eBook

Kontron AIS GmbH


Secure and Flexible IoT Device Management:

# Build or Buy?

Vanessa Kluge, Product Manager at Kontron AIS GmbH

Facing the Build vs. Buy Decision?

This eBook is your comprehensive guide to making an informed choice between developing your own IoT device management solution and purchasing a ready-made one. We cover all the key factors: How do you ensure maximum security and compliance for critical infrastructures? Which solution best aligns with your scaling plans and technical requirements? With practical checklists and real-world examples, this eBook helps you make a well-informed, strategic decision.

2025, January

eBook

# Table of Contents


Secure and flexible IoT device management: build or buy? 3

01 Which solution is right for your business? 4
- 1.1 Build (in-house development) 4
- 1.2 Buy (third-party solution) 5

02 Effort estimation for an in-house solution 7
- 2.1 Development costs 7
- 2.2 Infrastructure costs 8
- 2.3 Maintenance and support costs 8
- 2.4 Security implementation 9
- 2.5 Scalability planning 9
- 2.6 Total cost estimation (first year) and time estimate 10
- 2.7 Conclusion 11

03 Checklist: key steps and resources for secure and scalable device management 11

04 Technical requirements and solutions 15
- 4.1 How IoT devices connect 15
- 4.2 How to deploy an application to the device 17
- 4.3 How can Kontron hardware and software help? 17
- 4.4 How can IoT devices be securely maintained remotely? 21

05 Cybersecurity requirements for digital products and solutions 22
- 5.1 Security by design 23
- 5.2 Security by default 24
- 5.3 Differences, role distribution, and distinctions 26
- 5.4 Security aspects of KontronOS and KontronGrid 29

06 Why make it complicated, when it can be simple? 37

07 Checklist and decision support – build or buy? 39

08 Conclusion: „build or buy“ as a key question 41

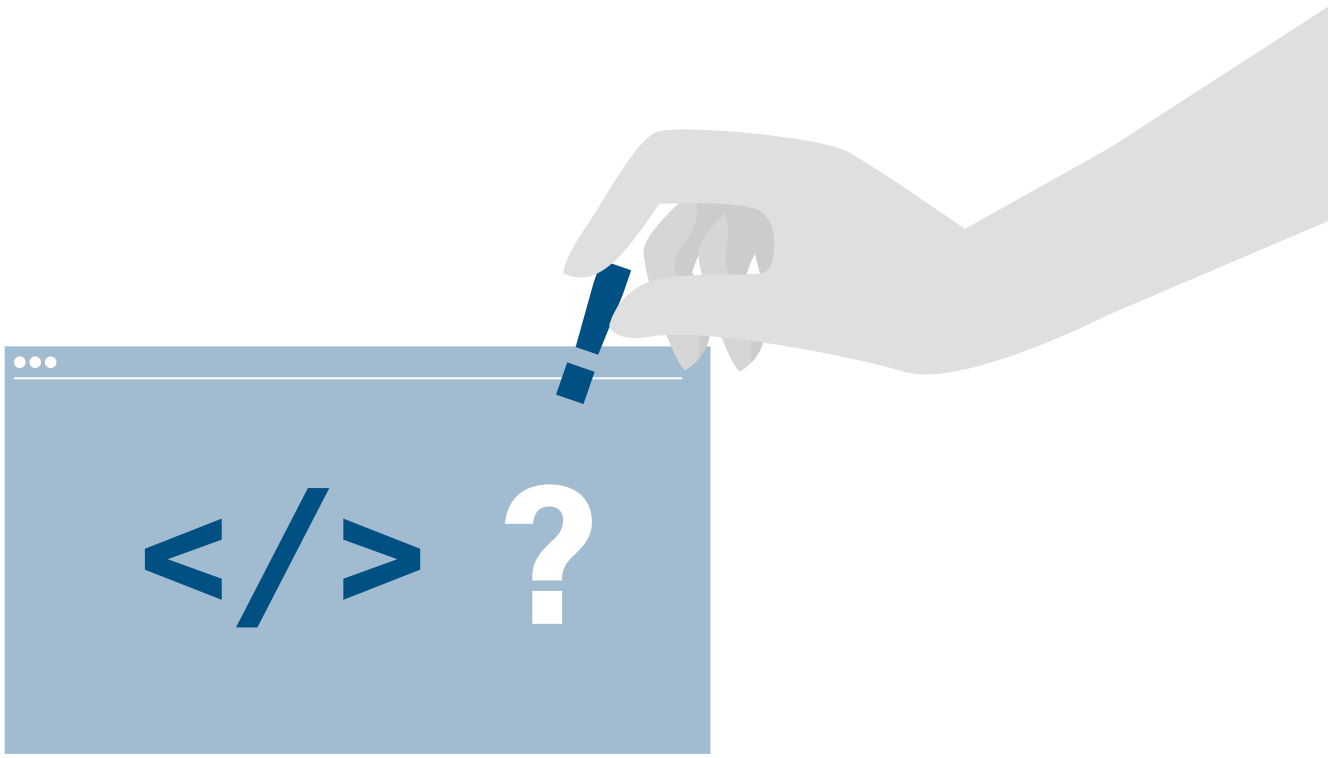
# Secure and flexible IoT device management: build or buy?

The Internet of Things (IoT) is evolving at a rapid pace – bringing with it new challenges for IT teams and software developers. One of the most significant of these challenges is the efficient management of edge devices, which are frequently deployed in large volumes, harsh environments, or remote areas. These devices form the backbone of modern IoT architectures, yet managing them is a delicate balancing act between security, scalability, and efficiency.

A central challenge is ensuring secure and reliable remote access to these devices. Since many IoT devices serve as gateways to machine and customer networks, management solutions must meet strict compliance requirements. Data security and system integrity are top priorities, making the development and implementation of security measures a complex task.

So, the question is: Should you build a custom solution tailored to your specific needs? Or should you leverage a proven, off-the-shelf solution? Both approaches have their pros and cons, and weighing factors like cost, time investment, resource availability, and scalability is crucial.

This eBook takes a deep dive into the challenges of managing edge devices and provides a detailed comparison of in-house development versus adopting a market-ready solution. The goal is to equip IT decision-makers with the insights they need to choose the best approach for their unique requirements.



# 1 Which solution is right for your business?

Build or Buy? There's no one-size-fits-all answer.
Each option comes with its own set of advantages and challenges, and the right choice depends entirely on your unique requirements, resources, and priorities. To give you a clear overview, here are the key arguments for both approaches.

## 1.1 Build (in-house development)

### Advantages

**Customization**
A custom-built solution can be tailored precisely to your company's needs, offering greater flexibility in terms of features and functionality. Seamless integration with existing systems ensures the software is perfectly aligned with your IT environment.

**Control**
Your company retains full control over the entire development process, security protocols, and updates. Intellectual property (IP) stays in-house, which can be a competitive advantage.

**Scalability**
The solution can grow alongside your business without being restricted by the limitations of a third-party provider.

**Security**
You can implement your own security standards and protocols to ensure sensitive data is managed safely and in compliance with internal policies.

**Potential long-term cost savings**
If used across multiple projects over an extended period, an in-house solution can become more cost-effective in the long run.

### Challenges

**High upfront costs**
Development costs can be significant, including hiring specialized developers, purchasing development tools, and allocating resources.

**Specialized expertise required**
Building a complex IoT fleet management solution requires a team with highly specific technical skills, which may be difficult to recruit or expensive to retain. Security requirements are constantly evolving and must be considered from the early development phase.

**Time-consuming**
Developing a solution from scratch takes time, potentially delaying implementation and putting your competitive edge at risk.

**Ongoing maintenance effort**
Continuous maintenance, updates, and troubleshooting must be handled internally, requiring dedicated teams and resources.

**Risk of failure**
Every development project carries the risk of failure. If the solution doesn't work as expected, it can result in wasted resources and increased costs. This risk is especially high for companies unfamiliar with agile software development practices.

## 1.2 Buy (third-party solution)

### Advantages

**Fast implementation**
Third-party solutions are typically ready to use, allowing for quick deployment and minimizing downtime.

**Lower upfront costs**
Initial costs are often lower since there's no need for lengthy development and testing phases.

**Proven technology**
Market solutions are already tested and established, reducing the risk of technical issues.

**Regular updates**
Vendors provide frequent updates, security patches, and new features, ensuring that the solution remains up to date.

**Support and service**
Comprehensive support services mean that issues can be quickly resolved by experienced professionals, and guided onboarding helps ensure a smooth transition.

### Challenges

**Limited customization**
Off-the-shelf solutions may not meet every company-specific requirement, leading to compromises in functionality or performance.

**Vendor dependence**
Relying on a vendor for updates, support, and further development can be risky, especially if the provider changes direction or discontinues the product.

**Potential for higher long-term costs**
Subscription fees, licensing costs, and additional charges for extra users or features can add up over time, potentially making the solution more expensive in the long run.

