Collaborative Security Framework

Version 2.0

23 December 2025

Abstract

The Cross-Workstream on Security presents this Collaborative Security Framework which aims to establish a unified approach to security within the IPCEI-CIS project by fostering collaboration among partners.

This framework will identify and standardize security components, promote interoperability, and ensure baseline security compliance across all identified building blocks.

TABLE OF CONTENTS

IPCEI-CIS Partner	Co-author
NBIP (COORDINATOR)	HANS PUNT
	JOHAN LAMPRECHT
ADVA NETWORK SECURITY	Wolfgang Keil
	Mario Wenning
OKTAWAVE	Pawel Maslowski
ATOS EVIDEN	Patrice Bleuze
	Thierry Winter
	Hafeda Bakhti
FONDAZIONE BRUNO KESSLER	Roberto Doriguzzi-Corin
	Marco Zambianco
	Daniele Santoro
	Salvatore Manfredi
	Stefano Berlato
LINDNER GROUP	Lucas Hollweck
SECUNET SECURITY NETWORKS AG	Matthias Votisky

Executive Summary

This document presents an updated version of the Collaborative Security Framework (CSF) describing all major security components, in relation to each other and related to the complete 8ra (IPCEI-CIS) project. Consequently, we are establishing a comprehensive, holistic security framework working toward the security-by-design principle.

It is based on a collaboration of all partners connected to the 8ra (IPCEI-CIS) project describing their security contributions and offering an answer to all security-related development questions. The completeness and detail of this document will represent the security-based cooperation of all partners in the 8ra project.

This security framework will identify and standardize security components, promote interoperability, and ensure baseline security compliance across all identified 8ra security topics. Since security is not static, this document will be constantly updated.

This version is already integrated in the 8ra reference architecture of Workstream 2 and the infra-structure blueprint of Workstream 1 where we are currently working on integrating it into the eDC blueprint and Workstream 3.

Scope

The CSF aims to establish a unified approach to security within the IPCEI-CIS project by fostering collaboration among partners. This approach is based on the fact that every project within the 8ra project will have at least one major security component where, in reality, there will be multiple security components per individual project.

This framework will identify and standardize security components, promote interoperability, and ensure baseline security compliance across all identified 8ra security topics. We are using a building block structure where every project in the 8ra project will deliver one or more building blocks.

For this document we are using the following process:

CSF Process

1. Identify security components within IPCEI-CIS projects, the building blocks.
2. Classify components based on topic, purpose, architectural integration layer, and additional relevant attributes.
3. Analyze and document differences between similar or duplicate components.
4. Collect technical and functional details about each security component's operation.
5. Identify gaps and propose solutions for missing essential security components.
6. Establish a baseline security compliance posture and a standardized integration language.
7. Evaluate the security posture compliance for each component through the various pilots in the 8ra project.
8. Facilitate collaboration with partners to align with baseline security requirements.
9. Develop a standardized integration language to ensure compatibility across components.
10. Iterate through the steps of this process to reach a security baseline and maturity.

Phase One: Information Gathering

Security Building Blocks Identification and Classification

Understanding each individual project's status regarding security, and understanding their needs, is started by taking inventory. For this we have designed a building block format where every partner connected to the Cross-Workstream Security (xWSS) has completed several of these templates.

These security Building Blocks, provided by partners, are being categorized based on their domain, purpose, relevant architecture integration layers, compliance attributes, and other security-related attributes that enable cross-partner and cross-platform integration. This classification will facilitate the mapping of components to a standardized security framework, improving interoperability and risk assessment.

A collaborative fact-finding exercise will be conducted with partners to identify and analyze similar security components. Technical discussions will help distinguish nuanced differences between duplicate or overlapping components. This process is ongoing where all new partners in the xWSS are asked to send in their building blocks.

The current status and analyzes of these provided building blocks are described in Annex 1 Building Blocks Overview.

API Identification and Classification

As part of the first phase of the CSF, a comprehensive inventory of security-related Application Programming Interfaces (APIs) is being compiled and analyzed. The xWSS team is identifying all APIs exposed by the security building blocks provided by partners and classifying them based on their purpose, domain, and relevant architecture integration layers. This classification process highlights the relationships and dependencies between components – for example, which security services offer similar functionalities or interfaces and reveals any overlap in API capabilities across different partner solutions. By cataloging APIs and their attributes such as authentication methods and data formats, the CSF ensures a clear mapping of how security components can interoperate within the 8ra ecosystem, laying the groundwork for a standardized integration approach.

Building on this inventory, the CSF will work to define a common integration language for security services, aligning with the broader reference architecture's goal of interoperability. In practice, this means identifying opportunities to harmonize or translate APIs so that partners' security modules can plug into a unified framework seamlessly. Where multiple components serve a similar role, their APIs will be compared and any inconsistencies addressed either by adopting a best-of-breed interface or by developing an adaptation layer to bridge differences. This two-pronged approach of classification and harmonization should yield an "API blueprint" that specifies standard methods for accessing security functions across the project. Ultimately, the API identification and classification effort enables the CSF to reduce integration friction, ensure all security components communicate effectively, and provide a uniform experience to both internal and external consumers of security services.

Technical and Functional Documentation

The CSF will collect and document detailed technical and functional specifications aligned with the 8ra Reference Architecture to ensure interoperability and compliance. This documentation process involves gathering information on how components are compiled, including the programming languages used, dependencies, and compilation methods. Additionally, the CSF will record details about API calls, authentication mechanisms, data structures, and supported protocols to ensure seamless integration with the IPCEI-CIS platform.

Standard frameworks or open-source software implemented within each component will be identified, allowing for an analysis of commonalities and potential security vulnerabilities. Furthermore, the documentation will capture compliance with industry standards such as OAuth, OpenID Connect, TLS, and encryption libraries. This structured approach will provide a comprehensive view of each security component's functionality, enabling the CSF to assess compatibility, evaluate security risks, and develop a standardized integration approach for all partners.

Identifying Gaps and Proposing Solutions

To identify gaps in security coverage and deliver a comprehensive security approach, the CSF will conduct a comprehensive analysis of existing Security Building Blocks and APIs and compare them against the topic setup described further on in this document. Industry best practices with regard to security, regulatory requirements, and emerging threats and/or future technologies will be part of the gap analysis where we also identify future threats.

This analysis will highlight any missing essential security components and areas where existing solutions may be insufficient.

Once gaps are identified, the CSF will engage with partners to collaboratively design mitigation strategies, whether through the enhancement of existing solutions, the development of new components, or procurement of external services. Additionally, risk assessments will be performed to prioritize solutions based on their impact and feasibility.

The framework will ensure that identified gaps are addressed through structured recommendations and coordinated implementation efforts among partners.

Phase Two: Architecture and Security Topic Description

Based on the newly-published architecture design in the 8ra project the CSF provides an overview below of this architecture and the allocation of, by the xWSS identified security topics connected to this architecture.

Architecture Design 8ra

The IPCEI-CIS Reference Architecture focuses on the integration and orchestration of infrastructure and workload lifecycle management across a hybrid edge-cloud environment. For this purpose, it organizes the functional components in layers that provide different levels of abstraction and helps to manage the complexity of a Multi-Provider Cloud-Edge Continuum, and domains, that represent cross-cutting aspects applicable to all layers (e.g. management or security).

The architecture includes the following core layers:

1. **Application Layer:** It provides the functionality required to design and develop applications and functions and to expose them for use. It includes end-user applications and function catalogues.
2. **Data Layer:** It contains functionalities for data collection, processing, and exchange, making it simple, scalable, sustainable, trustworthy and distributed over the Cloud-Edge Continuum. These data artifacts can be applied both in end-user applications, by third parties, and internally to optimize the edge-cloud system at different levels.
3. **AI Layer:** Its aim is to create a fundamental set of advanced functionalities to a hybrid approach: sets of data and processing resources distributed or centralized depending on the use cases. As with data artifacts, they are also applicable to end-user applications or to internal edge-cloud optimization.
4. **Service Orchestration Layer:** It addresses the integration and management of multiple cloud and edge services and applications in a unified and automated way across diverse multi-provider environments. In the context of multi-cloud and edge ecosystems, service orchestration is essential for handling the complexity of deploying and managing applications at scale, especially when dealing with distributed resources across cloud and edge infrastructures.
5. **Cloud-Edge Platform Layer:** It allows the allocation and lifecycle management of resources over the virtualized Cloud-Edge Continuum and the deployment and chaining of application components to deliver a service over a certain geographical area. It includes components that facilitate and manage the complexity of computing environments that require multiple cloud technologies and providers, enabling interoperability, compatibility, and portability across a Multi-Provider Cloud-Edge Continuum.
6. **Virtualization Layer:** It provides the necessary abstraction over the physical hardware resources (compute, storage, networking) to facilitate their dynamic allocation, usage, and sharing, hiding the heterogeneity of the hardware infrastructure. It provides a virtual runtime environment in which the applications are deployed and executed. This layer is linked to the Software-Defined Network (SDN) or other management systems of public and private networks.

