HighIntegritySystems

WITTENSTEIN

RTOS Security: SAFERTOS® and the Cyber Resilience Act

Issue 1.1 - July 20, 2026

WITTENSTEIN high integrity systems
Americas: +1 408 625 4712
ROTW: +44 1275 395 600
Email: sales@high**integrity**systems.com
Web: www.high**integrity**systems.com

Contents

List of Figures

List of Notation

ACP	Access Control Policy	OACP	Object Access Control Policy
API	Application programming interface	RTOS	Real Time Operating System
CRA	Cyber Resilience Act	SVD	Software Defined Vehicle
ESM	Enhanced Security Module	SBOM	Software Bill of Materials
HAZOP	Hazard and Operability	TARA	Threat Analysis and Risk Assessment
MPU	Memory Protection Unit	TCB	Task Control Block
MMU	Memory Management Unit	WHIS	WITTENSTEIN high integrity systems

References

1. Regulation (EU) 2024/2847 of the European Parliament and of the Council of 23 October 2024 on horizontal cybersecurity requirements for products with digital elements (Cyber Resilience Act).
2. ISO/SAE 21434: Road Vehicles — Cybersecurity Engineering.
3. IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems.
4. ISO 26262: Road Vehicles — Functional Safety.
5. IEC 62304: Medical Device Software — Software Life Cycle Processes.
6. RTCA DO-178C: Software Considerations in Airborne Systems and Equipment Certification.
7. IEC 61882: Hazard and Operability Studies (HAZOP Studies) – Application Guide

CHAPTER 1 Introduction

The Cyber Resilience Act (CRA) is changing how manufacturers and software suppliers approach cybersecurity for digital products sold in the European Union. From September 2026, manufacturers will be required to meet vulnerability reporting obligations, with the main requirements taking effect in December 2027. While the regulation is often discussed in the context of connected devices, compliance also depends on the software components that sit beneath them. For suppliers of foundational software, this represents a significant shift in responsibility.

Real Time Operating Systems (RTOS) occupy a particularly important position in this discussion. They rarely attract the same attention as user-facing software, yet they provide the foundation for task scheduling, resource management and system isolation within embedded devices. In short, if a vulnerability exists at this level, the security and resilience of the entire product can be affected. This makes assurance, traceability, and resilience key considerations under the CRA.

SAFE**RTOS**® from WITTENSTEIN high integrity systems (WHIS) was developed for applications where reliability and predictability are non-negotiable, and where compliance with demanding safety standards is already a requirement. Building on this foundation, WHIS has incorporated security-focused development practices and runtime protection mechanisms that align closely with the principles underpinning the CRA.

This paper explores how the CRA applies to foundational software components, why operating systems are likely to attract a heightened level of scrutiny, and how SAFE**RTOS**® helps developers of safe and secure embedded systems address modern cybersecurity and compliance requirements.

CHAPTER 2 The Cyber Resilience Act

From a software component standpoint, the CRA establishes that commercially distributed software is not merely a development artefact, but a regulated product with defined cybersecurity obligations [1].

A software component—such as an operating system, middleware, or library—falls within scope when it is placed on the EU market for integration into other products. Here the component supplier assumes responsibility for the cybersecurity properties of the component itself, independent of the final system in which it is used. This represents a shift in accountability, requiring suppliers to demonstrate that their components are secure, well-governed, and maintainable throughout their lifecycle.

A CRA-focused review can be understood to centre on three key themes. First, the component must demonstrate that it has been developed with security in mind [1]. This includes the application of secure design principles, controlled functionality, and clearly defined operational boundaries, ensuring predictable behaviour and minimising unintended attack surfaces. Second, there must be a structured approach to vulnerability management, enabling the identification, assessment, remediation, and communication of vulnerabilities across potentially wide deployment bases. Third, the component must support transparency and traceability through clear versioning, documentation, and suitability for inclusion in a Software Bill of Materials (SBOM), allowing downstream users to assess integration risks.