**Integration challenges**
Integrating a third-party solution with existing systems and workflows can be complex, sometimes requiring additional resources or middleware.

Companies with specific requirements, substantial financial resources, and strong technical expertise often lean toward in-house development. However, if speed and core functionality are the priority, a proven off-the-shelf solution is usually the better choice.

**At a glance:**

| Build (in-house development) | | Buy (third-party solution) |
|---|---|---|
| High | Customization | Low to medium |
| Full | Control | Limited |
| Tailored | Scalability | Standardized |
| Internal security standards | Security | Vendor-independent |
| High | Upfront costs | Low |
| Long | Implementation time | Short |
| Internal | Maintenance required | Externally supported |
| Risk of development failure | Risk | Vendor-dependent risk |

# 2 Effort estimation for an in-house solution

Developing an in-house IoT device management solution for edge devices is a strategic decision that requires significant investments in time, money, and personnel. To make an informed choice, it's essential to realistically assess the potential costs and resource requirements.

The following overview outlines the key cost factors, based on typical salary and cost structures in Europe, for managing an initial fleet size of 100 devices in the first year, with an increase of 100 additional devices per year.

## 2.1 Development costs

Development costs include expenses for software development, hiring personnel, project management, and related expenditures.

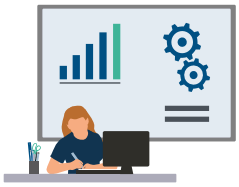
### Team composition:

**Project manager:**

- 1 FTE (Full-Time Equivalent)
- Estimated annual salary: € 80,000 - € 100,000

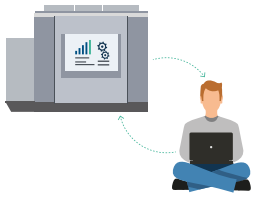
**Software developers:**

- 3 - 5 FTEs (front-end, back-end, and edge-computing-experts)
- Estimated annual salary per developer: € 60,000 - € 90,000



**UI / UX designer:**

- 1 FTE
- Estimated annual salary: € 50,000 - € 70,000



**QA / Test engineers:**

- 1 - 2 FTEs
- Estimated annual salary per QA engineer € 50,000 - € 70,000

**Security specialist:**

- 1 FTE
- Estimated annual salary: € 70,000 - € 100,000

**Time estimate:**

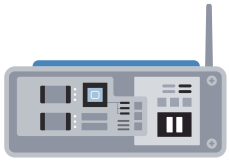
- 9 - 12 months (initial development)

**Total development costs:**

- € 580,000 - € 860,000 (first year)
- € 120,000 - € 180,000 per year (ongoing development costs)

## 2.2 Infrastructure costs

Infrastructure costs include expenses for hardware, cloud services, and other essential infrastructure components.



**Hardware:**

- € 70,000 - € 120,000 for edge devices, servers, and networking equipment*

  * Includes 100 edge devices at € 500 each, minimum € 25,000 for initial server costs

**Cloud services:**

- €78,600 - € 165,000* for computing power, storage, networking (e. g., AWS, Azure, Google Cloud), and analytics / monitoring

  * Costs vary depending on data processing complexity

**Development tools and licenses:**

- ~ € 15,000 for software licenses and development environments

**Total Infrastructure Costs:**

- ~ € 164,000 - € 300,000 (first year)
- € 50,000 - € 110,000 (ongoing infrastructure costs)

## 2.3 Maintenance and support costs

Ongoing maintenance, troubleshooting, updates, and user support must also be factored in.

**Ongoing maintenance team:**

- 2 developers: € 60,000 - € 90,000 per year per developer
  (To reduce dependencies, a team of three developers is recommended.)
- 1 - 2 support staff: € 40,000 - € 60,000 per year per person

**Annual maintenance costs:**

- € 160,000 - € 300,000

## 2.4 Security implementation

Ensuring the security of an IoT device management solution requires both initial and ongoing investments across multiple layers:

- **Software:** Security audits, encryption, secure protocols, regular updates, access controls
- **Operating System (included):** Hardened OS versions, Secure Boot
- **Hardware (partially included):** TPM/HSM for root-of-trust, physical tamper protection, key management

**Initial security implementation:**

- Security audits: € 40,000 - € 80,000
- Encryption and secure communication protocols: € 25,000 - € 60,000
- Security tools/software licenses: € 15,000 - € 40,000

**Ongoing security monitoring:**

- Security specialist: € 40,000 - € 60,000 per year (allocated for 100 devices)
- Security tools: € 15,000 - € 40,000 per year
- Security audits: 20.000 - 40.000 per year

**Total security costs (first year):**

- Initial security implementation: € 80,000 - € 180,000
- Ongoing security costs: € 75,000 - € 140,000 per year

## 2.5 Scalability planning

Scalability planning includes preparing for future growth, ensuring the solution can handle an increasing number of edge devices.

**Scalability design and implementation:**

- Architectural planning: € 40,000 - € 80,000
- Cloud infrastructure scaling costs: € 40,000 - € 120,000 per year

**Total scalability costs:**

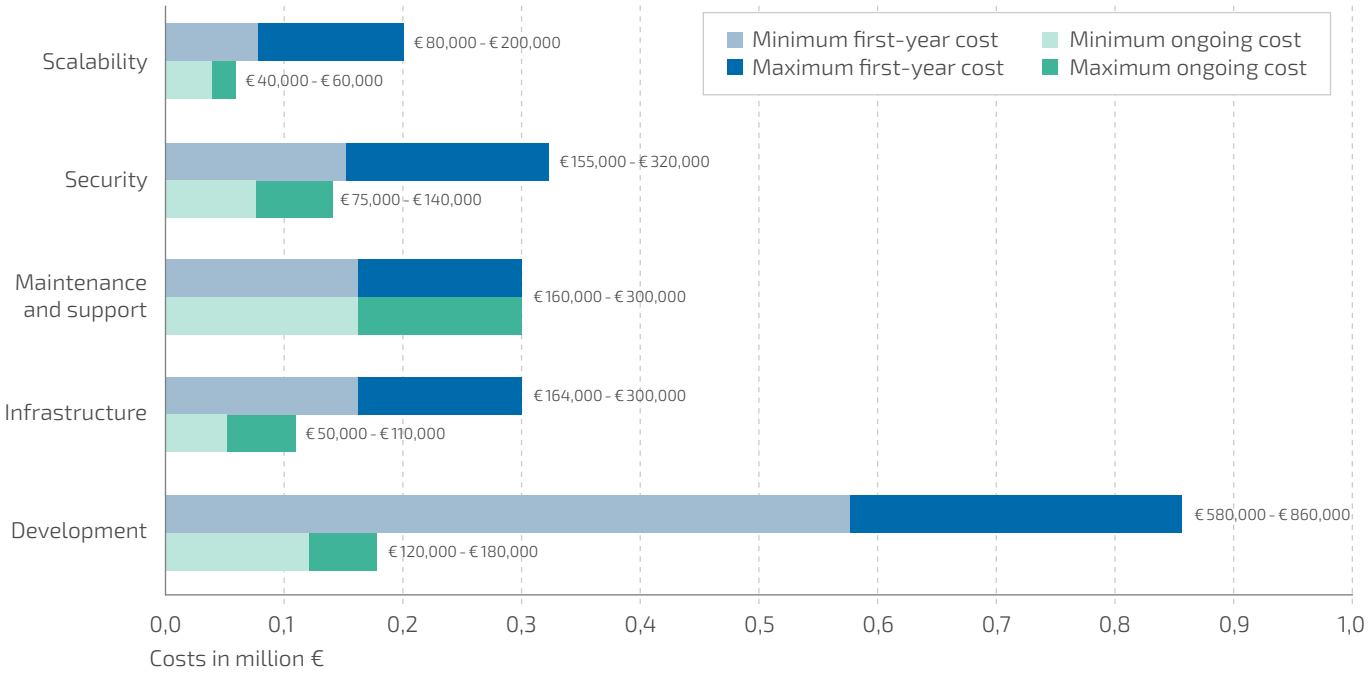
- € 80,000 - € 200,000 ((first year)
- € 40,000 - € 60,000 (ongoing scalability costs)

## 2.6 Total cost estimation (first year) and time estimate

### Total cost estimation

The following overview summarizes the estimated costs for the first year of developing an in-house IoT device management solution:

| Cost component | First-year costs | Ongoing-costs |
|---|---|---|
| Development costs | € 580,000 - € 860,000 | € 120,000 - € 180,000 |
| Infrastructure costs | € 164,000 - € 300,000 | € 50,000 - € 110,000 |
| Maintenance and support | € 160,000 - € 300,000 | € 160,000 - € 300,000 |
| Security | € 155,000 - € 320,000 | € 75,000 - € 140,000 |
| Scalability | € 80,000 - € 200,000 | € 40,000 - € 60,000 |
| **Total estimation** | **€ 1.15 Mio - € 1.98 Mio** | **€ 0.45 Mio - € 0.79 Mio** |




### Time estimation

The total time for full implementation is around 12 to 18 months. The times for development, security implementation and scalability planning have already been taken into account for this estimate.