7. **Network Systems, SDN Controllers:** They provide the capabilities to manage physical and virtualized/cloudified networking elements to build network services in the geographically distributed Cloud-Edge Continuum.
8. **Physical Layer:** It includes all the physical hardware resources required to implement the Cloud-Edge Continuum (compute, storage, and networking). It is closely connected to the physical network infrastructure that supports communication among the computing nodes in the continuum and the connectivity of users to that continuum.

Identified security topics listed by architecture layer:

1. **Physical Layer**
   - Hardware-tampering protection
   - Secure hardware supply chains
   - TEMPEST shielding and RF isolation
   - Hardware-based encryption modules (TPM, HSM)
   - Secure boot and firmware validation

2. **Data Layer**
   - MAC address spoofing protection
   - Secure handshakes (e.g., WPA3)
   - VLAN and switch security
   - Wireless communication security (Wi-Fi, 5G)

3. **Network Layer**
   - Secure routing protocols (e.g., RPKI, BGPsec)
   - IPsec and VPNs
   - IP spoofing prevention
   - IPv6 privacy and security
   - Network segmentation and firewalls

4. **Transport Layer**
   - Transport Layer Security (TLS)
   - DDoS mitigation
   - Secure session management
   - Rate limiting and flow control mechanisms

5. **Session Layer**
   - Token-based session management
   - Identity Federation and SSO
   - Session hijacking protection

6. **Presentation Layer**
   - Data encryption and decryption
   - Compression and encoding security
   - Data integrity and checksum verification

7. **Application Layer**
   - Web application security (e.g., OWASP Top 10)
   - API security
   - Secure software development lifecycle (SDLC)
   - Identity and access control

- User privacy controls
- Multi-Factor Authentication (MFA)

The above listing is non-exhaustive and based on gathered information up to the day of publication of this document.

Topics in the Collaborative Security Framework

Based on a holistic view of security in the 8ra project, and the available information during the information gathering sessions, we have determined ten major topics which we need to cover.

Each topic has a topic owner who is responsible for gathering information from project partners, structuring all available information into a clear description which contributes to the scope of this document and engaging with 8ra partners interested in this topic. Each major topic has several sub-topics.

Topic 1: Identity & Access Management

1.0 General Description

This topic provides the foundation of trust across the Cloud-Edge Continuum by ensuring that identities are properly verified and that access to resources is consistently enforced. Identity & and Access Management (IAM) remains at the core of every secure system, but in an environment defined by distributed computing, edge devices, and cross-border data flows, IAM must evolve into a unified, adaptive capability. The CSF will commit to the delivery of a reference architecture that integrates decentralized identity, contextual access controls, multi-factor authentication, identity federation, and Zero Trust principles within the reference architecture cross-workstream. By aligning contributions from partners, IAM becomes not only a technical safeguard but also a shared trust layer across Europe's digital infrastructure.

1.1 Decentralized Identity

Decentralized Identity (DID) introduces a paradigm shift, moving control of identity away from centralized providers and giving ownership to individuals and organizations themselves. This model allows users to present only the attributes required for a transaction, enhancing privacy and reducing opportunities for impersonation or mass breaches. The CSF will classify ongoing DID experiments and align them with European regulatory and interoperability requirements. In doing so, it ensures that the 8ra ecosystem can leverage portable, tamper-resistant credentials as the foundation for trust across services, while reducing dependency on foreign identity providers.

1.2 Role-Based Access Control

Role-Based Access Control (RBAC) continues to be essential in structured enterprise environments, offering a scalable way to assign permissions based on job functions. However, in the cloud-edge reality, static role models alone are insufficient. Evolving toward attribute-based and context-aware access is necessary to capture dynamic risk signals, such as device health, network conditions, or user behavior. The CSF will map partner approaches to RBAC and emerging Attribute-Based Access Control (ABAC) models, creating guidance on when and how to deploy each. This ensures that access decisions remain both manageable and adaptive to rapidly changing conditions.

1.3 Zero Trust Architecture

Zero Trust Architecture represents a fundamental departure from perimeter-based security. Its principle of “never trust, always verify” requires continuous validation of identity and context at every access request. The CSF will operationalize Zero Trust across the shared framework by embedding guidance for continuous authentication, micro-segmentation of resources, and risk-based policy enforcement. Through this, the ecosystem gains resilience against insider threats, credential theft, and lateral movement, reinforcing IAM as the active enforcement point for Zero Trust principles.

1.4 Multi-Factor Authentication

MFA and biometrics are essential to strengthening the assurance of identity verification. From hardware tokens and mobile push notifications to biometric techniques such as fingerprint, facial, or behavioral recognition, MFA adds layered resilience. Yet these approaches raise concerns around usability, privacy, and spoofing resistance. The CSF will compile best practices for deploying MFA and biometrics, balancing user convenience with high assurance levels, while ensuring alignment with GDPR and ethical safeguards. This guarantees that authentication mechanisms remain trustworthy, even as adversaries develop more sophisticated bypass techniques.

1.5 Identity Federation

Identity federation, powered by standards such as OAuth2 and OpenID Connect, enables secure interoperability across services and organizations. Federation is critical for cloud-edge ecosystems, where no single provider controls the entire identity lifecycle. However, federation introduces dependencies on trust anchors, token integrity, and cross-domain governance. The CSF will classify and integrate partner experiences in building federated identity systems, capturing both success stories and failure modes. This body of knowledge will inform a set of baseline federation patterns, ensuring that inter-organizational access is reliable, secure, and scalable across the European digital market.

1.6 eIDAS Legislation

The eIDAS legislation was published in 2014 and first version of its Technical Specifications in January, 2016. These described the architecture, interfaces and cryptographic requirements of eIDAS-related IAM systems. The IAM system of E-Group was developed based on these Technical Specifications and that was the reason why Hungarian Government could start operating it as the citizen gateway (like an eIDAS node). Time has passed and current version of eIDAS modified the interface requirements. Instead of XML-based SAML, JSON-based OID-based protocols shall be supported in eIDAS ecosystem, such as OID4VCI and OID4VP. This transition requires protocol mapping that is not easy for all cases (e.g. just Implicit Flow of OID4VP or OIDC is currently supported by eIDAS-specific SAML; reversed application of cryptographic layers hardens ensuring end-to-end confidentiality and authenticity verifications together; lack of finalized standards means uncertainty for session/consent invalidation/logout functions of OIDC on which OID4VCI and OID4VP rely on). But still, this transition has to be made and the security level of original operation has to be

maintained. Beyond interface mapping, the transition of cryptographic layer has to be also supported: signature and encryption functions shall be able to apply quantum-safe algorithms. NIST PQC announced the winner quantum-safe algorithms in 2022 and it seems that CNSA roadmap is sustainable: such algorithms have been implemented in major cryptographic libraries (such as OpenSSL 3.5, Java 24, Bouncy Castle 1.81, .NET 10.0 rc1), finalized data structure standards will be published soon (for applying ML-KEM in JSON/XML Security layer or providing quantum-safe SSL/TLS communication channel), or have already been published (such as IETF RFC 9881 and IETF RFC 9882 for CMS format of ML-DSA). So, application developers can move on, they can start modifying their own solutions. And so, these are the objectives E-Group in IPCEI-CIS project.

Beyond these subtopics, IAM must also address hybrid realities: identities spanning multiple clouds, IoT devices, and edge nodes operating under varying levels of trust. Cross-domain challenges such as just-in-time access provisioning, privileged account management, and lifecycle automation become critical for operational security. By coordinating classification and alignment of these efforts, the CSF ensures that IAM evolves from a patchwork of local solutions into a federated, adaptive trust fabric.

Ultimately, IAM within the CSF delivers more than technical enforcement. It creates a shared foundation of digital trust, ensuring that every user, device, and service can be verified and governed consistently across Europe. In doing so, it strengthens the continent's digital sovereignty by embedding secure, privacy-preserving identity practices at the heart of its cloud-edge security architecture.

Topic 2: Network Security

2.0 General Description

Network Security covers the protection of data in transit and the defense of network infrastructure. This includes DDoS protection, firewalls, secure network segmentation, encrypted communications (e.g. VPN/IPsec tunnels), intrusion detection/prevention systems, and secure routing protocols. The CSF will consolidate and classify network security components provided by different partners, integrating them into a cohesive multi-provider network defense strategy. By mapping each solution from secure DNS to DDoS mitigation into the framework, the CSF ensures that the entire Cloud-Edge Continuum benefits from a coordinated network security posture, and it identifies any gaps where additional controls or partner contributions may be needed.

