 62443, ISO/IEC 27001, and GDPR, making it suitable for global deployment. Moreover, the framework is modular and compatible with Agile and DevOps, which helps industrial teams adopt security gradually without disrupting existing workflows. Additionally, because the framework is technique-agnostic, organizations can plug in their preferred tools (e.g., SonarQube, OpenVAS, HashiCorp Vault) without being locked into specific vendors.

The SecureDev-IIoT framework is adaptable across a variety of industrial domains. In smart manufacturing and energy sectors, it supports secure PLC updates, control logic integrity, SCADA protection, and grid monitoring. It also benefits healthcare IoT and supply chain automation by securing connected medical devices, asset tracking systems, and RFID infrastructure. In all these domains, the need for early risk identification, safe deployment, and reliable maintenance is consistent, validating the generalizability of the proposed approach.

Although SecureDev-IIoT ensures secure IIoT systems, however, its implementation has certain challenges. Introducing threat modeling and secure coding practices in early phases of SDLC could delay project initiation [28]. Additionally, development teams need to have familiarity with threat modeling techniques, hence, their appropriate training is required. Moreover, the integration of security tools into secure development process can be complicated supplementary delaying the project timeline. Furthermore, the success of the proposed framework is dependent on the maturity of the organization, i.e., the willingness of the resources to employ SSDLC principles. Strong project management and security leadership is also required so that development teams may not opt for short-term development shortcuts while bypassing SSDLC protocols.

As security tools continue to evolve, future platforms may offer tighter integration between threat modeling, code analysis, and deployment pipelines. For example, automated mapping of threat models to CI/CD policies or AI-powered vulnerability detection could reduce manual overhead and improve response times. These advancements would help overcome current limitations in scalability and ease of use, especially in complex IIoT environments with frequent device changes and software updates.

6.1 Practical Scenario Illustration:

Consider a smart manufacturing plant that uses PLCs to control robotic arms and conveyor systems. During the requirements phase, STPA-Sec is applied to identify control actions that may lead to unsafe operations, such as unauthorized speed adjustments. STRIDE helps analyze data flows between PLCs and central systems to uncover spoofing or tampering threats. DREAD scoring is used during testing to rank the severity of detected vulnerabilities, ensuring that high-risk issues are addressed first. Secure development automation supports regular secure firmware updates on edge devices while maintaining compliance. This scenario shows how SecureDev-IIoT can be applied step by step in a realistic industrial setting to strengthen both safety and security.

7 Conclusion

The industrial IoT systems are essential across various sectors; hence, they require secure, robust, and

maintainable software. This study presented a comprehensive review of multiple threat modeling techniques, secure SDLC practices, and secure development process to systematically integrate them into the IIoT lifecycle. We proposed a conceptual framework called SecureDev-IIoT that supports threat analysis with SSDLC phases and automates security implementation through modern CI/CD pipelines. SecureDev-IIoT helps organizations build secure software by integrating security in all phases of SDLC, hence proactively providing consistent shield against evolving threats. The framework proposed a flexible foundation that could be adapted across diverse industrial domains. In future, the work will aim to extend the framework through toolchain support and also validate it in real-world IIoT environments including smart manufacturing and energy sectors.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported without any funding.

Conflicts of Interest

The authors declare no conflicts of interest.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Hou, K. M., Diao, X., Shi, H., Ding, H., Zhou, H., & de Vaulx, C. (2023). Trends and challenges in AIoT/IIoT/IoT implementation. *Sensors, 23*(11), 5074. [Crossref]

[2] Sheng, C., Zhou, W., Han, Q. L., Ma, W., Zhu, X., Wen, S., & Xiang, Y. (2025). Network traffic fingerprinting for IIoT device identification: A survey. *IEEE Transactions on Industrial Informatics*. [Crossref]

[3] Bahaa, A., Abdelaziz, A., Sayed, A., Elfangary, L., & Fahmy, H. (2021). Monitoring real time security attacks for IoT systems using DevSecOps: a systematic literature review. *Information, 12*(4), 154. [Crossref]

[4] De Oliveira, G. W., Nogueira, M., dos Santos, A. L., & Batista, D. M. (2023). Intelligent VNF placement to mitigate DDoS attacks on industrial IoT. *IEEE Transactions on Network and Service Management, 20*(2), 1319-1331. [Crossref]