## 2.7 Conclusion

Developing an in-house IoT device management solution for edge devices requires significant investment. First-year costs are estimated to range between €1.15 million and €1.98 million. However, this investment allows for a tailor-made solution that precisely meets your company's requirements. At the same time, you retain full control over future developments, security standards, and the scalability of the solution.

With an estimated time to market of 12 to 18 months, in-house development can provide long-term operational benefits, such as increased efficiency through optimized fleet management, cost savings via predictive maintenance features, and enhanced data security. Therefore, the decision to build an in-house solution should primarily depend on your company's strategic goals and available resources.

# 3 Checklist: key steps and resources for secure and scalable device management

This checklist provides an overview of the critical tasks involved in developing an in-house IoT device management solution. It outlines the necessary steps to ensure a future-proof and reliable device fleet management system. Additionally, this checklist helps you assess whether an in-house development is the right fit for your needs – or if an off-the-shelf solution might be the more efficient choice.

| | |
|---|---|
| **Secure and redundant operating system**<br><br>Goal:<br>Ensure device operation security and resilience | ☐ Harden the operating system by configuring its components to minimize attack surfaces, such as disabling unnecessary services and functions<br>☐ Implement regular updates and patches to fix known security vulnerabilities and keep the OS up to date<br>☐ Enable rollback functions to revert to previous stable versions<br>☐ Set up multi-factor authentication (MFA) for OS-level access |
| **Data privacy and security**<br><br>Goal:<br>Protect sensitive data and ensure regulatory compliance | ☐ Encrypt data at rest and in transit<br>☐ Implement data anonymization where possible<br>☐ Use secure communication protocols like SSL/TLS for encrypted connections<br>☐ Ensure data integrity with checksums and digital signatures to prevent unauthorized modifications |

### Scalable and high-performance infrastructure

**Goal:**
Handle growth and increased loads without performance degradation

- Implement load balancing and auto-scaling functions
- Use distributed systems to eliminate single points of failure
- Optimize infrastructure spending with cost-management strategies

### Vulnerability management with CVE (Common Vulnerability Enumeration) / CWE- (Common Weakness Enumeration) databases

**Goal:**
Identify and mitigate security risks proactively

- Conduct regular vulnerability scans
- Implement continuous monitoring for new vulnerabilities
- Set up alert mechanisms for newly discovered threats
- Automate the patching process for identified security issues
- Use tools to compare devices with up-to-date CVE/CWE databases

### Disaster recovery and business continuity

**Goal:**
Prepare for potential failures and ensure operational continuity

- Perform regular backups of critical data
- Regularly test recovery procedures to ensure reliability

### Version control

**Goal:**
Maintain control over software versions and fix vulnerabilities efficiently

- Keep a clear record of all software versions
- Ensure backward compatibility

### Regulatory compliance and legal requirements

**Goal:**
Minimize legal risks by adhering to industry standards

- Comply with regulations such as GDPR and ISO / IEC 27001
- Conduct regular audits to verify compliance
- Ensure proper documentation and reporting mechanisms

### Onboarding process

Goal:
Secure and efficient integration of new devices

- Use secure credentials and authentication for device registration
- Automate device provisioning and configuration
- Implement certificate management

### Device management

Goal:
Comprehensive control, configuration, and maintenance of the device fleet

- Automate firmware updates and deployments
- Enable scalable management of large device fleets
- Utilize integrated analytics to monitor device health

### Device health monitoring

Goal:
Detect and resolve issues early

- Set up alerts for unusual activity
- Use predictive analytics for issue forecasting
- Integrate self-repair mechanisms for automated trouble-shooting

### Seamless connectivity

Goal:
Ensure reliable and secure communication between devices and management systems

- Implement a fallback communication channel
- Optimize connectivity for low-bandwidth environments
- Use end-to-end encryption for data transmission

### Remote support

Goal:
Enable efficient remote maintenance and troubleshooting

- Ensure secure remote access with audit logs
- Allow remote diagnostics and repairs
- Implement secure remote wipe in case of theft or compromise

### Intuitive user interface

Goal:
Enhance usability for device management

- Provide customizable dashboards and reports
- Implement role-based access control to ensure that only authorized users can access sensitive functions

| | |
|---|---|
| **Edge computing capabilities**<br><br>Goal:<br>Improve performance through local processing | ☐ Deploy edge-level security to protect local data |
| **SBOM (Software bill of materials)**<br><br>Goal:<br>Manage vulnerabilities and ensure software transparency | ☐ Regularly review SBOM for outdated or vulnerable components<br>☐ Share SBOM with stakeholders to maintain transparency<br>☐ Use automated tools to generate and verify SBOM |
| **Docker Registry security**<br><br>Goal:<br>Ensure secure management of Docker images | ☐ Regularly scan images for vulnerabilities<br>☐ Implementation of access controls<br>☐ Implement access controls and role-based permissions<br>☐ Use image signing to verify integrity |

# 4 Technical requirements and solutions

## 4.1 How IoT devices connect

Many cloud applications rely on processing data from the physical world – from sensors, machines, or other devices. However, this data cannot always be directly processed in the cloud due to missing protocols or the lack of a secure connection. This is where IoT devices, often referred to as edge devices, come into play. Their name reflects their function: they operate at the edge of a central cloud system, bridging the gap between the real world and digital infrastructure.

The primary role of these devices is to convert raw data from existing formats into a structure that the cloud can process. Since this typically requires minimal computing power, IoT devices are often small, cost-effective computing units. These devices are usually seen as a technical necessity rather than a direct source of value for end users. As a result, companies focus on minimizing both hardware and software costs, from operating systems to applications.

Despite their compact design and limited functionality, IoT devices are particularly vulnerable to security risks. Their placement at the edge of cloud systems makes them a potential weak point, capable of compromising the entire data infrastructure. Many of these devices are installed in hard-to-reach or unattended locations, such as directly on machines or in remote industrial facilities. This exposes them to both physical and digital attacks.

Security requirements for IoT devices are often underestimated, leading to potentially severe consequences. A compromised device can not only put sensitive data at risk but also jeopardize the integrity of the entire system. Therefore, it is crucial to carefully plan and implement security measures specifically tailored for this class of devices.

## Excursus: system maintenance

The deployment of an operating system initiates a continuous process. New security vulnerabilities are constantly being discovered, putting the devices running this software at risk. As software complexity continues to rise, so does the number of vulnerabilities. In 2023 alone, over 26,000 vulnerabilities[1] were registered – 1,500 more than the previous year. This highlights the importance of ongoing security analysis and the rapid mitigation of discovered vulnerabilities.

Furthermore, exploits (code used to exploit vulnerabilities) are already available for approximately 80 % of all known vulnerabilities[2], making a swift response critical. Vulnerabilities are a primary target for attackers – about 3 out of 4 cyberattacks focus on vulnerabilities[3] that are at least two years old. While eliminating risk entirely is impossible, prompt patching and updates after a vulnerability disclosure can significantly reduce the risk of exploitation.

To ensure long-term security, organizations should continuously monitor vulnerabilities through trusted sources such as CVE (Common Vulnerabilities and Exposures) or CWE (Common Weakness Enumeration). When a critical vulnerability is identified, it is essential to have a well-defined, established process for quickly rolling out patches across the entire fleet.

### Software Bill Of Materials (SBOM)

When building an operating system, all external components – such as libraries, drivers, or other software modules – should be documented and compiled into a comprehensive list. This process can follow established standards, such as CycloneDX, and be provided as a machine-readable file.

The SBOM file should be distributed alongside the software, enabling users to conduct their own security monitoring. Additionally, importing the SBOM into database-driven security tools can automate the tracking of vulnerabilities.

### CVE- and CWE-Scans

By maintaining a database of all software components, organizations can automate vulnerability monitoring using up-to-date CVE/CWE data sources, such as the National Vulnerability Database (NVD). For each software component, relevant security information is retrieved from these sources, allowing companies to quickly identify and apply patches or updates to minimize risk proactively.

[1] https://www.comparitech.com/blog/information-security/cybersecurity-vulnerability-statistics und https://blog.qualys.com/vulnerabilities-threat-research/2023/12/19/2023-threat-landscape-year-in-review-part-one, 10.12.2024
[2] https://unit42.paloaltonetworks.com/state-of-exploit-development, 10.12.2024
[3] https://www.checkpoint.com/downloads/resources/cyber-security-report-2021.pdf, 10.12.2024

## 4.2 How to deploy an application to the device

**Installation process:**
Installing an application on an IoT device requires several carefully planned steps to ensure an efficient and error-free deployment. The process starts with preparing the runtime environment, often using Docker, followed by deploying the application. Automated tools and scripts play a key role in streamlining this process.

1. **Prepare the Runtime environment:**
   A Docker Engine (Runtime) must be installed on the IoT device. This runtime ensures that the application runs in an isolated and consistent environment, independent of the underlying hardware.