2.1 Firewalls and Segmentation

This section comprises three subsections that cover the topics of firewalls and segmentation individually and in combination. First, we define the basic requirements and functionalities that the firewall must fulfill to perform its task of network protection. The second subsection introduces network design choices for segmenting the network to enhance security. Lastly, we describe the interdependencies between firewalls and network segmentation, focusing on the interplay and joint dependencies.

2.1.1 Firewalls for distributed Network Defense

Following the open-source paradigm of the Cloud-Edge Continuum, the firewall must be based on open-source code bases and facilitate multiple tenants, which is closely related to identity and access management. As a reference, the open-source clouds, such as OpenStack, Kubernetes, Proxmox, CloudStack, and OpenNebula, fulfill comparable requirements. To ensure scalability and facilitate a wide deployment of 8ra, the firewall administration must be automated. Besides scalability, the automation enables reproducibility in administration and prevents human-introduced errors. Furthermore, the automation must align with security operations to follow comparable models and requires synchronization between both topics.

To fulfill the requirements, the firewalls have to provide the following functionalities:

- Perimeter protection
- Internal segmentation
- Host protection
- Tenant-specific administration

The perimeter protection restricts public access to cloud APIs and must block external threats. However, there is a trade-off between blocking and ease of usability that needs to be addressed. The internal segmentation must ensure mutual isolation between tenants, such that hypervisors, storage, and network management cannot interfere. Furthermore, individual nodes within the Cloud-Edge Continuum must be protected on a host-level basis, depending on their functionalities and the data they store. Lastly, users must be able to customize security and firewall rules for self-managed VMs and/or containers.

On a high-level basis, these functionalities require a stateful packet filtering for tracking valid connections and blocking malicious traffic. Furthermore, the application and control plane must be isolated, which is closely related to the aspect of network segmentation. There must be role-based policies for mutually separated tenants, and strict monitoring and logging must be implemented. To ease the separation of tenant-specific traffic and simplify network protection, the Cloud-Edge Continuum should be segmented into different network types. The document covers this aspect in the following subsection.

2.1.2 Network Segmentation

Besides the firewall for facilitating network protection, segmenting the network into multiple VLANs and restricting routing limits the harm in the case of security loopholes. With respect to the OSI model, segmentation should be performed across multiple network layers. First, the management network must be isolated from the other networks, which requires a separate VLAN and a separate IP network. Furthermore, the management network is only reachable via a VPN or the administrative network. The firewall rules for the management network are strict due to the potential damage that could result from security breaches.

In addition to the management network, there must be an isolated network for hypervisors and computing nodes. Since hypervisors and computing nodes interact with storage and controllers for distributed computing tasks, the range of protocols must be limited. Furthermore, there should be no direct access from the public or by tenants. Similarly, the storage network must be segmented and isolated from tenant and public networks. Neither of these networks must be directly accessible by tenants' networks, but it must be used in the background.

To accommodate multiple tenants as desired by 8ra, the tenant networks must be isolated from one another. In cases where required interactions occur between tenants, connectivity can be established on the IP layer via a routed firewall that is also capable of monitoring. The access of tenants must be controlled through entry points via HTTPS with optional reverse proxies, Web Application Firewalls (WAFs), and MFAs. The access management facilitates encryption and tenant identification.

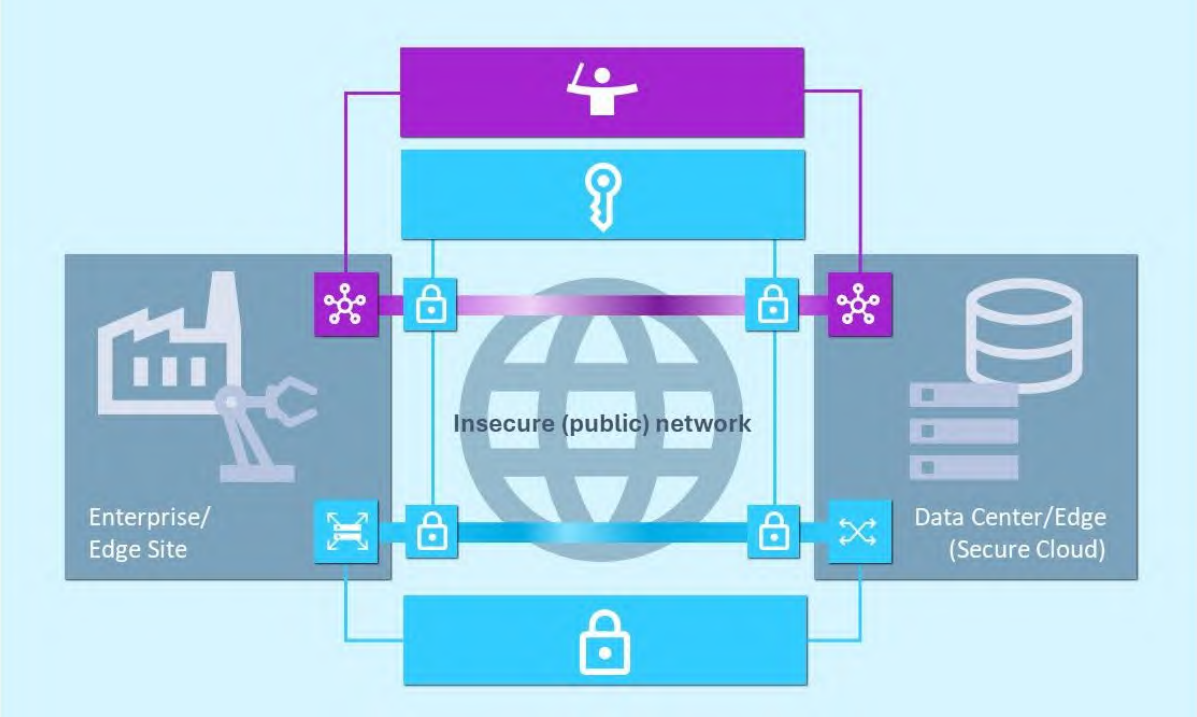
2.1.3 Interplay between Firewall and Segmentation

The two previous subsections describe firewall and segmentation individually; furthermore, the subsection instructs on setting up a secure environment for the Cloud-Edge Continuum. However, the interplay between firewall and network segmentation is crucial for achieving a secure and operational cloud infrastructure.

To link the two measures, the firewall and network segmentation, the firewall rules must represent the network segmentation through a separate set of rules for each network segment. Furthermore, network segmentation must be implemented by the firewall and the network design at layers two and three of the Open Systems Interconnection (OSI) model. Mutual connectivity between tenants' network segments must be ensured by routed firewalls, which interconnect tenants' networks at Layer 3 under the condition that firewall rules are fulfilled. In addition to both measures, firewall and network segmentation, application-aware blocking should be implemented by using a WAF that acts on layer 7 of the OSI model for the login of tenants and APIs facing the public. The measures and recommendations in this section are fulfilled by open-source cloud software, such as OpenStack, Kubernetes, and Proxmox, and should now break the paradigm of 8ra.

2.2 Tunneling Protocols

Since the previous section focuses on defending the network infrastructure, this section covers the protection of data in transit by encrypting them. To transport the encrypted data across the network tunneling needs to be applied. Depending on what transport layer the tunneling is applied, different tunneling protocol can be used. The suggested network segmentation is based on layers two and three, which gives a clear preference for Media Access Control Security (MACsec) and Internet Protocol Security (IPsec) for facilitating secure connectivity. In addition to encryption at layer two and/or three, further encryption efforts are discussed in the third subsection, which covers application-layer encryption or encryption on the data layer.



Independent of the tunneling protocol are algorithms for the key encapsulation mechanism and the encryption. Symmetric encryption should be based on the Advanced Encryption Standard (AES) algorithm. For the key encapsulation mechanism, the target of crypto agility should be approached. Due to the threat of quantum computing, classical key encapsulation based on elliptic curves or factoring large prime numbers is insufficient for long-term security. Hence, encryption devices that are re-configurable with respect to the key encapsulation mechanism or use post-quantum cryptography must be used to ensure protection in transit.

2.2.1 MACsec

Network segmentation is a requirement for securely operating a shared cloud infrastructure. Hence, segmentation into multiple Virtual Local Area Networks (VLANs) is a requirement, especially with respect to the separation between management and data plane. VXLAN provides a solution for large-scale deployment of cloud infrastructure and the distribution of resources within network segments, facilitating protection in transit in combination with MACsec. Alternatively, one can use VXLAN in conjunction with IPsec to protect data in transit and support distributed network segments at the layer 2 level.