These expectations elevate software components from passive dependencies to accountable elements of the supply chain, with obligations that persist throughout their operational life.

2.1 Scope and Exclusions

While the CRA has broad applicability, it does not apply uniformly to all software, and understanding its boundaries is essential when determining scope.

The most notable exclusion applies to software developed and distributed without commercial intent, such as certain open-source projects. However, this exclusion is limited: once open-source software is incorporated into a commercially distributed product, responsibility transfers to the entity placing that product on the EU market. Similarly, software developed solely for internal use, and not made available on the EU market, is generally outside the scope of the CRA.

The regulation also recognises existing sector-specific frameworks in industries such as automotive, aerospace, and medical systems. Where equivalent cybersecurity requirements are already enforced, the CRA avoids duplication by deferring to these established regimes.

A further distinction is made between standalone components and internally integrated software. Components supplied independently, such as a licensed RTOS or development SDK, are in scope as products in their own right. In contrast, components that are never distributed separately are assessed only within the context of the final product.

These boundaries clarify where formal obligations apply, while still reinforcing the broader expectation of secure and well-managed software development practices.

This white paper explores the type of security that can be provided by an RTOS, and uses SAFE**RTOS**® and its Enhanced Security Module (ESM) as an example of good practice.

2.2 CRA Risk and Assurance Expectations

The CRA introduces a relationship between the level of concern associated with a product and the level of assurance that must be demonstrated. Rather than applying uniform requirements, the regulation scales expectations based on the role, exposure, and potential impact of the software.

Broadly speaking, components that provide foundational functionality, such as operating systems or security services, attract greater scrutiny because failures or vulnerabilities can have widespread consequences. As the level of concern increases, so too do the expectations for rigour, evidence, and verification.

Practically, this creates a tiered model in which baseline components demonstrate compliance through standard development and internal assurance processes, while higher-risk or critical components require stronger verification, greater traceability, and potentially independent assessment. Widely reused components may also attract increased scrutiny due to their systemic impact, even in the absence of direct exposure.

Therefore, compliance under the CRA is not a binary judgement, but a graduated assessment in which the required level of assurance is directly aligned with the criticality and impact of the component.

2.3 Foundational Software

CRA discussions often focus on attack surface and connectivity when evaluating cybersecurity risk. For foundational software components such as an RTOS, impact is often a more significant consideration than external exposure. An RTOS may never communicate externally, yet it remains one of the highest-consequence elements within an embedded system.

CHAPTER 3 SAFE**RTOS**®

3.1 SAFE**RTOS**®

SAFE**RTOS**® is a commercial real-time operating system (RTOS) for safety-critical embedded systems requiring predictable behaviour and compliance with functional safety standards. Developed by WITTENSTEIN high integrity systems (WHIS), it provides deterministic task scheduling, memory management, and inter-task communication to support time-critical applications. While SAFE**RTOS**® follows the functional model of FreeRTOS™ and offers a similar API, it is an independently developed RTOS engineered using rigorous safety-critical processes to meet the assurance and certification requirements of high-integrity systems.

SAFE**RTOS**® is widely deployed in industries such as automotive, aerospace, medical devices, industrial automation, and railway systems, where software failures can have significant safety implications. It is supplied with comprehensive certification evidence to support compliance and/or certification to internationally recognised functional safety standards, including IEC 61508 SIL 3 [3], ISO 26262 ASIL D [4], IEC 62304 Class C [5], and DO-178C DAL A [6], helping product developers streamline regulatory approval activities.

3.2 Why SAFE**RTOS**® is effected by the CRA

SAFE**RTOS**®, as a standalone software component with no direct external interfaces, presents a distinct profile when assessed under the Cyber Resilience Act (CRA) in terms of both level of concern and assurance expectations. From an exposure perspective, SAFE**RTOS**® does not provide external communication interfaces and therefore does not directly introduce external entry points; however, it still exposes an internal attack surface through its APIs and resource control mechanisms. This results in a relatively limited attack surface when considered in isolation.