2. **Deploy the application:**
   The application is stored as a Docker image in a Docker registry. The registry serves as a centralized storage location from which devices can retrieve images.

3. **Automation:**
   The Docker Command Line Interface (CLI) helps automate build and release processes to enhance efficiency and accuracy. Automation reduces human error and enables scalable deployment across large fleets of devices.

## 4.3 How can Kontron hardware and software help?

Kontron's Yocto-Linux®-based IoT devices come pre-installed with the necessary Docker runtime. This makes them the optimal foundation for containerized applications.

**Update mechanism:**
Regular updates are essential to keep applications on IoT devices secure and functional. Over-the-Air Updates (OTA) allow flexible updates without physical access. Especially for large device fleets, a phased rollout is recommended: updates are first tested on small groups before being gradually deployed to all devices. This helps minimize errors and reduce failure risks.

The IoT device management solution KontronGrid significantly simplifies this process and ensures reliable execution. Using the template function, updates can be efficiently prepared by bundling Docker containers, Compose specifications, and the appropriate operating system into a central template. These templates can be applied with just a few clicks to individual devices or entire device groups, automating and accelerating the update process.

This also enables progressive update mechanisms. Subgroups of devices, such as those categorized by region, customer, or device type, can be targeted for updates first, before rolling out changes to the entire fleet. This ensures that updates function correctly and that the device fleet stays aligned with the latest technological advancements.

### Flexibility and expansion

The system architecture should be designed in a way that allows applications to be easily expanded or modified to accommodate new requirements. This demands a flexible system structure and the right tools for managing applications.

The described update mechanism is an example of this flexibility, covering not only the operating system but also Docker containers and Docker Compose configurations. KontronGrid supports this flexibility through an architecture that enables OS and application updates, as well as the management of Docker containers and Compose configurations. This allows even complex applications to be orchestrated on a device, ensuring seamless coordination between multiple Docker containers.

### Automation with Docker and CLI

Automation is a key factor in professional development processes. The workflow, from committing a source code change to building the application and deploying the resulting files, must be secure and fully automated. This also includes creating and uploading Docker images.

The native functionality of the Docker Command Line Interface (CLI) plays a central role in this process. It enables IT teams and developers to automate manual processes, such as managing permissions and uploading new images, efficiently. By automating repetitive deployment tasks, time is saved, and human errors are minimized.

### Reducing complexity with Docker Compose

In reality, IoT applications often do not consist of individual containers but rather a series of containers that need to work together seamlessly. These interactions between containers increase the complexity of the application and require an intelligent management system. Docker Compose provides a smart solution for this challenge. It was developed for managing multi-container applications and is seamlessly integrated into the KontronGrid platform.

For software developers and architects, Docker Compose is the tool to reduce the complexity in the management of containers to a minimum.

Docker Compose is based on a service-oriented architecture (SOA), which allows applications to be divided into smaller, manageable services that interact with each other. This not only improves the maintainability and scalability of applications but also provides a clearly structured organization of the application architecture. The biggest advantage of Docker Compose is that it simplifies the entire process of container orchestration.

Instead of managing individual containers, Docker Compose treats applications as a unified system. The entire configuration is centralized in a single file, enabling clear and efficient management. This significantly reduces deployment effort and avoids complicated rollout strategies.

For software developers and architects, Docker Compose is the tool to minimize complexity in container management.

### Simplified orchestration and configuration

Thanks to Docker Compose, service collaboration is optimized, and container management is designed to ensure seamless interaction within KontronGrid. Efficient management and clear structures are the focus, addressing the challenges of modern IoT applications.

**The following functions can be implemented:**

1. **Creation of Docker Compose projects**, including configuration in YAML format. This allows for fast and standardized deployment of applications consisting of multiple containers.
2. **Monitoring of Compose projects** using the Compose Console, which provides detailed debugging and diagnostic information through log entries. This ensures transparent control over the state of the containers.
3. **Editing of existing Compose projects**, including optimization of application configurations to ensure maximum efficiency and adaptability.
4. **Individual or collective control of Compose projects**, such as restarting or stopping individual or all containers. This simplifies management, especially for globally distributed IoT devices.
5. **Automated restart of containers in case of unexpected behavior or issues.** This ensures that operational failures are immediately corrected, and applications continue running smoothly.
6. **Deletion of Compose containers without losing configuration data.** Projects can be restored as needed, providing a flexible and scalable solution for changing requirements.
7. **Final deletion of Compose projects from an edge device** to free up storage space or remove outdated applications.
8. **Using Docker Compose as a template for fleet management.** This simplifies the process of managing multiple devices and applications, as the centralized configuration file can be duplicated and adapted for other devices or projects at any time.

## Excursus: Docker Container applications

### Docker on Linux®

Docker is tightly integrated into the Linux® environment, leveraging various Linux® kernel features such as cgroups and namespaces to provide isolated and efficient containers. This makes Docker an ideal platform for modular and scalable applications.

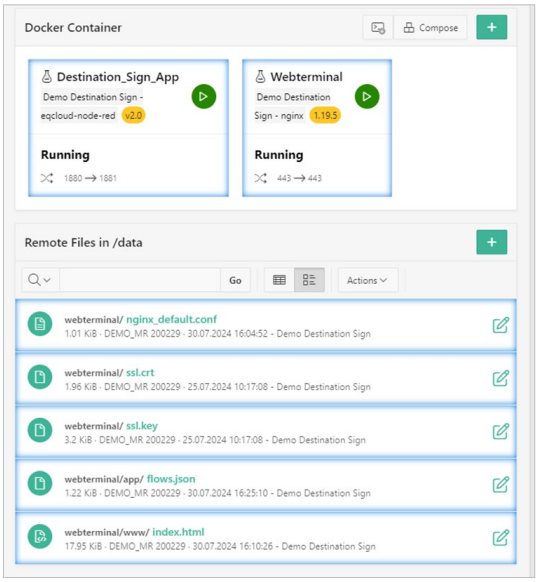
**Example: central control system for destination displays**

A practical example of Docker deployment is the simulation of a central control system that manages a bus destination display. This solution consists of multiple Docker containers:

1. NodeRed-Container (Destination_Sign_App):
   - This container simulates bus operations by updating the line number and destination of a virtual bus via a mini web service every minute.
   - The project is loaded remotely into the container, demonstrating how a central service dynamically controls destination information.
   - Although the simulation does not involve a real bus or physical display, it illustrates how a centralized control system could manage bus destination displays.

2. Web Terminal Container:
   - This container acts as a virtual display board since no physical sign is available. The bus destination and line number are visualized in a web-based interface.
   - Access via URL: https://172.16.102.191 and https://172.16.102.141
   - The web terminal retrieves real-time data from the NodeRed container, demonstrating how containers communicate with each other.

3. Web Server Container (Nginx):
   - A Docker container that deploys a web server with Nginx and SSL configuration.
   - This setup illustrates how to create and manage a secure web server within a Docker environment.




This application was deployed as a template on devices DEVICE 001 and DEVICE 002, showcasing how easily Docker environments can be standardized and scaled.

### Additional Docker application examples

Docker enables the creation of versatile applications that can be flexibly operated within containers. Examples include:

**NodeRed Container:**
- Manages machine connections and stores data in a database

**Database Container:**
- Stores machine data and allows for quick querying and analysis

**UserUI Container:**
- Provides a user interface for analyzing machine data

**Web erver or dashboards:**
- Creates interfaces and management platforms for various applications

**FabEagle®Connect Container:**
- Low-code interface framework for implementing complex interfaces in factory environments

## 4.4 How can IoT devices be securely maintained remotely?

Once IoT devices are in operation, it is essential that they function without interruptions. The integrated health monitoring provides insights into important parameters (memory and CPU usage, connection status, uptime temperature, etc.) of the IoT devices. If issues arise, service and support teams must respond quickly to prevent connection disruptions or data loss.

With KontronGrid, remote maintenance of IoT devices can be performed browser-based via either Remote Shell Desktop (RDP) or Secure Shell Services (SSH). The former allows access to the device's console, while the latter enables a graphical remote desktop connection. If the root cause of the issue is at the machine level, a temporary virtual network connection (VPN tunnel) can be established to securely access machines and systems connected to the IoT gateway, server, or PC within the customer's network.

VPN tunneling not only allows the operation of programming software or parameterization of the respective control system (PLC) but also enables location-independent collaboration of multiple service technicians via remote maintenance on a machine or device. The connections are automatically logged.

# 5 Cybersecurity requirements for digital products and solutions

The software development for IoT devices must not only be technically flawless but also comply with a wide range of legal requirements. These include the General Data Protection Regulation (GDPR), NIS-2, and the ISO 27001 standards. These regulations establish clear guidelines for data protection, data security, and the processing of personal data. Their goal is to ensure the integrity of systems and the confidentiality of data throughout the entire lifecycle of the devices.