The advantages of MACsec include high line rates and minimal impact on communication overhead. Hence, especially within data centers, the protection of data in transit is suitable for MACsec. However, the combination of VXLAN and MACsec also enables a distributed Layer 2 network segment based on the encapsulation of Layer 3 User Datagram Protocol (UDP) packets. This creates a two-fold tunneling effect – first, the encryption tunnel between the endpoints protects the data. Second, the LAN is tunneled through a higher-layer network to ensure connectivity. Furthermore, this approach is transparent to the IP network and does not require additional configurations.

Regarding data protection, MACsec also encrypts IP, UDP/TCP, and VXLAN headers. Source and destination MAC addresses are readable, and additional tags for Layer 2 connectivity are available.

2.2.2 IPsec

In contrast to MACsec, IPsec influences the configuration of the Layer 3 network and incurs a larger overhead. However, it can also be combined with VXLAN to protect the data in transit. Depending on the overlay network, it can be combined with MACsec. Alternatively, one can use MACsec within data centers and IPsec for WAN connectivity between different sites. IPsec may limit throughput due to the lack of hardware with comparable data rates to those of Layer 2.

Regarding data protection, IPsec also encrypts User Datagram Protocol (UDP)/ Transmission Control Protocol (TCP) headers. Source and destination MAC and IP addresses are readable, and additional tags for layer 2 and 3 connectivity are available. For secure data in transit, the recommended encryption combines both tunneling protocols, MACsec and IPsec. The underlay network must rely on MACsec for all links that carry sensitive data. The overlay network must rely on VXLAN protected by MACsec or IPsec and must fulfill the requirements for network segmentation. For reachability and interfaces facing the public, IPsec tunnels are recommended. In addition to these protection schemes for data in transit, further protection schemes are available for the application layer, as outlined in the following subsection.

2.2.3 Application Layer Data Protection

The two previous subsections describe the recommendations for data protection within layers two and three. In addition to these measures, encryption must be implemented on the application layer (layer 7) between clients and servers. The recommended standard for encrypting the application data layer is TLS, which is embedded in HTTPS as an example of an exemplary application. In case of stricter security and encryption requirements, TLS can be extended to mutual TLS. However, the mutual extension requires valid certificates on both ends, making the public key infrastructure more complex.

Again, the differentiation into network segments and services within the distributed cloud infrastructure plays a crucial role in protecting data on the application layer. User-facing services must rely on TLS for end-to-end encryption and service authentication. Furthermore, services and microservices within the cloud infrastructure must rely on TLS to approach the Zero Trust policy. The inherent advantage of the layered architecture is that it encrypts sensitive data even in the event of a malfunction of the underlying physical and virtual network.

The following list summarizes the security layers within this section:

- MACsec (L2) encrypts the physical network infrastructure
- IPsec (L3) ensures tunneling across untrusted networks for interfaces to the public
- TLS/mTLS (L7) encrypts and authenticates services and applications

Following the paradigm of open-source software usage and building on the state-of-the-art allows for the reuse of software for IPsec and TLS, such as OpenSSL or the Linux kernel's IPsec stack. OpenSSL also provides extensions for PQC key encapsulation mechanisms that can be added. Since MACsec is often implemented with the use of hardware acceleration to foster high throughputs, dedicated equipment with certifications must be used.

2.3 Secure Routing and Path Validation

After focusing on intrusion detection and prevention, this section describes standards and best practices for routing securely and validating the paths in the overlay network. This section is closely related to the previous one, as it aims to prevent attackers from misconfiguring the network and thereby increasing the damage. Similar to network segmentation, it is a preventive measure that hardens the cloud infrastructure. The first subsection covers security mechanisms for the Border Gateway Protocol (BGP) routing protocol. Second, we describe measures for SRv6 in a similar fashion.

2.3.1 BGP Security Mechanisms

Routing in the overlay between distributed data centers required for a cloud infrastructure can be realized with BGP. Due to the complexity of configuring large networks, attackers influencing BGP can significantly amplify their impact, resulting in outages. Different standards provide measures to secure routing and enable path validation. First, RFC 5925 enables strong authentication for TCP sessions used by BGP. Second, RFC 8205 enables path validation of BGP updates, preventing tampering with them. Lastly, RFC 6480 enables the validation of BGP route origins and prevents route hijacking.

2.3.2 SRv6 Security Mechanisms

This subsection recommends security measures for SRv6 routing similar to the previous measures for securing BGP. The header data of the SRv6 routing protocol imposes the threat of disclosing topology information. The countermeasure involves authenticating the origin of the communicating routing information. Near the authentication, there is a recommendation to mutually whitelist all nodes that are allowed to execute SRv6 functions. This approach is transferable to a centrally controlled network utilizing SDN. Nodes must use segment information from trusted controllers only. For path validation, the SRv6 header data must be extended by inserting Type-Length-Value (TLV) messages. Combined with a Public Key Infrastructure (PKI), the extended header data facilitates path validation.

2.4 Secure DNS (DoH/DoT)

to be added in a future version of the document

2.5 Intrusion Detection/Prevention

to be added in a future version of the document

Topic 3: Data Security

3.0 General Description

Data Security is concerned with protecting data at rest, in transit, and in use. It spans end-to-end encryption, data loss prevention (DLP) measures, robust key management systems, and emerging techniques like post-quantum cryptography. The CSF will deliver classifications of all data protection mechanisms across partners, establishing common standards for encryption and data handling throughout the platform. By mapping partner contributions (e.g. encryption services or key vaults) to this domain, the framework ensures consistent data confidentiality and integrity measures are applied project-wide. It also helps in verifying that each partner's solutions meet compliance requirements for data privacy and that any critical gaps (such as missing encryption capabilities) are addressed collaboratively.

3.2. At-rest and in-transit Encryption

"At-rest" and "in-transit" encryption are two complementary classes of data protection methodologies that ensure confidentiality. Encryption "at rest" protects the data while it is stored on a persistent medium, its goal is to ensure that even if somebody gains unauthorized access to the storage (e.g., a stolen disk, leaked database, or compromised backup), the data remains unreadable without the appropriate cryptographic keys. Conversely, in-transit encryption protects the data while they are being transmitted between two or more parties. Its goal is to avoid unauthorized access to information sent through a misconfigured, or attacked (subject to Man-in-the-Middle) transmission.

3.2.1 At-rest encryption

to be added in a future version of the document

3.2.2 In-transit Encryption

TLS (Transport Layer Security) is a collection of highly customizable cryptographic protocols which became the de facto standard for securing internet communications. They were initially introduced in 1999 to ensure the privacy of communications over the internet. In the past 26 years, TLS has been modified and refined to enhance its efficiency and eliminate a wide range of potential attacks by disabling or rewriting certain features. This has resulted in the production of three subsequent versions, the most recent of which (TLS 1.3) was published in 2018.

The level of customizability that TLS provides has a significant drawback: it delegates the responsibility for secure configuration to the system administrator, who is not necessarily required to possess a comprehensive understanding of network security, attacks against cipher suites, and malicious use of TLS extensions. Because of this, numerous cybersecurity agencies worldwide have begun the issuance of technical guidelines to guide system administrators in achieving a secure configuration and to establish a national security threshold.

The security of the TLS protocol depends on many configurable elements that a system administrator has to set. These elements range from the protocol's version to the allowed cryptographic algorithms, which have to be chosen from a list of more than one hundred elements, many of which are not secure. The documents aid the sysadmins in choosing the right elements by specifying a requirement for every configurable element of a webserver by taking into account published standards, deprecated features, and known attacks.

More technical details in Annex.

3.3 End-to-End Encryption

to be added in a future version of the document

3.4 Data loss prevention

to be added in a future version of the document

3.5 Key management systems

to be added in a future version of the document

3.6 Post-quantum Cryptography

In the traditional federated learning setting, the models to be trained are specified by the clients. The server “blindly” aggregates the model updates received from clients in each training round and redistributes the aggregated model. In the second scenario, the server specifies the model to be trained, and the clients blindly train this prescribed model on their local data. In this case, the final trained model, including its hyper-parameters and architecture, may be shared with clients only with explicit permission from the server.

These two settings impose different security requirements, but they share a common objective: preventing the disclosure of a client’s training data to the server or to other clients. Achieving this requires encrypting model updates, since a variety of attacks can infer sensitive information about a client’s training data solely from the shared models or model updates.

In Federated Averaging (FedAvg), for example, each client sends an encrypted, weighted model update together with the corresponding encrypted weight to the server. The server then computes the encrypted sum of the weighted model updates and the encrypted sum of the weights separately. At the final step the server or client - after decryption - divides the former by the latter to obtain the weighted average of the model updates. This step depends on how much the server is trusted and who possesses the decryption key: the server or the client(s). Both scenarios can be supported.