However, the CRA places significant emphasis not only on exposure, but also on system impact. In this respect, SAFE**RTOS**® plays a critical role, as all software execution ultimately depends upon the RTOS. Therefore, SAFE**RTOS**® forms a foundational element of the system architecture and has significant influence over the integrity, availability, and predictable operation of the overall system.

This combination of low external exposure and high internal criticality means that the level of concern is driven primarily by potential impact rather than accessibility. As such, SAFE**RTOS**® warrants a correspondingly rigorous level of assurance commensurate with its potential impact on overall system operation, where confidence in its design, verification, and lifecycle management is essential.

This becomes particularly evident when considering the presence of compromised or untrusted application code. If higher-level software contains vulnerabilities, unintended functionality, or has been influenced by a malicious actor, the role of the RTOS becomes critical in enforcing boundaries and maintaining system integrity.

Area	Assessment
External attack surface	Low
System impact	Very High
Reuse across products	High
Foundational functionality	Very High
Assurance expectation	High
CRA scrutiny level	High

Table 1. *CRA Risk-Based Assurance Positioning for SAFE***RTOS**

3.2 SAFE**RTOS**® Response

SAFE**RTOS**® is developed using processes intended to achieve high levels of integrity and assurance, underpinned by internationally recognised functional safety certifications. These certifications demonstrate compliance with rigorous industry standards and confirm that SAFE**RTOS**® has been developed to meet the demanding requirements of safety-critical applications.

Achieving this level of certification requires disciplined development practices, including structured design methodologies, strict coding standards, comprehensive verification and validation activities, and robust configuration management. Each stage of the development lifecycle is controlled and systematically assessed to confirm correctness, predictability, and freedom from unintended behaviour.

As a result, SAFE**RTOS**® provides a deterministic and reliable execution environment suitable for systems where software failure could result in unacceptable safety consequences. This safety-driven development life cycle aligns closely with the expectations of the CRA, offering a strong baseline of rigour, traceability, and assurance upon which security capabilities can be built.

3.3 Adding Security to the SAFE**RTOS**® development lifecycle

Within the context of the CRA, SAFE**RTOS**® must demonstrate not only functional correctness but also resilience against actions that could undermine system integrity. As a foundational component, it must enforce isolation, maintain the integrity of critical system resources and security boundaries, and remain trustworthy even when hosting untrusted application code.

To address these cybersecurity requirements, the SAFE**RTOS**® development lifecycle incorporates a structured Threat Analysis and Risk Assessment (TARA) process. This process identifies potential cybersecurity weaknesses and misuse scenarios, evaluates their impact, and defines appropriate mitigations, resulting in a comprehensive set of security requirements. The TARA methodology was originally introduced as part of SAFE**RTOS**®'s adoption of ISO/SAE 21434 [2] cybersecurity engineering practices and has since been applied more broadly to support the secure development of all SAFE**RTOS**® security-related products.

This analysis is performed alongside the established Hazard and Operability Study (HAZOP) [7] methodology used to derive safety requirements. Applying both approaches within a unified development process ensures that safety and security considerations are addressed in a consistent and structured manner.

The resulting security requirements are implemented using the same safety-critical lifecycle, ensuring full traceability from requirement through to verification. For security-related products, additional assurance is provided through dedicated security testing, including fuzz testing to evaluate robustness under unexpected inputs and penetration testing to assess resilience against realistic attack scenarios. Together, these activities provide evidence that SAFE**RTOS**® and its ESM can withstand both unintended misuse and deliberate attempts to compromise system behaviour.