**Application security:** Implementation of Security by Design and Security by Default to minimize vulnerabilities already in the development phase.

**Network security:** Measures to protect IT infrastructure from unauthorized access.

**Operational security:** Promoting secure behaviors in IT and software usage in daily business operations.

**End-user awareness:** Training and measures to proactively minimize risks caused by human error.

**Disaster recovery and business continuity:** Structures for rapid response and recovery after security incidents.

As a hardware and software supplier, Kontron plays a central role in the supply chain and supports companies in affected industries in meeting the requirements of the Cyber Resilience Act (CRA). We ensure that the Kontron components integrated into complete solutions, both hardware and software, are secured throughout their entire lifecycle, thereby facilitating compliance with cybersecurity requirements. We rely on two fundamental security principles: Security by Design and Security by Default. These two principles are explained in more detail in the following chapters.

## 5.1 Security by Design

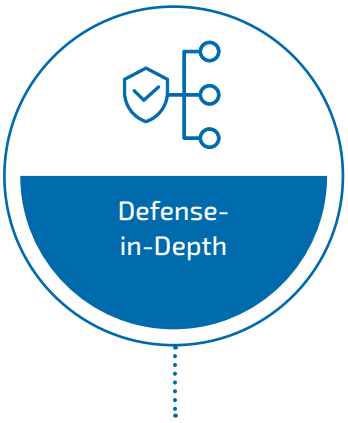
Security should not be treated as an afterthought – it must be at the core of every product development process, from the initial concept to final implementation. The Security by Design approach ensures that security aspects are systematically integrated from the very beginning of the development process. This is particularly crucial for IoT solutions, as both hardware and software are especially vulnerable to cyberattacks.

Security must be at the core of every product development process, from the initial concept to final implementation.

With the introduction of the **Network and Information Systems Directive (NIS-2)** and the upcoming **Cyber Resilience Act (CRA)**, this approach is becoming even more significant. Both regulations explicitly require that products and services be designed in such a way that security risks are minimized from the planning phase and that high resilience against cyber threats is achieved.

**In the context of NIS-2 and CRA, Security by Design includes the following measures in particular:**


Potential security risks are identified as early as the planning phase and mitigated through targeted design decisions




Even if one layer is compromised, the entire system remains protected through a multi-layered security strategy, which includes:

- Network segmentation through secure zones
- Multi-factor authentication (MFA) and granular access control
- Regular patches and updates to address newly discovered vulnerabilities


Regular audits and penetration testing are conducted to continuously evaluate and improve system security.

## 5.2 Security by Default

A central requirement of the NIS-2 directive and the CRA is the so-called security by default. This principle requires security measures to be integrated into systems, products and processes by default and activated automatically. This ensures that the highest level of security is already guaranteed ex works, without the need for additional configurations. Comprehensive technical, organizational and regulatory measures are required to meet these requirements. The most important steps are summarized below.

### 1. Technical measures

#### 1.1 Secure default configuration

- Systems and products must be delivered with secure default settings and pre-enabled security functions, including:
  - Disabling unnecessary services
  - Restrictive access controls
  - Minimizing attack surfaces
  - Firewalls
  - Encryption
  - Access restrictions

#### 1.2 Automatic security updates

- Automated update mechanisms to ensure systems remain up to date
- Regular vulnerability scans and timely implementation of security patches

#### 1.3 Protection against vulnerabilities

- Proactive detection of security gaps using vulnerability management tools
- Regular security assessments, including penetration testing

#### 1.4 Encryption and authentication

- Implementation of modern encryption standards for stored and transmitted data
- Multi-factor authentication (MFA) as a standard security measure
- Principle of least privilege for user accounts to restrict unnecessary access

### 1.5 Network security

- Network segmentation to contain potential security incidents
- Use of Intrusion Detection and Prevention Systems (IDS/IPS)
- Encryption of data transmission using secure protocols like TLS

## 2. Procedural measures

### 2.1 Risk analysis and security assessments

- Continuous risk analyses and security evaluations form the foundation for targeted security strategies
- Documentation of risks and corresponding countermeasures

### 2.2 Incident response and monitoring

- Establishment of an Incident Response Plan for cybersecurity incidents
- Real-time monitoring of critical systems and networks for rapid detection of security threats

### 2.3 Training and awareness

- Regular training sessions for employees on current threats and security best practices
- Building a security-conscious corporate culture through awareness programs

### 2.4 Supply chain security

- Integration of security requirements into contract terms
- Full transparency over all suppliers and their security status
- Security audits and regular assessments of supplier and third-party security measures
- Actionable recommendations to improve security posture when vulnerabilities are identified

### 3. Regulatory measures

#### 3.1 Proof of compliance

- Documentation of Security by Default compliance through:
  - Regular audits
  - Creation of cybersecurity reports

#### 3.2 Cooperation with authorities

- Adherence to reporting obligations for security incidents
- Close collaboration with authorities to meet regulatory requirements

## 5.3 Differences, role distribution, and distinctions

**Key differences at a glance:**

| Security by Design | | Security by Default |
|---|---|---|
| During the development and design phase | Time of implementation | During deployment (out-of-the-box) |
| Proactive integration of security into design | Focus | Pre-configured for maximum security |
| Developers and designers | Target audience | End users |
| Prevent security issues | Approach | Secure usage without additional configuration |
| Threat modeling, secure architecture | Examples | Disabling unused features |

The security of digital products and solutions is a shared responsibility. The IEC 62443 standard, established by the International Electrotechnical Commission (IEC), is an internationally recognized framework for securing industrial automation and control systems. It provides a comprehensive approach to cybersecurity, defining requirements and processes for all involved parties, including operators, product suppliers, and service providers.

IEC 62443 distinguishes between different stakeholders, requiring each to comply with specific parts of the standard. As a leading manufacturer of hardware and software solutions, Kontron assumes responsibility as a Product Supplier, ensuring that security requirements are met throughout the entire product lifecycle.

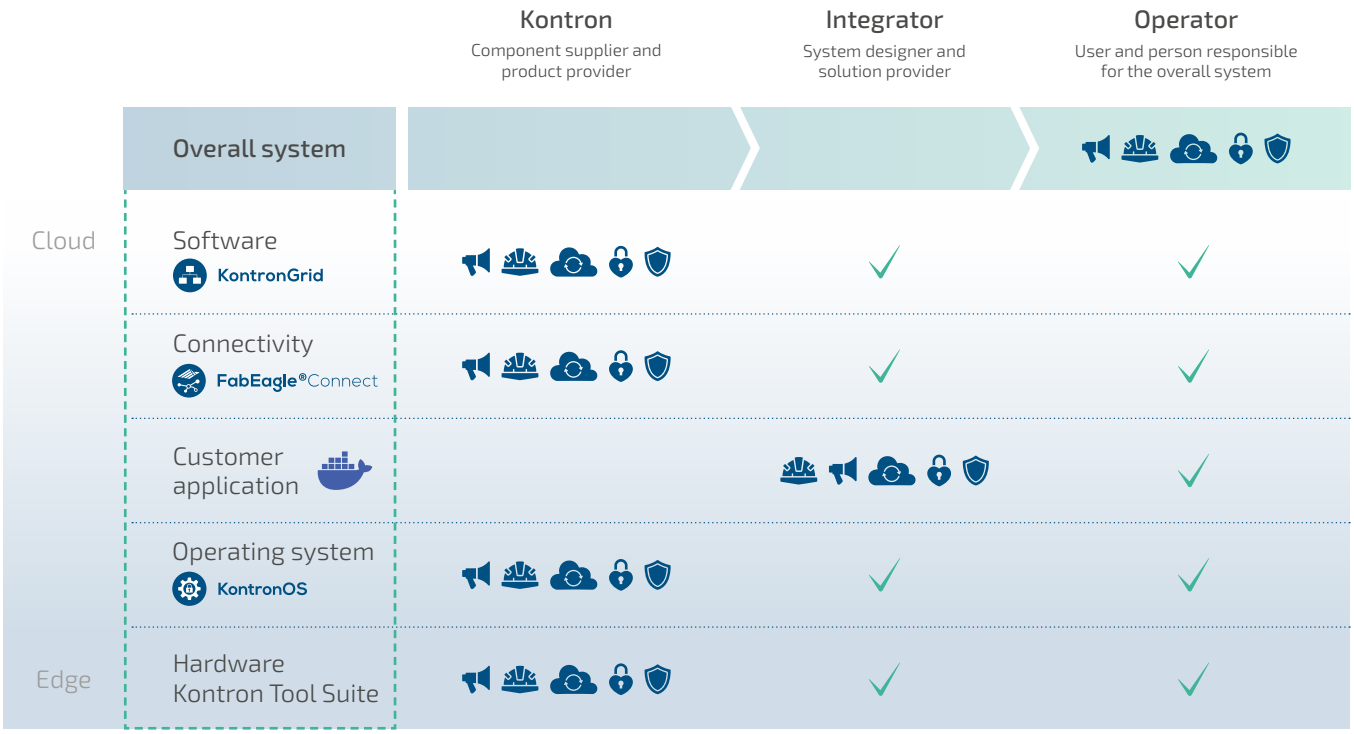
Additionally, our products support IoT solution providers who need to protect their supply chain and integrate secure components. Kontron's solutions help integrators, manufacturers, and operators alike respond effectively to security incidents and strengthen their system resilience.