A “holy-grail” solution would be the use of Fully Homomorphic Encryption (FHE), which would prevent both the server and the clients from accessing the aggregated model while still enabling

training on private data and, potentially, a private model. However, FHE is currently feasible only for special, typically small models and remains computationally expensive even in those cases.

A vision is to rely on additive homomorphic encryption (or secure aggregation) schemes to hide model updates from the aggregator in both scenarios. In this approach, clients perform full-fledged local model training, while the server is restricted to summing encrypted model updates. In the second scenario, additional protection mechanisms, such as carefully designed access management or vendor-specific Trusted Execution Environments (TEEs), can be used to prevent a client's unauthorized access to the model or its architecture when required.

Importantly, the objective is not only to protect individual clients' model updates, but also to conceal intermediate iterations of the jointly trained model (i.e. intermediate model snapshots). Indeed, traditional secure aggregation schemes typically allow access to the aggregated model after each federated round. In contrast, our goal is to disclose only the final model at the end of the last federated round. This is critical because intermediate models can be exploited to reconstruct clients' training data, effectively negating the privacy guarantees provided by homomorphic encryption or secure aggregation.

Beyond analyzing quantum-safe (e.g., lattice-based) homomorphic encryption schemes, an important research question is whether and how these techniques can be combined with secure aggregation, and for which types of model training fully homomorphic encryption (FHE) is practically realizable. In addition, it is also important to investigate which robust federated aggregation schemes beyond FedAvg can be implemented using secure multi-party computation.

Topic 4: Application Security

4.0 General Description

This topic addresses the security of software applications and services, including their development and runtime protection. It involves secure coding practices, code analysis (static and dynamic testing), application firewalls like WAF, API security, and the use of Software Bill of Materials (SBOM) to track components. The CSF will map and classify all application security tools and practices contributed by partners for example: cataloging code scanning tools or runtime protection modules to ensure they align with the reference architecture's application layer. By doing so, the CSF facilitates a baseline for secure software development and deployment across the project. Partners' contributions are integrated so that vulnerabilities are identified early, common security standards like OWASP Top 10 mitigation are followed, and all applications in the IPCEI-CIS ecosystem benefit from a robust and consistent security posture.

4.1 Secure Coding Practices

Secure coding practices encompass a set of principles and techniques that developers employ to prevent vulnerabilities in software applications. These practices include input validation, proper authentication and authorization mechanisms, secure data handling, and defensive programming techniques. The stakes are remarkably high: a single overlooked SQL injection vulnerability or an

improperly configured authentication system can expose millions of users' personal data, result in financial losses, or even endanger lives in critical systems.

Input Validation and Sanitization – Implement comprehensive validation for all user inputs, including type checking, length restrictions, and format verification. Sanitize data before processing to prevent injection attacks such as SQL injection, XSS, and command injection.

Principle of Least Privilege – Grant applications and services only the minimum permissions necessary to perform their functions. Avoid running processes with administrative or root privileges unless absolutely required, and implement RBAC.

Secure Authentication and Session Management – Use strong password policies, Multi-Factor Authentication, and secure session handling. Implement proper timeout mechanisms, secure cookie flags (HttpOnly, Secure, SameSite), and protect against session fixation and hijacking.

Error Handling and Logging – Implement comprehensive error handling that prevents exposure of sensitive information in error messages. Maintain detailed security logs for audit trails while ensuring logs don't contain passwords or sensitive data.

Secrets Management – Never hardcode credentials, API keys, or cryptographic keys in source code. Use dedicated secrets management solutions or cloud provider services, and rotate secrets regularly following security policies.

4.2 Static and dynamic analysis

Static and dynamic analysis represent two complementary pillars of software quality assurance that together provide comprehensive insight into application security and reliability. Static analysis examines source code, bytecode, or binaries without executing the program, identifying potential vulnerabilities, coding standard violations, and logic errors before the software ever runs. This approach offers significant advantages: it can analyze all possible execution paths, detect issues early in the development cycle when they are least expensive to fix, and integrate seamlessly into continuous integration pipelines. Tools like linters, security scanners, and formal verification systems can automatically flag common vulnerabilities such as buffer overflows, SQL injection points, or memory leaks. By catching these issues during development rather than production, static analysis dramatically reduces both security risks and maintenance costs while enforcing consistent coding standards across development teams.

Static Application Security Testing (SAST) – Integrate automated code analysis tools into the development pipeline to identify vulnerabilities in source code without executing the program. Tools should scan for common weaknesses like buffer overflows, race conditions, and insecure cryptographic implementations.

Dynamic Application Security Testing (DAST) – Perform runtime testing of applications in their deployed environment to identify vulnerabilities that only manifest during execution. This includes testing for authentication flaws, configuration errors, and runtime injection vulnerabilities.

Software Composition Analysis (SCA) – Scan third-party libraries and dependencies for known vulnerabilities using CVE databases. Automate the identification of outdated components and receive alerts for newly discovered vulnerabilities in the dependency chain.

Interactive Application Security Testing (IAST) – Deploy agent-based testing that combines SAST and DAST approaches, analyzing code behavior during automated testing or QA processes. This provides real-time vulnerability detection with lower false-positive rates.

Continuous Integration Security Gates – Establish automated security checkpoints in Continuous Integration and Continuous Deployment (CI/CD) pipelines that prevent code with critical vulnerabilities from progressing to production. Define clear security policies with severity thresholds and require remediation before deployment approval.

4.3 Application Firewall

Web Application Firewalls (WAF) have become an indispensable component of modern cybersecurity infrastructure, serving as a critical shield between web applications and the constant barrage of malicious traffic on the internet. Unlike traditional network firewalls that operate at the network layer, WAFs specifically protect web applications by filtering, monitoring, and blocking HTTP/HTTPS traffic based on predefined security rules. They defend against common attack vectors such as SQL injection, Cross-Site Scripting (XSS), cross-site request forgery (CSRF), and zero-day exploits that target application-layer vulnerabilities. In today's threat landscape, where automated bots continuously scan for weaknesses and attackers exploit vulnerabilities within hours of their disclosure, a WAF provides essential real-time protection. It acts as a virtual security guard, examining every request to the application and blocking suspicious patterns before they can reach vulnerable code, effectively buying organizations precious time to patch vulnerabilities while maintaining service availability.

Rule-Based Traffic Filtering – Configure WAF rules to block common attack patterns including SQL injection, XSS, and remote file inclusion. Maintain both signature-based detection for known threats and anomaly-based detection for zero-day attacks.

Geographic and IP Reputation Filtering – Implement geo-blocking to restrict access from high-risk regions and leverage threat intelligence feeds to block traffic from known malicious IP addresses, botnets, and proxy services.

Rate Limiting and DDoS Protection – Configure request rate limits per IP address, session, or user to prevent brute force attacks and application-layer DDoS attempts. Implement progressive delays or CAPTCHA challenges for suspicious traffic patterns.

Virtual Patching – Deploy WAF rules to protect against newly discovered vulnerabilities while patches are being developed and tested. This provides immediate protection for applications that cannot be immediately updated due to business continuity requirements.

Logging and Monitoring – Enable comprehensive logging of blocked and allowed requests with detailed attack signatures. Integrate WAF logs with Security Information and Event Management (SIEM) systems for correlation analysis and establish alerts for attack pattern escalations or policy violations.

4.4 API Security

APIs have become the backbone of modern digital ecosystems, enabling seamless communication between applications, services, and platforms. From mobile banking apps to social media integrations and cloud services, APIs facilitate the data exchange that powers contemporary business operations. However, this central role also makes APIs an attractive target for cybercriminals. Unlike traditional web applications with user interfaces, APIs often handle sensitive data directly and programmatically, processing authentication tokens, personal information, and business-critical transactions at scale. A compromised API can expose vast amounts of data in a single breach, as evidenced by numerous high-profile incidents where attackers exploited weak API authentication, excessive data exposure, or inadequate rate limiting. The challenge is compounded by the sheer volume and complexity of modern API ecosystems – organizations often manage hundreds or thousands of APIs, many of which may be poorly documented, inadequately secured, or even forgotten entirely, creating what security professionals call "shadow APIs" that represent unknown vulnerabilities.

Authentication and Authorization – Implement robust API authentication mechanisms such as OAuth 2.0, JWT tokens, or API keys with proper validation. Enforce fine-grained authorization checks to ensure users can only access resources they are permitted to use.

Rate Limiting and Throttling – Protect APIs against abuse and denial-of-service attacks by implementing request quotas per user or API key. Use sliding window algorithms to enforce fair usage policies and prevent resource exhaustion.