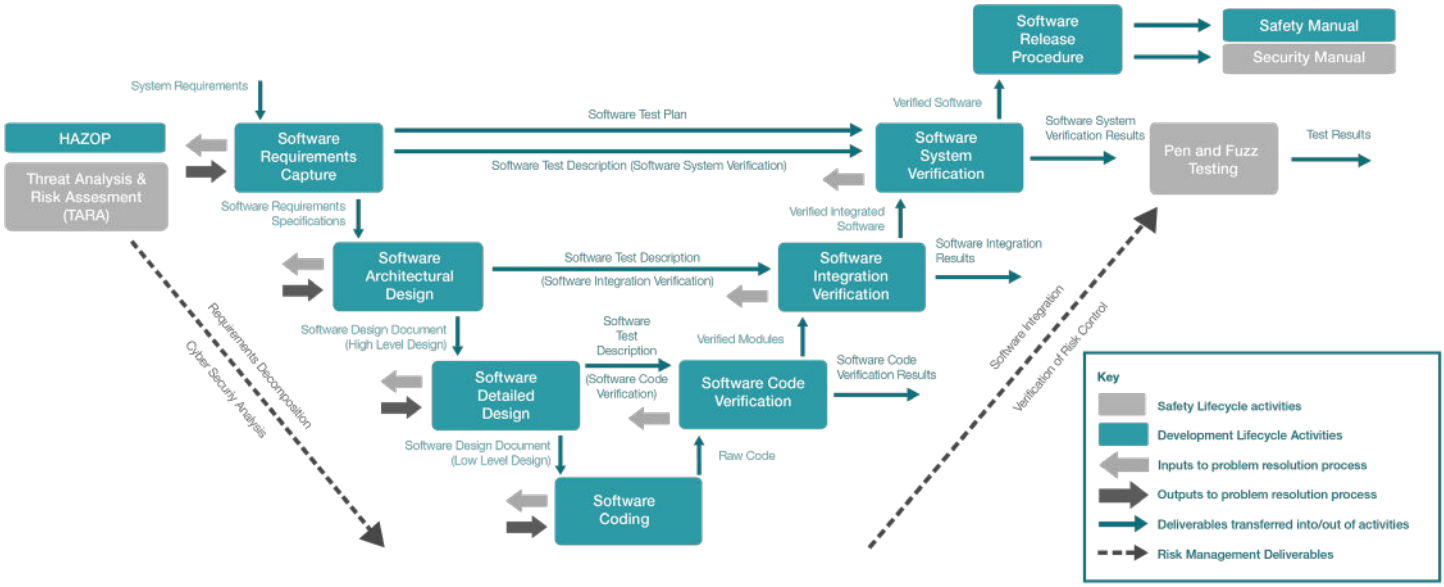

Figure 1. *How* SAFE**RTOS**® *Addresses CRA Cybersecurity Lifecycle Requirements*

3.4 SAFE**RTOS**® Specific Challenges

A key finding of the TARA is that SAFE**RTOS**® inherently exposes powerful APIs that allow developers to control system behaviour. These include mechanisms for suspending the scheduler, creating and managing tasks, and configuring memory protection regions. While essential for efficient embedded design, these capabilities introduce potential risk if misused.

The central challenge is determining whether a task invoking an RTOS API is acting legitimately and within its intended authority. If such mechanisms are not properly controlled, they may provide a pathway to compromise system integrity.

SAFE**RTOS**® is designed for deeply embedded systems where deterministic real time behaviour and certification evidence are primary requirements. Although SAFE**RTOS**® supports processors that provide either a Memory Protection Unit (MPU) or a Memory Management Unit (MMU), it employs a task-based protection model in which memory access is controlled through defined protection regions and task boundaries rather than separate process address spaces. Consequently, system security relies not only on memory protection, but also on the enforcement of least-privilege access to RTOS services, RTOS objects, and privileged functionality.

To address these challenges, WHIS has developed the Enhanced Security Module (ESM) extension to SAFE**RTOS**®, which is designed to limit the impact of compromised, malfunctioning, or unauthorised software components. By enforcing controlled interaction between software components and restricting unauthorised behaviour, it helps prevent faults or malicious actions from propagating beyond defined task boundaries.

3.5 SAFE**RTOS**® ESM Features

The objective of the ESM is to contain the impact of compromised, malfunctioning, malicious, or unauthorised user-mode tasks by preventing faults, misuse, or malicious actions from propagating beyond clearly defined security boundaries. This is achieved by enforcing strict isolation between user mode tasks, ensuring that each task can only access the resources, memory regions, and SAFE**RTOS**® services explicitly assigned to it.