A key player in this ecosystem is the Integrator, who plays a crucial role in delivering a secure and efficient IoT solution by handling various responsibilities:

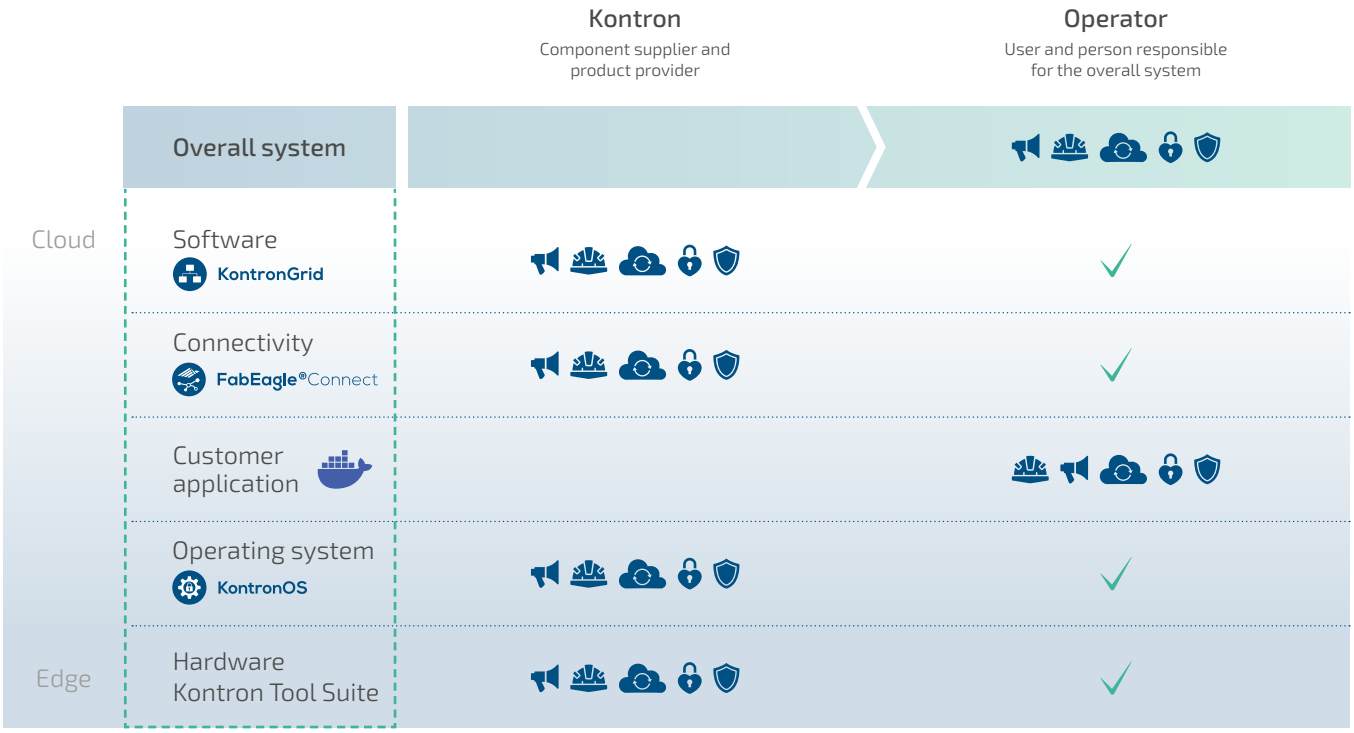
- **System integration:**
  Combining various hardware and software components into a functional IoT solution. Ensuring seamless communication between devices, platforms, and networks by integrating protocols and APIs.
- **Customizing and adaption:**
  Tailoring IoT solutions to the specific needs of operators or end customers. Developing custom applications, interfaces, dashboards, and analytics tools.
- **Project management:**
  Coordinating development projects, budgeting, and aligning all involved parties.
- **Technical expertise and consulting:**
  Advising operators and end customers on architecture, security, and scalability of the IoT solution. Ensuring that all components meet relevant security requirements.

In the collaborative ecosystem between Kontron as a hardware and software provider, integrators developing customized applications, and operators deploying and securing the entire system, each stakeholder plays a critical role.

The following diagrams illustrate two possible collaborative models involving two or three key stakeholders in the IoT security landscape.




Cyber security tasks and responsibilities using the example of an overall system with three stakeholders




Cyber security tasks and responsibilities using the example of an overall system with two stakeholders

End customer education · Disaster and business continuity · Operational securty · Program security · Network security

## 5.4 Security aspects of KontronOS and KontronGrid

As a hardware and software manufacturer, we take responsibility for securing the entire product lifecycle of our solutions. This means ensuring the highest security standards at every stage, from development and implementation to maintenance and operation. Our solutions, KontronGrid and KontronOS, provide comprehensive security features that align with international standards such as ISO 27001 and IEC 62443. This ensures that our products are not only high-performing but also resilient against modern cyber threats.

### ISO 27001

The ISO 27001 certification guarantees a comprehensive and systematic security management approach, covering all aspects of information security. Key components include:

- **Information security policies (A.5):** Development and enforcement of clear security guidelines
- **Organization of information security (A.6):** Establishment of clear responsibilities and structures
- **Personnel security management (A.7):** Employee screening and security training
- **Asset management (A.8):** Protection and management of all physical and digital assets
- **Access Control (A.9):** Role-based access rights and prevention of unauthorized access
- **Cryptography (A.10):** Implementation of modern encryption standards
- **Physical Security (A.11):** Protection of facilities and equipment against physical threats
- **Operational Security (A.12):** Ensuring stable and secure operational processes
- **Communication Security (A.13):** Protection of data transmissions
- **Supplier Relationships (A.15):** Integration of security requirements into the supply chain
- **Security Incident Management (A.16):** Proactive identification and resolution of security incidents
- **Disaster Recovery (A.17):** Planning and response strategies for unexpected events
- **Compliance (A.18):** Adherence to legal and regulatory requirements

In addition to technical security standards, we fully comply with the requirements of the General Data Protection Regulation (GDPR) to ensure the confidentiality, integrity, and availability of personal data:

### Confidentiality:

- **Access control:**
  Restricted access to sensitive facilities through tiered permissions, alarm systems, and monitoring
- **System access control:**
  Protection against unauthorized system use through password policies and firewalls
- **Data access control:**
  Granular access rights, audit trails, and secure deployments
- **Data separation control:**
  Strict separation of data based on processing purposes and use of isolated databases
- **Pseudonymization:**
  Protection of personal data through anonymization techniques

### Integrity:

- **Data transfer control:**
  Traceability of data transmission
- **Input control:**
  Logging of changes and access to personal data

### Availability and resilience:

- **Availability control:**
  Protection against data loss through backups and redundancy
- **Recoverability:**
  Regular testing of data restoration procedures

### Regular review and evaluation:

- Data Protection Management
- Incident Response Management
- Privacy-friendly default settings
- Contractual control of third-party data processing

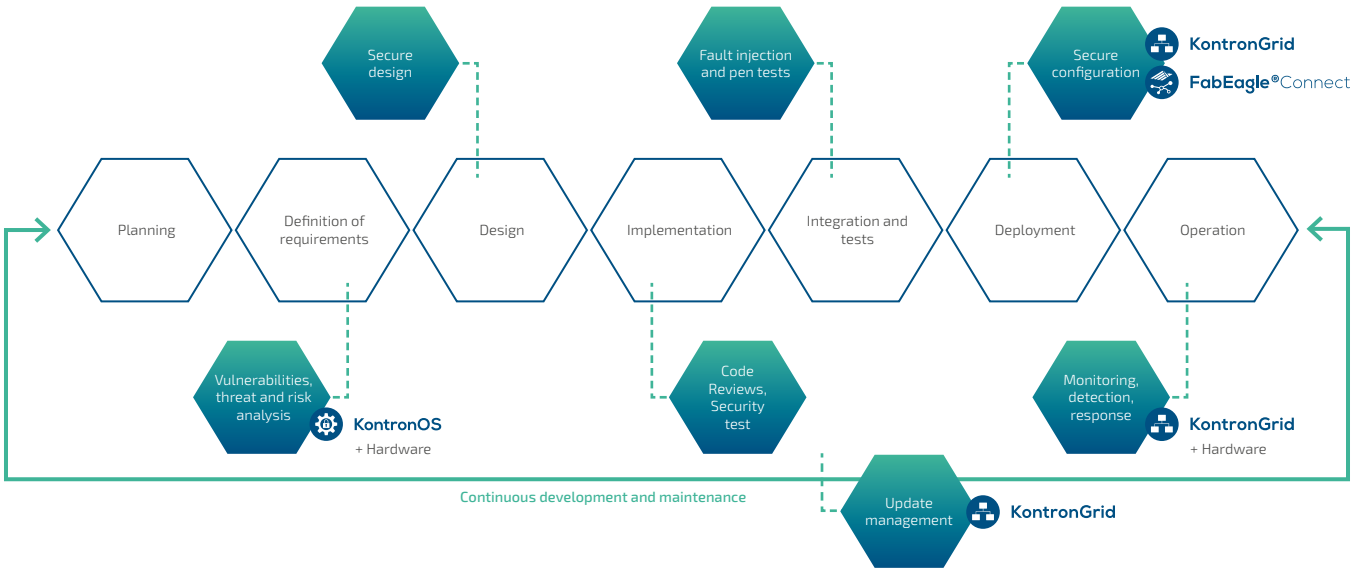
## IEC 62443

Security is a critical concern for industrial automation systems. The IEC 62443 standard provides a clear framework to systematically minimize cybersecurity risks. Kontron is taking a significant step forward: Our development process is currently undergoing certification according to IEC 62443-4-1, with certification expected to be completed in Q2 2025.