Input Validation and Schema Enforcement – Validate all API inputs against defined schemas using specifications like OpenAPI/Swagger. Reject requests that don't conform to expected data types, formats, and business logic rules to prevent injection attacks.

Encryption in-transit and at Rest – Enforce TLS 1.2 or higher for all API communications with strong cipher suites. Use proper certificate validation and implement certificate pinning for mobile applications to prevent man-in-the-middle attacks.

API Gateway and Versioning – Deploy API gateways to centralize security controls, monitoring, and traffic management. Implement clear versioning strategies to maintain backward compatibility while deprecating insecure legacy endpoints in a controlled manner.

4.5 Software Bill of Materials

Modern software development relies heavily on third-party components, open-source libraries, and dependencies that can number in the hundreds or even thousands for a single application. While

this approach accelerates development and reduces costs, it also introduces significant security and compliance risks that often remain invisible to organizations. A Software Bill of Materials (SBOM) addresses this blind spot by providing a comprehensive, structured inventory of all components within a software product, including their versions, licenses, and dependencies. This transparency is crucial because vulnerabilities in widely-used components like Log4j or OpenSSL can affect countless applications simultaneously, yet organizations cannot protect what they don't know they are using. When critical vulnerabilities emerge, an accurate SBOM enables teams to quickly identify affected systems and prioritize remediation efforts, transforming what might be weeks of manual investigation into hours of targeted response. Beyond security, SBOMs facilitate license compliance, helping organizations avoid legal risks associated with improperly used open-source components and ensuring adherence to licensing obligations that could otherwise result in costly litigation or forced code disclosure.

Comprehensive Component Inventory – Generate detailed SBOMs that catalog all software components, libraries, frameworks, and dependencies including version numbers, licenses, and origin sources. Use standardized formats like SPDX or CycloneDX for interoperability.

Automated Vulnerability Tracking – Correlate SBOM data with vulnerability databases (NVD, GitHub Security Advisories) to identify affected components when new CVEs are published. Establish automated alerting systems to notify teams immediately when dependencies are compromised.

License Compliance Management – Track open-source licenses across all components to ensure compliance with legal obligations and avoid incompatible license combinations. Identify components with restrictive licenses that may conflict with proprietary software distribution.

Supply Chain Transparency – Document the provenance and integrity of software components throughout the development and deployment lifecycle. Verify cryptographic signatures and checksums to detect tampering or unauthorized modifications in the supply chain.

SBOM Integration in CI/CD – Automatically generate and update SBOMs during build processes and store them alongside deployable artifacts. Make SBOMs available to security teams, compliance officers, and customers for transparency and incident response purposes.

More technical details in Annex.

Topic 5: Endpoint & Device Security

5.0 General Description

In the 8ra Cloud-Edge Continuum, the concept of a defensive perimeter has dissolved. Security can no longer rely on network boundaries or centralized choke points. Instead, trust must be anchored in the most fundamental element of the infrastructure: the physical device itself. Endpoint & Device Security serves as the foundational "Anchor of Trust" for the entire ecosystem, ensuring that every node – from the smallest industrial sensor to the most powerful far-edge server – operates in a verifiable, tamper-resistant state.

This topic implements a strict Zero Trust Architecture at the hardware level. The core premise is that software security is fundamentally strictly dependent on hardware integrity; if the underlying platform is compromised, no higher-level control – whether IAM, Network Security, or Application Logic – can be trusted. Therefore, we move from a model of assumed trust to one of cryptographic proof.

Strategic Objectives

Digital Sovereignty through Hardware Roots of Trust: By mandating immutable hardware anchors (TPM 2.0), we ensure that device identity and integrity are bound to silicon, preventing spoofing and unauthorized cloning.

Compliance by Design: The architectural requirements defined herein are explicitly mapped to the EU Cybersecurity Certification Scheme for Cloud Services (EUCS) at the High assurance level (CS-EL4) and the Cyber Resilience Act (CRA), providing a direct technical path to regulatory compliance.

Resilience in Hostile Environments: Acknowledging that edge devices often operate in physically uncontrolled environments, this framework enforces rigorous standards for tamper resistance and environmental hardening.

By establishing this hardware-backed root of trust, the Endpoint & Device Security domain provides the "verifiable evidence" required by other domains (SecOps, GRC) to authorize access and validate compliance dynamically.

5.1 Device Trust, Platform Integrity & Attestation

Establishing a verifiable chain of trust, anchored in immutable hardware, is the foundational requirement for ensuring platform integrity across the 8ra Cloud-Edge Continuum. The ability to computationally prove the trustworthy state of any device – from a sensor in an industrial setting to a core cloud server – is a cornerstone of a secure and resilient digital infrastructure. This objective directly supports the strategic goal of European Digital Sovereignty by making the trustworthiness of any participating device verifiable, independent of its physical location or network, thereby fostering a reliable and defensible digital ecosystem.

The core security goal for this domain is to ensure that every device boots and operates in a known, uncompromised state. This shall be achieved by cryptographically measuring all boot and runtime software components, binding the device's unique identity to its hardware, and enabling a trusted remote verifier to appraise the device's integrity state before granting it access to the continuum's resources. This moves beyond legacy trust assumptions and establishes a model of explicit, evidence-based trust.

To realize this vision, a set of specific architectural mandates must be adopted and enforced throughout the 8ra ecosystem, defining the non-negotiable technical baseline for platform security.

More technical details regarding architecture, implementation, etc. in Annex.

5.2 Device Lifecycle & Configuration Management

Device Lifecycle Management (DLM) and Configuration Management (CM) are foundational pillars for establishing and maintaining trust throughout the 8ra Multi-Provider Cloud-Edge Continuum. These processes are not mere administrative overhead; they are critical security functions that ensure every device – from edge sensors to core infrastructure – is onboarded, operated, and retired in a verifiably secure state. This framework adopts the formal practice of Configuration Management, encompassing configuration identification, change control, status accounting, and verification audits to maintain a trusted state. By enforcing a rigorous and consistent approach to device integrity, these disciplines directly uphold the core IPCEI-CIS principles of Digital Sovereignty and Zero Trust, guaranteeing that the digital foundation of Europe's cloud-edge infrastructure is both resilient and trustworthy from the silicon up.

The primary security objective of this sub-topic is to enforce a consistent, auditable, and secure posture for all endpoint devices throughout their entire operational lifecycle. This goal encompasses initial secure onboarding, continuous configuration hardening and monitoring, and cryptographically secure decommissioning. Successfully achieving this objective minimizes the collective attack surface of the ecosystem, prevents unauthorized modifications, and ensures continuous compliance with the stringent requirements for EUCS High Assurance.

More technical details regarding architecture, implementation, etc. in Annex.

5.3 Updates, Vulnerability & Software Supply Chain Security

In a distributed Cloud-Edge Continuum, the integrity of every device's software stack – from the immutable boot ROM to the final application – is the foundation of digital sovereignty. This foundational layer represents a primary target for sophisticated adversaries seeking to establish persistent, undetectable control over digital infrastructure. A compromise at this level subverts all higher-level security controls, rendering even the most robust operating system and application-level defenses inadequate. This chapter defines the mandatory architectural principles for ensuring the provenance, integrity, and resilience of all software and firmware throughout its entire lifecycle. These principles directly align with the high-assurance requirements of the EUCS and are a prerequisite for implementing a credible Zero Trust Architecture.

5.4 Local Data Protection & Key Handling on Devices

Local data protection serves as a foundational pillar for achieving digital sovereignty within the 8ra Cloud-Edge Continuum. The security and integrity of the entire ecosystem are predicated on the confidentiality and resilience of data at its most distributed and vulnerable point: the edge device. This section defines the mandatory, non-negotiable security primitives required to protect both data-at-rest and data-in-use on these devices. These requirements ensure that every device is fortified against a spectrum of threats, ranging from remote software attacks to sophisticated physical tampering. Adherence to these principles is essential for meeting the high assurance levels mandated by the EUCS and is a critical step toward building a resilient and trusted European digital infrastructure.

More technical details regarding architecture, implementation, etc. in Annex.

5.5 Endpoint Telemetry, Health Monitoring & Host Security Controls

This sub-topic represents the cornerstone of operational visibility and resilience within the IPCEI-CIS. The controls defined herein provide the continuous, verifiable evidence required to establish and enforce a Zero Trust Architecture across the entire Cloud-Edge Continuum. The continuous monitoring of endpoint health, integrity, and configuration is not merely a best practice; it is a foundational requirement for achieving the EUCS High assurance level. This transforms security from a static, point-in-time assessment into a dynamic, continuously validated process, ensuring that every device accessing the ecosystem is trustworthy.