The ESM includes the following key mechanisms:

Access Control Policy (ACP)
ACP restricts RTOS API access on a per-task basis. Each task is granted only the RTOS services it requires, with all other API calls denied.

At runtime, the RTOS enforces these permissions, preventing compromised or faulty tasks from accessing unauthorised APIs. For example, ACP can prevent tasks from creating or deleting tasks, altering scheduler behaviour, or stopping the scheduler unless explicitly permitted.

By enforcing the principle of least privilege and validating every API call, ACP reduces the attack surface, prevents privilege escalation, and limits the impact of a successful compromise.

Object Access Control Policy (OCP)
OACP controls access to RTOS objects on a per-task basis. Each task is granted access only to explicitly assigned RTOS objects, such as queues, semaphores, mutexes, and event groups. Attempts to access unauthorised objects are denied by the RTOS.

At runtime, the RTOS enforces these permissions, preventing compromised or faulty tasks from inspecting, modifying, or interfering with objects belonging to other tasks unless explicitly authorised.

By enforcing object-level isolation and the principle of least privilege, OACP aids in preventing task interference, limits the spread of faults or attacks, protects critical RTOS resources, and improves overall system security.

Memory Isolation
Memory Isolation uses the standard SAFE**RTOS**® MPU/MMU-based protection mechanisms.

User-mode tasks execute within dedicated memory regions, with access permissions enforced by the Memory Protection Unit (MPU) or Memory Management Unit (MMU). Tasks can access only their authorised code, data, and stack memory, while access to other tasks' memory, kernel resources, and protected system regions is denied. At

runtime, these protections prevent compromised or faulty tasks from reading, modifying, or executing unauthorised memory, helping to contain faults and security breaches.

By enforcing memory separation and least privilege, Memory Isolation protects kernel integrity, prevents unauthorised access to critical resources, and limits the impact of faults or attacks.

Unlike user mode tasks, privileged tasks retain access to the full system memory map. Their use should therefore be minimised and carefully controlled as part of the overall system security design.

SAFE**RTOS**® ESM is designed to minimise the amount of trusted software executing in privileged mode. Application functionality should execute in user mode wherever practical, with privileged execution reserved only for software that requires unrestricted access to system resources.

Data Obfuscation
Data Obfuscation protects critical kernel structures by using indirect references instead of exposing direct pointers to internal RTOS objects.

By limiting visibility of sensitive kernel data, it reduces opportunities for reconnaissance, facilitates the preservation of RTOS integrity, and provides an additional layer of protection alongside memory isolation and access control mechanisms.

Secure Portable Layer
The Secure Portable Layer enforces separation between user-mode and privileged-mode functionality. All transitions to privileged mode are mediated through trusted kernel interfaces, ensuring privileged operations are performed only through approved entry points and in accordance with the applicable access control policy.

By preventing direct access to privileged functionality, it assists in preventing privilege escalation, protects kernel integrity, and provides a secure foundation for other ESM security mechanisms.

Task Context Data Isolation
Task Context Data Isolation stores task context information within the RTOS-managed Task Control Block (TCB) rather than on the task stack.

By isolating critical context data, such as processor state and execution context, from application-owned memory, it prevents unauthorised access or manipulation, protects task integrity, and contains the impact of compromised or faulty tasks.

Security Event Monitor
The Security Event Monitor detects and reports unauthorised access attempts and policy violations. It monitors the system for events such as memory protection violations, unauthorised API calls, and access control breaches, helping to identify compromised, faulty, or malicious tasks.

When suspicious activity is detected, it can generate alerts or trigger application-defined responses to support incident containment.

By providing continuous security monitoring, it improves situational awareness, supports early threat detection, and maintains system integrity.

Collectively, these mechanisms ensure that a compromised user-mode task does not automatically grant access to RTOS services, RTOS objects, kernel data structures, or the memory regions of other tasks. By enforcing multiple layers of isolation and access control, the ESM assists in containing faults and security breaches to the smallest practical scope.