Why is IEC 62443 so important? The IEC 62443 standard is specifically designed to address the unique security requirements of industrial environments, complementing the well-established ISO 27001 guidelines. It covers key security aspects that are particularly critical in Operational Technology (OT). By adhering to these standards, we ensure that our products not only meet regulatory requirements but also effectively handle security challenges in industrial operations.

With our software solutions, including KontronGrid (IoT Device Management) and KontronOS (secure Linux-based operating system), we strengthen your IoT infrastructure at two critical points:

- Vulnerabilities are identified and addressed early
- Potential threats are effectively and securely mitigated




## Security features of our solutions in detail

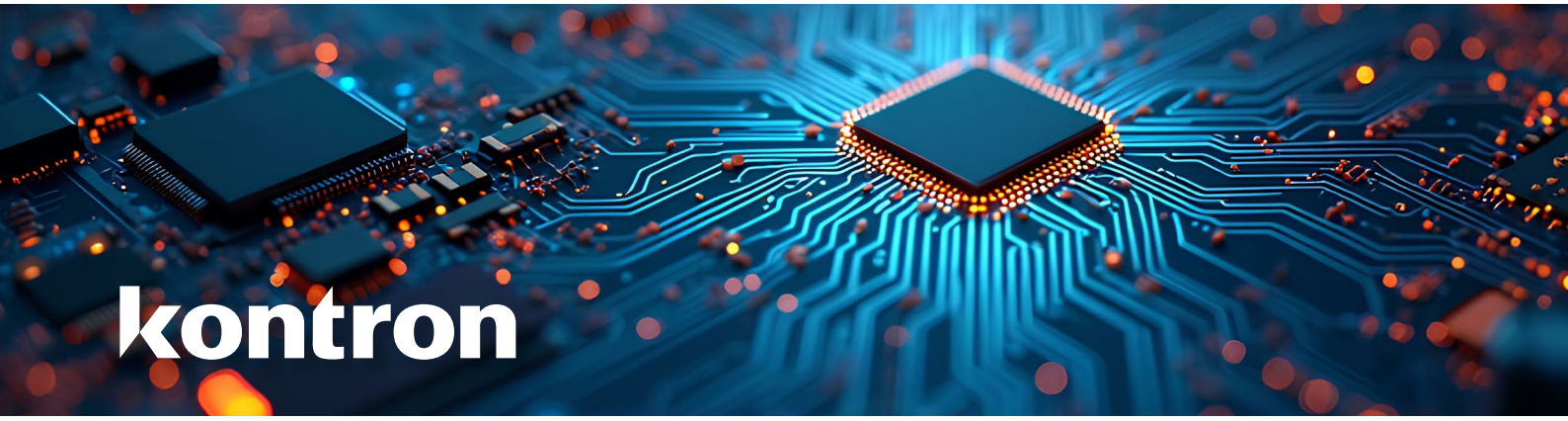
### Analysis of threats, risks, and vulnerabilities

Security starts with analysis. That's why we conduct regular vulnerability assessments to detect potential risks early. If critical vulnerabilities are identified, we provide patches as quickly as possible. At the same time, fallback mechanisms ensure that data integrity and availability are maintained even in crisis situations.

**Secure configuration**
Our products are delivered with secure default settings, ensuring protection against common attack patterns without requiring additional configuration.

**Monitoring, detection, and response**
Through continuous monitoring and targeted security surveillance, we can detect threats in real time and immediately initiate countermeasures.

**Update management**
Security updates are automated, ensuring that your systems are always up to date and that known vulnerabilities are addressed promptly.




## Hardware-level security

Security measures at the hardware level are just as crucial as those at the software level. Here are some key security features from our portfolio:

- **Certificate and encryption configuration:**
  Secure BIOS solutions, Secure Boot, and generation of secure keys provide protection directly at the hardware level.
- **Incident handling:**
  Our Product Security Incident Response Team (PSIRT) continuously monitors security incidents, conducts regular vulnerability checks, and cross-references findings with relevant security databases.
- **Systematic security updates:**
  Certified products receive periodic BIOS updates and patches to mitigate security vulnerabilities in the long term.
- **Secure development process:**
  The entire manufacturing process is tightly controlled, from product configuration to access control. Our processes are fully transparent and
- aligned with IEC 62443-4-1 requirements.


## Operational security with KontronOS

The hardened operating system KontronOS, based on Yocto-Linux®, is specifically designed to run customer applications on Intel® x86- or Arm®-based edge devices securely and reliably.

Pre-installed and optimized for connected systems, KontronOS ensures maximum security and up-to-date protection, even in critical environments. Additionally, it offers a range of advanced security features:

### SBOM (Software Bill of Materials):

A complete list of libraries and modules used provides transparency and enables the tracking of security vulnerabilities, including:

- A detailed inventory of all used libraries, modules, and their versions
- Systematic documentation and tracking of vulnerabilities identified in third-party components

By leveraging the SBOM, we can proactively detect known security issues, provide patches, or identify alternative solutions.

### Auto-detection and vulnerability scans:

This feature enables real-time detection of security vulnerabilities and patching needs. Our products are continuously scanned for known security issues by cross-referencing with databases such as CVE and CWE. This process includes:

- **Security issue descriptions:**
  Mapping identified vulnerabilities to affected components
- **Remediation guidelines:**
  Providing solutions such as patches or configuration adjustments to fix vulnerabilities

By automatically tracking third-party software, we ensure that we are always aware of the latest security risks and patches, keeping KontronOS secure and resilient.

### Network security:
The integrated firewall and network zones protect against unauthorized access and segment the network for enhanced security.

### Regular system updates and patching:
Security patches are continuously deployed to address known vulnerabilities.

### Redundancy and fallback mechanism:
This feature ensures system availability in case of failures or attacks.

### Minimalist operating system configuration:
The operating system is reduced to essential kernel configurations, with automatic updates disabled to minimize potential security risks. Explicit file permissions and device-specific logins further enhance security.

### Deactivation of unnecessary ports:
Shutting down unused services and ports and disabling root access reduces the attack surface on security systems.

### Principle of least privilege:
Only the necessary permissions are granted, minimizing the risk of insider threats or misconfigurations.

### Centralized logging:
User activity logging and documentation of deployed open-source licenses ensure traceability.

- **Encryption:**
  All connections (to the cloud, web interface, etc.) are secured with state-of-the-art encryption algorithms.
- **Integrity:**
  UEFI Secure Boot or HAB Secure Boot ensures the integrity of firmware and OS software.
- **Application isolation:**
  Strict separation of the operating system and applications using container virtualization (Docker) and/or Mandatory Access Control Systems (AppArmor) enhances operational security.

# KontronGrid

## Secure IoT Device Management with KontronGrid

KontronGrid is our IoT device management solution for edge devices. It simplifies the deployment of container applications, automates updates, monitors devices, and provides fast remote support, making it an ideal choice for efficiently managing globally distributed device fleets.

Security is a top priority at the IoT device management level as well.
With KontronGrid, we offer:

### Update management
Supports continuous development and maintenance cycles through automated updates.

### User and access management
Multi-factor authentication (MFA), Role-based access control for each service and Granular user group management ensures secure access control.

### End-to-end encryption
All data is transmitted in encrypted form using SSL / TLS 1.2 to ensure secure communication between the services. Customer data is encrypted in accordance with AES 128.

### Device and service authentication
Each individual device is strictly separated and each service is authenticated, which prevents unauthorized access. Establishing a remoting connection requires two-step authentication. The SSH tunnel is RSA 2048 bit encrypted. Unused VPN connections are automatically terminated and tunneling into the customer network is optional.