[5] Sarjan, H., Ameli, A., & Ghafouri, M. (2022). Cyber-security of industrial internet of things in electric power systems. *IEEE Access, 10*, 92390-92409. [Crossref]

[6] Kavitha, D., & Thejas, S. (2024). Ai enabled threat detection: Leveraging artificial intelligence for advanced security and cyber threat mitigation. *IEEE Access*. [Crossref]

[7] Khan, R. A., Khan, S. U., Akbar, M. A., & Alzahrani, M. (2024). Security risks of global software development life cycle: Industry practitioner's perspective. *Journal of Software: Evolution and Process, 36*(3), e2521. [Crossref]

[8] Barrera, D., Bellman, C., & Van Oorschot, P. (2023). Security best practices: a critical analysis using IoT as a case study. *ACM Transactions on Privacy and Security, 26*(2), 1-30. [Crossref]

[9] Ali, A., Husain, M., & Hans, P. (2025). Federated Learning-Enhanced Blockchain Framework for Privacy-Preserving Intrusion Detection in Industrial IoT. *arXiv preprint arXiv:2505.15376*.

[10] Crothers, E. N., Japkowicz, N., & Viktor, H. L. (2023). Machine-generated text: A comprehensive survey of threat models and detection methods. *IEEE Access, 11*, 70977-71002. [Crossref]

[11] Ali, M., Raza, A., Akram, M. A., Arif, H., & Ali, A. (2025). Enhancing IOT Security: A review of Machine Learning-Driven Approaches to Cyber Threat Detection: Enhancing IOT Security: A review of Machine Learning-Driven Approaches to Cyber Threat Detection. *Journal of Informatics and Interactive Technology, 2*(1), 316-324. [Crossref]

[12] Benmalek, M. (2024). Ransomware on cyber-physical systems: Taxonomies, case studies, security gaps, and open challenges. *Internet of Things and Cyber-Physical Systems, 4*, 186-202. [Crossref]

[13] Kim, K. H., Kim, K., & Kim, H. K. (2022). STRIDE-based threat modeling and DREAD evaluation for the distributed control system in the oil refinery. *ETRI Journal, 44*(6), 991-1003. [Crossref]

[14] Yu, J., Wagner, S., & Luo, F. (2021). Data-flow-based adaption of the system-theoretic process analysis for security (STPA-sec). *PeerJ Computer Science, 7*, e362. [Crossref]

[15] Mohanty, R. K., Padmaja, C. V. R., Kanaparthi, S. K., & Rajan, A. (2025). Unified threat modeling: Strategies for comprehensive risk assessment in modern systems. In *Integrating Technology in Problem-Solving Educational Practices* (pp. 429-450). IGI Global. [Crossref]

[16] He, P., Du, X., Li, Y., Guo, H., & Cui, J. (2025). An integration methodology of safety and security requirements for autonomous vehicles. *Journal of Transportation Safety & Security, 17*(3), 253-271. [Crossref]

[17] Alauthman, M., Al-Qerem, A., Aldweesh, A., & Almomani, A. (2025). Secure SDLC Frameworks: Leveraging DevSecOps to Enhance Software Security. In *Modern Insights on Smart and Secure Software Development* (pp. 77-118). IGI Global Scientific

Publishing. [Crossref]

[18] Yu, Z., Gao, H., Cong, X., Wu, N., & Song, H. H. (2023). A survey on cyber–physical systems security. *IEEE Internet of Things Journal, 10*(24), 21670-21686. [Crossref]

[19] Rathee, G., Ahmad, F., Jaglan, N., & Konstantinou, C. (2022). A secure and trusted mechanism for industrial IoT network using blockchain. *IEEE Transactions on Industrial Informatics, 19*(2), 1894-1902. [Crossref]

[20] Hameed, A., Violos, J., & Leivadeas, A. (2022). A deep learning approach for IoT traffic multi-classification in a smart-city scenario. *IEEE Access, 10*, 21193-21210. [Crossref]

[21] Ajiga, D., Okeleke, P. A., Folorunsho, S. O., & Ezeigweneme, C. (2024). Designing cybersecurity measures for enterprise software applications to protect data integrity. *Computer Science & IT Research Journal, 5*(8), 1920-1941. [Crossref]

[22] Akerele, J. I., Uzoka, A., Ojukwu, P. U., & Olamijuwon, O. J. (2024). Increasing software deployment speed in agile environments through automated configuration management. *International Journal of Engineering Research Updates, 7*(02), 028-035. [Crossref]

[23] Mustonen, J. (2024). Designing a security framework for enhanced monitoring and secure development during the software life cycle.