The primary security objective for this sub-topic is to establish and maintain a continuous, verifiable, and resilient security posture for all endpoints participating in the IPCEI-CIS. This is achieved by ensuring every device – from cloud servers to edge nodes – operates in a known-good, hardened state, that all security-relevant activity is comprehensively logged for analysis, and that any deviation from the established security baseline is detected and remediated in near real-time. This approach stands in stark contrast to the outdated "castle-and-moat" security model, which is ineffective for decentralized edge environments where the perimeter is dissolved and trust cannot be assumed based on network location alone. The principles outlined here are fundamental to a "never trust, always verify" Zero Trust model.

More technical details regarding architecture, implementation, etc. in Annex.

5.6 Physical & Environmental Protection for Edge/Far-Edge Devices

The strategic importance of physical and environmental protection for edge and far-edge devices within the 8ra Cloud-Edge Continuum cannot be overstated. Unlike centralized cloud infrastructure housed in secure data centers, these devices often operate in unsupervised, semi-public, or environmentally harsh locations, such as factory floors, cell towers, and public infrastructure. This exposure makes them primary targets for physical tampering, theft, sabotage, and environmental damage, which can undermine the integrity and availability of the entire cloud-edge ecosystem. Failure to secure these distributed assets provides a direct vector for adversaries to compromise critical Operational Technology (OT) and escalate attacks into the core cloud infrastructure, posing a direct threat to the safety and operational continuity of EU critical sectors.

The objective of this chapter is to establish a set of mandatory controls that ensure the trustworthiness, integrity, and operational availability of these critical assets. By defining a robust security posture at the physical layer, we align directly with the European Union's strategic goals for digital sovereignty and critical infrastructure resilience, as outlined in the Directive on the resilience of critical entities (CER Directive). This section will detail the specific architectural requirements, standardization baselines, and implementation strategies necessary to achieve this objective, forming a foundational layer of trust for the entire cloud-edge ecosystem.

More technical details regarding architecture, implementation, etc. in Annex.

5.7 Gap Analysis & Future Roadmap

To maintain the 8ra vision of a sovereign, future-proof continuum, the current building blocks must evolve to address emerging threats and constraints.

5.7.1 Post-Quantum Cryptography (PQC) Readiness

Gap: Current TPMs and Secure Boot flows rely heavily on RSA and ECC, which are vulnerable to Quantum threats.

Roadmap: Transition to LMS/XMSS (stateful hash-based signatures) for firmware signing as an interim step. Adopt Dilithium or Kyber algorithms for key exchange and signing once standardized in TPM 2.0 specifications (or TCG 2.0+ revisions). Implement "Crypto-Agility" in update pipelines to support swapping algorithms without hardware replacement.

5.7.2 Heterogeneous & Constrained Edge

Gap: Full TPM 2.0 implementation is challenging on ultra-low-power IoT sensors (MCUs) and legacy brownfield devices.

Roadmap: Standardize on Device Identifier Composition Engine (DICE) architectures for constrained devices, providing a lighter-weight root of trust compatible with the IPCEI-CIS verification framework. Develop "Legacy Proxy" patterns where secure gateways provide attestation on behalf of non-compliant sensors.

Topic 6: Security Operations (SecOps)

6.0 General Description

This topic covers the operational aspects of security, including continuous monitoring, threat detection, incident response, and security assurance activities. It involves SIEM systems for centralized logging, alerting and analytics, automated threat response or Security Orchestration, Automation, and Response (SOAR) playbooks, and regular exercises (red team/blue team) to test defenses. Through the CSF, partners' SecOps tools and services will be identified and integrated into a unified operational security framework. The CSF will deliver a coordinated approach to monitoring and incident handling. Examples would be ensuring that logs and alerts from all components feed into a common analysis platform and establishing joint incident response procedures. By mapping each partner's monitoring and response capabilities, the framework improves overall situational awareness and ensures that security events anywhere in the continuum can be detected and addressed quickly through collaborative efforts.

6.1 Continuous monitoring and logging

to be added in a future version of the document

6.2 SIEM systems

to be added in a future version of the document

6.3 Threat detection and response

to be added in a future version of the document

6.4 Automated incident response

to be added in a future version of the document

6.5 Red team/blue team exercises

to be added in a future version of the document

Topic 7: Governance, Risk & Compliance

7.0 General Description

The CSF will establish a common reference for security certification schemes and compliance within the European regulatory landscape. Its primary objective is to define principles, requirements, and guidelines that ensure a consistent, transparent, and robust approach to cybersecurity certification and compliance across multiple domains. It addresses key initiatives such as the European Cybersecurity Certification for ICT products, the EUCS, the CRA, and the Cloud Sovereignty Framework. The document seeks to:

- **Promote Harmonization:** Align certification practices and security standards across Member States to facilitate mutual recognition and interoperability.
- **Ensure Trust and Compliance:** Provide assurance that certified products, services, and infrastructures meet stringent cybersecurity requirements, supporting regulatory compliance and resilience against evolving threats.
- **Support Risk-Based Approaches:** Define certification levels and evaluation criteria proportionate to the risk and criticality of the asset or service.
- **Enable Transparency and Accountability:** Establish clear roles, responsibilities, and processes for certification bodies, manufacturers, and service providers.
- **Foster Digital Sovereignty:** Encourage frameworks that safeguard data protection, privacy, and control over critical infrastructures, particularly in cloud environments.

7.1 Securing IT Components in the EU – Legal Framework and Certification Schemes

Europe's cybersecurity architecture blends a horizontal product safety law (the CRA), EU level certification schemes (EUCS for cloud services and EUCC for ICT products), and an operational risk management directive (NIS2). In parallel, a Cloud Sovereignty Framework, not a single EU statute but a set of principles emerging from national schemes and EU debates, guides how cloud services should protect data from extra EU jurisdiction, ensure operational control, and meet stringent

security baselines. Together, these instruments help organizations design secure products, buy trustworthy cloud services, and evidence compliance to regulators and customers.

7.2 Cyber Resilience Act – Security by Design for Digital Products

The CRA introduces essential cybersecurity requirements for all "products with digital elements" placed on the EU market (secure design, vulnerability handling, updates, and transparency). It provides conformity assessment routes and, crucially, allows EU cybersecurity certification to serve as a presumption of conformity when the certificate attains at least Substantial assurance (e.g., under EUCC). ENISA has published a mapping showing how EUCC can be used "seamlessly" to demonstrate CRA compliance, reducing audit duplication for manufacturers.

Implication: If you build cryptographic hardware, network equipment, or embedded software, a single EUCC evaluation can anchor CRA compliance and provide third-party assurance for buyers and regulators.

7.3 EUCC – EU Common Criteria-based Certification for ICT Products

The EU Cybersecurity Certification Scheme on Common Criteria (EUCC) is the EU's first operational scheme under the Cybersecurity Act. It is voluntary by design, applies to ICT products (hardware, software, components), and issues certificates at two assurance levels: Substantial and High. EUCC blends classical Common Criteria with modern lifecycle obligations, vulnerability monitoring and patch management, so certificates remain meaningful over time. Since February 2025, Member States have started issuing EUCC certificates, which ENISA publishes on the EU portal.

Scope examples: secure ICs and smart card platforms, HSMs, digital tachograph components, routers, firewalls, and other ICT products; the scheme's state of the art documents and cryptography guidelines align with agreed mechanisms used in Europe's CC ecosystem.

7.4 EUCS – EU Cloud Services Certification: Purpose and Status

The EUCS scheme aims to certify cloud services (IaaS/PaaS/SaaS) at graded assurance levels, harmonizing fragmented national requirements, and providing an EU recognized baseline for cloud security. As of 2025, EUCS remains under political discussion at EU level; progress has been slowed by debates about sovereignty/data immunity conditions for the highest tier. Nonetheless, ENISA's draft materials define scope and intent, and buyers already prepare for EUCS style requirements.

Interim practice: Several Member States use national frameworks for sensitive workloads, notably France's SecNumCloud qualification, which couples technical controls with juridical immunity from extra EU laws and strict operational control. These national schemes often act as proxies until EUCS is finalized.

7.5 Cloud Sovereignty Framework – Principles Shaping Trustworthy Cloud

The Cloud Sovereignty Framework is best understood as a set of practical principles, data residency, immunity from non-EU jurisdiction, EU controlled operations, and robust security baselines, drawn

from national schemes (e.g., SecNumCloud), vendor implementations, and the ongoing EUCS debate. The sovereignty discussion is a central reason EUCS adoption has been politically complex, with some Member States pushing for stronger immunity and localization requirements at the high assurance levels.