### Logging
All security-relevant events are fully logged to enable a rapid response in the event of incidents.

### Test and validation

Security-relevant functions of our products are systematically and, where possible, automatically tested. Here are some examples:

- **HTTPS security of KontronGrid:**
  Before every rollout, a comprehensive test is performed with the goal of achieving an A+ rating.
- **REST-API tests:**
  Using tools like Apache JMeter and Azure DevOps, we test scenarios for different user types (Admin, Invalid User, Regular User) to ensure that all REST calls are secure and function correctly.
- **Risk-based testing:**
  In addition, we conduct risk-based tests, where risk groups and levels are defined based on parameters such as significance, occurrence frequency, reproducibility, and acceptance.

To detect potential security threats early and respond effectively, we rely on structured threat modeling based on the S.T.R.I.D.E Framework. This framework helps systematically analyze security risks and develop appropriate countermeasures. The six key threat categories are:

- **Spoofing:**
  Impersonation of users or services
- **Tampering:**
  Manipulation of data or systems
- **Repudiation:**
  Denial of actions performed, without proof
- **Information disclosure:**
  Unauthorized disclosure of sensitive information
- **Denial of service:**
  Prevention of access to services through overload
- **Escalation of privilege:**
  Increasing access rights by exploiting security vulnerabilities

This systematic threat modeling forms the basis for our security strategies and is integrated into the development and operation of our products.

# 6 Why make it complicated, when it can be simple?

Development teams of IoT solutions want to focus entirely on application development without being held back by manual or complex deployment processes. They expect a secure and automated deployment process that also works seamlessly for testing purposes. IT managers place particular value on tamper-proof IoT devices that are reliably protected from external threats. Even in the event of an issue, attackers should only be able to access a device with significant effort.

Product managers of IoT solutions strive to provide users with a first-class product experience. This means ensuring high reliability, maximum availability, and always up-to-date software applications – from the user interface to real-time data analysis directly at the edge. Chief Digital Officers (CDOs) also recognize a significant monetization potential that goes beyond the physical solution itself. Custom applications and the growing demand for personalization require flexible scalability and modularity that span the entire product lifecycle.

With KontronGrid, our IoT device management solution for edge devices, we offer an efficient, secure, and scalable solution for IoT device management and remote monitoring. KontronGrid reduces IT complexity, increases cost efficiency, and provides the highest security standards. It supports monetization through digital services, accelerates product development cycles, and optimizes control over globally distributed devices by simplifying the deployment of container applications, automating fleet updates, monitoring devices, and enabling fast remote support for efficient management of global device connections.

## Complete solution from a single source: integrated hardware and software – the answer is ManagedEdge IoT Bundle

Users benefit from a complete solution from a single source, ensuring seamless integration with perfectly coordinated hardware and software. In addition to the fleet management solution, the overall package is completed by application-specific hardware and a secure Linux®-based operating system tailored to it.

Everything is combined in an IoT bundle, offering numerous advantages: The devices are ready for immediate use thanks to pre-installed and pre-configured software, enabling a quick start into global device management.

At the same time, the IoT bundle enables cost-efficient prototyping and simplifies the development and rollout of new applications through optimized processes. Users also benefit from accelerated implementation and the ability to scale their solutions flexibly and efficiently. This makes the IoT bundle a future-proof foundation for industrial applications.

The portfolio of compatible Kontron hardware with tailored secure operating systems is continually growing and currently includes the following standard series:

- KBox A-Serie
- SOM AL-i-Serie

Custom hardware is one of Kontron's core competencies, providing component and device manufacturers with high flexibility and adaptability to their target applications. Specific requirements for hardware and software can be implemented seamlessly.

# 7 Checklist and decision support – build or buy?

The following checklist helps you quickly and easily assess the key criteria for your decision and provides a clear overview of which approach is best suited for your company: In-house development or ready-made solution. Go through each point and evaluate what is most important for your situation.

| | Build<br>(in-house development) | Buy<br>(third-party solution) |
|---|---|---|
| Time and market launch urgency<br>**Do you need a solution on the market quickly?** | ☐ **No:** If you have time, you can develop your own solution, which can be tailored to your needs in the long term. | ☐ **Yes:** A ready-made solution offers quick integration and immediate operational readiness. |
| Cost planning and budget<br>**Can you afford high initial investments or prefer predictable ongoing costs?** | ☐ **Investment readiness:** In-house development requires higher initial spending for development, infrastructure, and personnel, but may be more cost-effective in the long run. | ☐ **Prefer predictable ongoing costs:** The ready-made solution offers plannable licensing, or subscription fees and reduces high upfront costs. |
| Internal resources and expertise<br>**Do you have an internal team with the necessary expertise to develop and maintain an IoT platform?** | ☐ **Yes, we have the necessary skills:** In-house development offers flexibility but also greater responsibility for maintenance and updates. | ☐ **No, our resources are limited:** The ready-made solution reduces the burden on internal teams and takes care of maintenance, support, and security. |
| Flexibility and customization<br>**Do you need very specific customization and deep integration into existing systems?** | ☐ **Yes, a tailored solution is crucial:** In-house development allows full customization to your needs, but at the cost of time and resources. | ☐ **No, a flexible standard solution is enough:** A ready-made solution often offers sufficient customization options via APIs and configuration without the development effort. |

| | Build<br>(in-house development) | Buy<br>(third-party solution) |
|---|---|---|
| **Security and compliance**<br>**Is security a critical aspect for your IoT solution?** | ☐ **Yes, and we can ensure it internally:** If you have a strong internal security team, in-house development with customized security protocols could be beneficial. | ☐ **Yes, but we prefer proven security systems:** Ready-made solutions come with integrated security protocols, continuous updates, and CVE monitoring. |
| **Scalability and growth**<br>**Are you planning to grow rapidly and scale your IoT fleet globally in the near future?** | ☐ **No, our growth will be gradual:** In this case, in-house development with customized scalability might be advantageous in the long term. | ☐ **Yes, we expect fast growth:** A ready-made solution is designed for scalability from the start and supports global expansion without significant additional effort. |
| **Maintenance and long-term support**<br>**Are you ready to handle maintenance and patching of the solution in the long term?** | ☐ **Yes, we can handle maintenance internally:** In-house development requires a dedicated team for ongoing maintenance, bug fixes, and updates. | ☐ **No, we don't want to worry about maintenance:** The ready-made solution offers integrated support and regular updates, minimizing the effort. |
| **Technological dependency and independence**<br>**Do you want full control over the solution or are you willing to rely on a third-party vendor?** | ☐ **Full control is crucial:** In-house development offers complete independence but comes with the responsibility for all updates and further developments. | ☐ **We prefer a proven third-party solution:** With a ready-made solution, you can rely on the vendor's expertise but may need to make compromises in customization. |
| **Risk and innovation management**<br>**Are you willing to take on the risk of a new development to potentially gain long-term competitive advantages?** | ☐ **Yes, we are willing to take risks:** In-house development can be more innovative, but it carries the risk of delays or cost overruns. | ☐ **No, we want a proven, secure system:** A ready-made solution offers less risk, as it has already been tested and optimized. |

# 8 Conclusion: "build or buy" as a key question

If you prioritize speed, low initial investment, external expertise, and security from a single source, we recommend the **ManagedEdge IoT-Bundle**. It offers an immediately available, scalable, and secure IoT management platform that allows you to focus on your core applications. On the other hand, if full customization, long-term control, and the use of internal resources are essential, in-house development might be advantageous. However, you should consider the associated time, budget, and risk factors.

A ready-made solution, such as KontronGrid and KontronOS, allows companies to focus on core applications and business success while taking care of the management, security, and scalability of their IoT fleet. It eliminates the complexity of edge device management, offering a secure and scalable solution that is ready for immediate use. At the same time, such a complete offering relieves companies of technical and operational tasks, providing seamless integration, constant updates, and optimized management, without the risks and high costs of in-house development.

With a proven IoT device management solution, companies can pragmatically address increasing demands for innovation and security in the IoT environment. Not only can they ensure the reliability of their IoT devices, but they can also flexibly adapt to future requirements.

**Build or Buy?** Let us work together to determine which path is the optimal solution for your company. Click here for a personalized consultation.

## About Kontron AIS GmbH

Kontron AIS GmbH has been setting benchmarks in industrial software development for over 30 years. An experienced team of more than 250 employees stands for proven software products and customized digitalization solutions. This enables machine and equipment builders as well as factory operators to break new ground in automation and secure long-term competitive advantages. Together with its customers, Kontron AIS implements intelligent digitalization strategies worldwide, paving the way for the smart manufacturing of tomorrow.

As a subsidiary of Kontron AG, Kontron AIS provides comprehensive IoT solutions from a single source, covering both hardware and software. A global network ensures project management, service and support on a global scale.

Further information: **www.kontron-ais.com**

Company contact
Kontron AIS GmbH | Otto-Mohr-Str. 6 | 01237 Dresden | +49 (0) 351 2166 0 | sales@kontron-ais.com