[24] Ali, M., Mazhar, T., Al-Rasheed, A., Shahzad, T., Ghadi, Y. Y., & Khan, M. A. (2024). Enhancing software defect prediction: a framework with improved feature selection and ensemble machine learning. *PeerJ Computer Science, 10*, e1860. [Crossref]

[25] Padmapriya, V. M., Thenmozhi, K., Hemalatha, M., Thanikaiselvan, V., Lakshmi, C., Chidambaram, N., & Rengarajan, A. (2025). Secured IIoT against trust deficit-A flexi cryptic approach. *Multimedia Tools and Applications, 84*(9), 5625-5652. [Crossref]

[26] Lalar, S., Kumar, T., Kamboj, S., & Kumar, R. (2024). Security challenges and solutions in cloud, fog, and edge computing for sustainable development. In *Cloud and Fog Optimization-based Solutions for Sustainable Developments* (pp. 178-200). CRC Press.

[27] Veldi, S. R. (2025). Infrastructure-as-Code with Scripting: A Technical Review. *Journal of Computer Science and Technology Studies, 7*(6), 345-352. [Crossref]

[28] Reyes-Acosta, R. E., Mendoza-González, R., Oswaldo Diaz, E., Vargas Martin, M., Luna Rosas, F. J., Martínez Romo, J. C., & Mendoza-González, A. (2025). Cybersecurity Conceptual Framework Applied to Edge Computing and Internet of Things Environments. *Electronics, 14*(11), 2109. [Crossref]

[29] Hwang, I., Cho, H., & Kim, S. (2025). Deriving Usability Evaluation Criteria for Threat Modeling Tools. *IEEE Access*. [Crossref]

[30] Bar, K. (2025). AI for Code Synthesis: Can LLMs Generate Secure Code?. *Available at SSRN 5157837*. [Crossref]

[31] Gajera Jr, A. (2025). Comparative Analysis of Jenkins, GitLab CI, and GitHub Actions: Performance Evaluation in CI/CD Pipelines.

[32] Khan, I. A., Keshk, M., Pi, D., Khan, N., Hussain, Y., & Soliman, H. (2022). Enhancing IIoT networks protection: A robust security model for attack detection in Internet Industrial Control Systems. *Ad Hoc Networks, 134*, 102930. [Crossref]

Misbah Ali is a PhD scholar at COMSATS university Islamabad, with research interests in Machine Learning, Deep Learning, Generative Artificial Intelligence, and Software Engineering. Her work focuses on the development of secure, intelligent systems across domains such as healthcare, education, and industrial cyber-security. She has authored multiple peer-reviewed publications and presented her research at international conferences. She also contributes to the academic community as a reviewer for several reputed journals. (Email: talktomisbah.ali@gmail.com)

Haroon ARIF received his Bachelor's degree in Computer Science from Preston University, Islamabad, and his Master's degree in Cybersecurity from the Illinois Institute of Technology, Chicago. He is a cybersecurity professional and researcher with expertise in cloud security, threat intelligence, post-quantum cryptography, and AI-driven threat detection.He has published multiple papers on topics such as AI-enhanced cloud security, dynamic cryptographic algorithm selection, and resilient enterprise architectures. His research aims to develop innovative, data-driven solutions for securing modern digital infrastructures. (Email: harif@hawk.iit.edu)

Aamir Raza is currently pursuing a Master's degree in the Department of Information Technology and Management at the Illinois Institute of Technology, Chicago, USA. His research interests include Industrial IoT security, secure software development, DevSecOps practices, and threat modeling. He is particularly focused on integrating software engineering principles with modern cybersecurity approaches to build resilient systems in industrial environments. (Email: araza7@hawk.iit.edu)

Moomna Nazir holds a Bachelor's degree in Computer Science from COMSATS University Islamabad, Pakistan. Her academic interests focus on machine learning, IoT systems, and software design. She is passionate about applying practical security methods to real-world computing environments and continues to explore research opportunities in the field of emerging technologies. (Email: moomnanazir@gmail.com)