The ESM is an optional module that includes enhanced security capabilities for systems requiring additional assurance under frameworks such as the CRA, complementing the baseline controls present within SAFE**RTOS**®.

3.6 SAFE**RTOS**® Vulnerability Management

In alignment with CRA requirements, WHIS provides mechanisms for reporting, assessing, tracking, and communicating security-relevant issues and mitigations throughout the supported lifetime of SAFE**RTOS**®, supporting the CRA expectation for ongoing cybersecurity maintenance [1].

Suspected vulnerabilities are reported through an official support channel, ensuring controlled and confidential handling. Each report is analysed to determine its validity and potential impact. Where an issue is confirmed, it is formally documented and managed through the SAFE**RTOS**® errata system, providing a transparent mechanism for communicating findings, impacts, and recommended mitigations.

This means that vulnerabilities are handled consistently and responsibly, supporting ongoing security maintenance in line with CRA expectations.

3.7 SAFE**RTOS**® Software Bill of Materials (SBOM)

SAFE**RTOS**® has, since its introduction, been distributed with a comprehensive Software Version Description (SVD) document. This contains a complete inventory of the system's source and header files, including their location, integrity checksums, and associated licensing information.

The SVD provides many of the transparency and traceability characteristics expected from a modern SBOM, enabling visibility of the software's composition and supporting supply chain assurance activities [1]. It allows developers to verify that all expected files are present, correctly structured, and unaltered, ensuring that the software matches its intended, trusted state.

This level of visibility establishes both secure deployment practices and compliance with CRA expectations for supply chain transparency and software integrity.

3.8 Shared Security Responsibilities

SAFE**RTOS**® and its ESM provide the foundational controls necessary to support a secure system architecture, including access control, object isolation, memory protection, and security monitoring capabilities. However, it cannot determine the security policy of the final application or how these controls should be applied within a specific system. The system integrator is therefore responsible for defining which tasks operate in user mode or privileged mode, assigning appropriate API and object permissions, configuring memory protection regions, and applying the principle of least privilege throughout the system design. The effectiveness of the ESM depends upon the correct assignment of task privileges, access permissions, object ownership, and memory protection regions during system integration.

Where additional communication stacks, connectivity services, cryptographic modules, middleware, or application software are incorporated, the integrator is responsible for assessing and managing the resulting system-level cybersecurity risks.

SAFE**RTOS**® and its ESM have the mechanisms required to implement a secure and resilient architecture, but the overall security posture of the final product is determined by the system architecture, integrated software components, and security policies defined by the system integrator. Under the CRA, responsibility for the cybersecurity of the finished product ultimately rests with the entity placing that product on the EU market. The system integrator must therefore ensure that the overall security architecture is appropriate for the intended use of the product and that the controls provided by SAFE**RTOS**® and its ESM are configured, validated, and maintained accordingly.

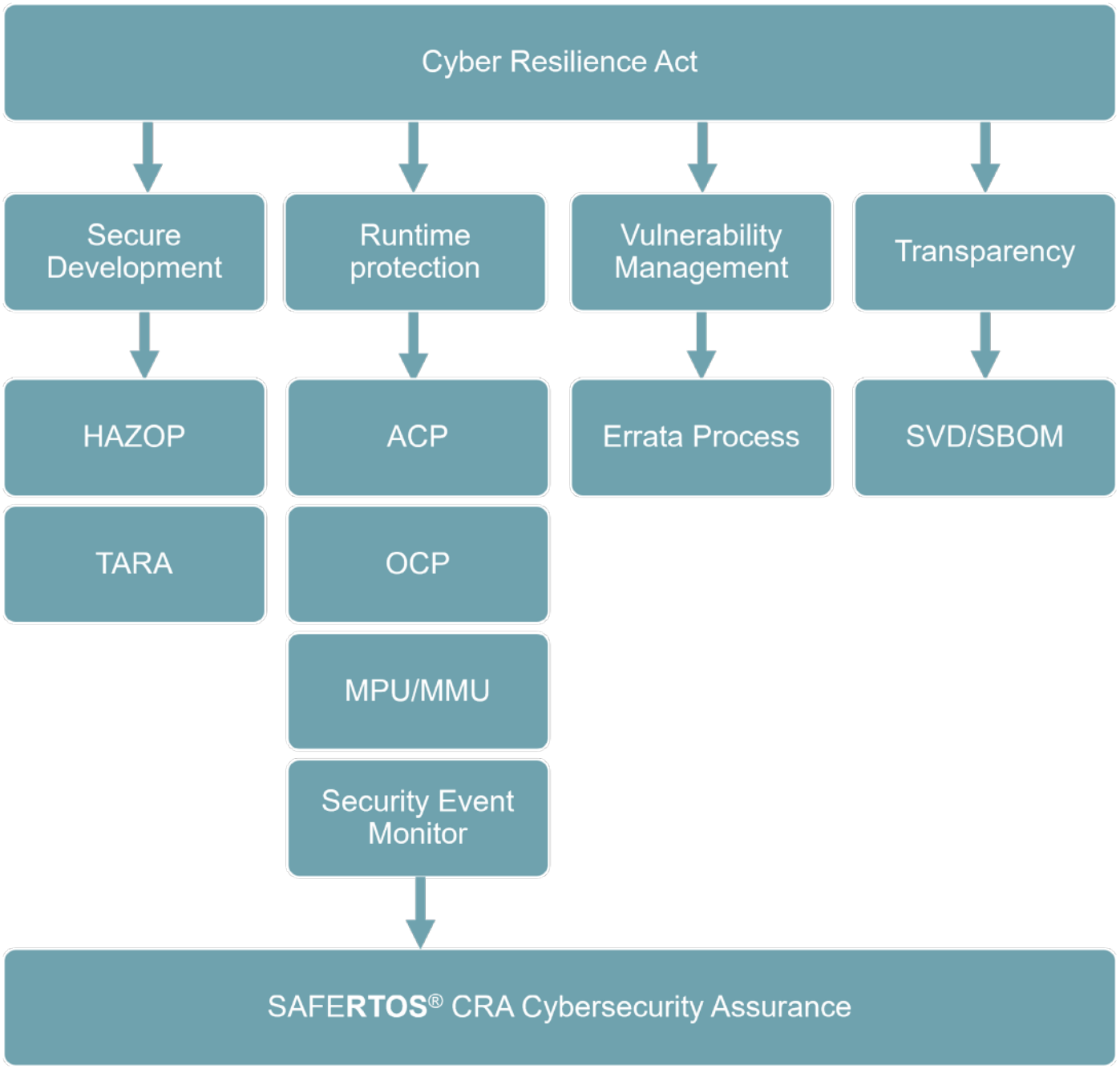

Figure 2. *How* SAFE**RTOS**® *and ESM Address CRA Cybersecurity Lifecycle Requirements*

CHAPTER 4 SAFE**RTOS**® and the CRA

4.1 SAFE**RTOS**® Alignment with CRA Expectations

The following table summarises how SAFE**RTOS**® and its ESM address the principal cybersecurity lifecycle expectations of the Cyber Resilience Act and provides traceability between CRA expectations, SAFE**RTOS**® security mechanisms, and supporting evidence.

CRA Expectation	SAFE**RTOS**® Response	Supporting Evidence
Secure by Design	Security requirements derived through TARA and integrated into the development lifecycle.	TARA process, security requirements, lifecycle traceability.
Minimise Attack Surface	Core RTOS provides no networking, cloud connectivity, or external communications by default.	Deliberately limited functionality and defined operational boundaries.
Protection Against Unauthorised Actions	ESM restricts API access, object access, and memory access according to defined policies.	ACP, OACP, MPU/MMU protection.
Isolation and Containment	Compromised user mode tasks are constrained from affecting other tasks or kernel resources.	Memory Isolation, ACP, OACP, Secure Portable Layer.
Protection of Critical Assets	Access to privileged functionality and kernel resources is controlled and monitored.	ACP, OACP, Secure Portable Layer, Security Event Monitor.
Security Verification	Security requirements are independently verified through dedicated testing activities.	Fuzz testing, penetration testing, security verification.
Vulnerability Management	Security issues are reported, analysed, tracked, and communicated through controlled processes.	Vulnerability reporting process and errata system.
Security Maintenance	Vulnerabilities can be assessed and communicated throughout the product lifecycle.	Ongoing support and errata management.
Transparency and Traceability	Full software inventory and version information supplied with each release.	Software Version Description (SVD).
Supply Chain Visibility	Software composition can be identified and verified.	SVD supports modern SBOM expectations.
Assurance and Governance	Development performed using certified safety and security processes with full lifecycle traceability.	Compliance with internationally recognised safety and security development standards and associated lifecycle processes.

Table 2. *How* SAFE**RTOS**® *and ESM Address CRA Cybersecurity Lifecycle Requirements*

The table demonstrates how SAFE**RTOS**® and its ESM address the principal cybersecurity lifecycle expectations of the CRA through a combination of secure development practices, runtime protection mechanisms, vulnerability management processes, and software transparency measures.

CHAPTER 5 Conclusion

5.1 Conclusion

The Cyber Resilience Act (CRA) introduces a fundamental shift in how software components are treated within the European market, placing clear cybersecurity obligations on suppliers and recognising software as a regulated product rather than a passive artefact. For component providers, this creates direct responsibility for ensuring secure development, effective vulnerability management, and full transparency throughout the software lifecycle.

Within this framework, SAFE**RTOS**® represents a software component that warrants a high level of assurance due to its foundational role within embedded systems. Although it has a limited external attack surface, its role as the core execution, scheduling, and resource management layer makes it critical to overall system integrity. Therefore, its level of concern is driven primarily by its systemic impact rather than its exposure, leading to elevated assurance expectations under the CRA.

SAFE**RTOS**® addresses these expectations by building upon a strong foundation of functional safety certification, which provides rigorous development processes, traceability, and verification. This safety-driven baseline is extended through the integration of security practices within the development lifecycle, including the use of Threat Analysis and Risk Assessment (TARA), combined with established safety analysis methods such as HAZOP.

To mitigate RTOS-specific risks, SAFE**RTOS**® implements structured control mechanisms, including access control and Object Access Contro lpolicies, ensuring that system services and resources are used only in authorised and controlled ways. These controls form part of the SAFE**RTOS**® Enhanced Security Module (ESM), which is designed to limit the impact of compromised or untrusted application code by enforcing task isolation, constraining API usage, and controlling access to system resources. In doing so, the ESM helps prevent faults or malicious behaviour from propagating beyond defined boundaries, thereby supporting overall system integrity.

In addition, SAFE**RTOS**® is in line with CRA lifecycle requirements through a structured vulnerability management process and long-standing support for software transparency via its Software Version Description, which aligns with modern SBOM expectations.

Overall, SAFE**RTOS**® with the Enhanced Security Module (ESM) demonstrates strong alignment with the cybersecurity principles and lifecycle expectations of the CRA. Its combination of a safety-certified development process, structured security engineering, vulnerability management, and runtime enforcement mechanisms provides a high level of assurance for systems requiring strong isolation, controlled resource access, and containment of compromised software. When correctly configured by the system integrator, SAFE**RTOS**® with ESM provides a robust foundation for the development of secure and resilient embedded systems within the evolving European regulatory landscape.

Contact Information

Your Feedback

We would be pleased to receive your comments and any suggestions for improvement to our software and documentation. Please email us at: support@highintegritysystems.com

Contact WITTENSTEIN high integrity systems

Address: WITTENSTEIN high integrity systems
3 Rivergate, 1st Floor
Temple Quay
Bristol
BS6 1EW
England

Americas +1 408 625 4712
ROTW +44 1275 395600

Email: support@HighIntegritySystems.com

Website www.HighIntegritySystems.com