Note: Sovereignty criteria are not yet standardized at EU level. Where public bodies or regulated sectors demand them, they currently rely on national schemes (e.g., SecNumCloud) or bespoke contractual controls until EUCS is adopted

7.6 NIS2 – Operational Risk Management for Critical Sectors

The NIS2 Directive strengthens cybersecurity across essential and important entities in critical sectors, digital infrastructure (including cloud services and data centers), energy, transport, banking/financial market infrastructures, health, public administration, and more. It mandates governance accountability, risk management, supply chain security, incident reporting, and continuous improvement, with national competent authorities supervising enforcement after transposition.

Key point: NIS2 is outcome-focused; it does not impose a single certificate. Instead, it expects entities to prove appropriate controls and supplier diligence; certifications and sovereignty controls become powerful evidence to satisfy these obligations.

7.7 How EUCC, EUCS, and the Cloud Sovereignty Framework Support NIS2

Supplier Due Diligence & Procurement Evidence

Under NIS2, entities must manage ICT and cloud supplier risk. Choosing EUCC certified products (e.g., HSMs, secure ICs, network equipment) gives third party assurance of product security, vulnerability handling, and patching, reducing diligence burden and strengthening audit evidence. Legal analyses note EUCC's value for demonstrating compliance under NIS2, DORA, and CRA, even as a voluntary scheme.

Cloud Assurance and Sovereignty Controls

When adopted, EUCS will provide EU recognized assurance levels for cloud services, helping entities meet NIS2 expectations on provider selection, contractual clauses, and continual oversight. Until then, a Cloud Sovereignty Framework based on SecNumCloud style controls (EU residency, immunity from extra EU law, EU controlled operations) can be used as evidence that sensitive workloads meet stringent NIS2 supply chain and risk management requirements, especially in public sector or regulated contexts.

Continuous Improvement and Vulnerability Management

NIS2 emphasizes ongoing risk management and incident response. EUCC certificates include surveillance obligations, vulnerability monitoring and patch management, creating an evidence trail

aligned with NIS2's continuous improvement ethos. This linkage simplifies audits and remediation after advisories/CVEs.

Synergy with CRA (Product Law)

Where NIS2 entities also place products on the market, EUCC can anchor CRA conformity (via presumption of conformity) and support NIS2 supplier assurance, joining product safety with operational resilience in one recognized framework. This is particularly useful for cryptographic hardware or edge devices deployed in critical infrastructures.

7.8 Putting It Together: A Practical View

Design & Build (CRA + EUCC): Use EUCC to meet CRA essential requirements and deliver trustworthy ICT products with ongoing vulnerability management.

Buy & Operate (EUCS + Sovereignty + NIS2): For cloud workloads, prepare for EUCS; in the interim, apply a Cloud Sovereignty Framework (e.g., SecNumCloud controls) to satisfy NIS2's supplier risk expectations for sensitive data and services.

Demonstrate Compliance (NIS2): Document certifications (EUCC today; EUCS when available), sovereignty measures, and continuous security processes as evidence for NIS2 audits and supervision

Bottom line:

+ EUCC (today) and EUCS (tomorrow) are assurance tools that make NIS2 provable.
+ The Cloud Sovereignty Framework bridges current policy gaps for cloud assurance, pending EUCS adoption.
+ The CRA aligns product-level security by design with these assurance paths, creating a coherent route from secure products to resilient operations under NIS2.

7.9 Legal consequences of NIS2 non-compliance for a cloud provider

Under NIS2, cloud computing service providers are treated as part of the digital infrastructure / digital providers family and are covered by a Commission Implementing Regulation[1] that sets technical and methodological requirements for these entities EU wide. In other words, if you offer IaaS/PaaS/SaaS into the EU market, you are presumptively regulated.

Supervisory Exposure and corrective Measures

If you are classified as an essential entity (many cloud providers are), national authorities can apply proactive supervision. They may perform on-site inspections and off-site supervision, run regular, targeted or ad hoc security audits (with audit costs typically borne by the provider), conduct security scans, and compel access to data, documents, and evidence of control implementation. Art. 32

If you are treated as an important entity, authorities act ex-post (after indications of non-compliance), but they still hold powers to conduct inspections, targeted audits at your expense, security scans, and broad information/evidence requests.

NIS2 also makes clear that supervisory authorities must be able to operate effectively and independently and should coordinate with GDPR regulators where incidents entail personal data breaches, so a single failure can trigger multi-regulator scrutiny.

Administrative Fines (harmonized EU ceilings)

Failure to implement risk management measures (Art. 21) or to meet incident reporting duties (Art. 23) can lead to significant administrative fines (Art. 34):

- **For essential entities:** up to at least €10 million or 2% of worldwide annual turnover (whichever is higher).
- **For important entities:** up to at least €7 million or 1.4% of worldwide annual turnover (whichever is higher).
- These fines are in addition to other supervisory measures (e.g., audits, inspections, information demands) that authorities can impose.

[1] Commission implementing Regulation C(2024) 7151 https://digital-strategy.ec.europa.eu/en/library/nis2-commission-implementing-regulation-critical-entities-and-networks, https://nis2directive.eu/

Incident Reporting Failures (24 h / 72 h / 1 month)

NIS2 requires a 24-hour early warning, a 72-hour incident notification, and a final report within one month for any significant incident. Missing these deadlines can itself be a breach that exposes you to the fine regime above (because it violates Art. 23).

Board Level Liability and Governance

NIS2 elevates cybersecurity to the management body: boards must approve the risk management measures required by Art. 21, oversee their implementation, and can be held liable for infringements. Directors must also receive regular cybersecurity training; regulators can test that this is happening.

Procurement, Certification, and Market Access Risks

Member States and, where needed, the European Commission may require the use of EU cybersecurity certification schemes (e.g., for ICT products or services in your stack) to demonstrate compliance with Art. 21. If you lack the required assurance, you can be barred from tenders or customer contracts that reference these schemes.

Compounded Exposure with other Regime

For cloud providers, non-conformity that results in a personal data breach will usually trigger parallel GDPR enforcement (coordinated via NIS2 Art. 31(3)), increasing litigation and remediation exposure (regulatory, civil, and contractual).

What this means in Practice for Cloud Providers

Expect audits at your cost and intrusive evidence requests. Authorities can require targeted audits by independent bodies and demand the underlying evidence, not just policy statements (Art. 32). Budget and plan for this.

Treat the 24 h/72 h/1 month reporting clock as non-negotiable. Build playbooks and evidence packages that let you file within the windows – even amid uncertainty.

Prepare the board. Minutes and approvals for NIS2 risk measures should be explicit, train directors and record attendance and content.

Harden your supply chain posture. Supervisors will examine the quality of suppliers and secure development practices across your stack; weak links can drive findings and fines.

Align with EU certification where relevant. Procurement teams increasingly require EU level certification to evidence Art. 21 controls; anticipate this to avoid commercial exclusion.

Bottom line: For a cloud provider, NIS2 non-compliance isn't just a technical issue it's a legal, financial, and governance exposure that can combine audits, binding corrective steps, and fines up

to at least €10 million/2% (essential) or €7 million/1.4% (important), plus knock on GDPR risk where personal data is involved. The most effective mitigation is to operationalize Art. 21 controls, industrialize incident reporting, and equip the board to provide and prove oversight.

Topic 8: Resilience & Availability

8.0 General Description

This topic focuses on keeping services reliable and available even under adverse conditions. Key aspects include DDoS mitigation strategies, redundancy and failover mechanisms, disaster recovery planning, and even techniques like chaos engineering to test system robustness. The CSF will integrate and classify all measures related to resilience contributed by partners, delivering a comprehensive continuity strategy for the project. This involves mapping out how each partner's components achieve high availability, for example, identifying which services have built-in failover or backup, and how they interconnect across providers. By coordinating these strategies, the CSF ensures that critical cloud-edge services maintain agreed Service Level Agreements for uptime and that recovery procedures are in place. The framework will highlight any weaknesses in continuity such as single points of failure and facilitate joint improvements, thereby enhancing the overall reliability of the multi-provider environment.

8.1 DDoS Mitigation

DDoS and large-scale network attacks remain among the most visible threats to service availability, growing in both scope and sophistication. AI driven botnets, and amplification vectors capable of exceeding terabit scale traffic volumes. These attacks have the potential to disrupt critical European digital services and cloud-edge operations. NBIP, as a key partner in this domain, will deliver a federated detection and mitigation capability that strengthens resilience against such events. By embedding DDoS defense mechanisms into the CSF framework, partners will benefit from collective situational awareness and coordinated mitigation measures, ensuring that service availability remains protected even under extreme stress.

8.2 Redundancy and Failover

Eliminating single points of failure through redundancy and automatic failover is fundamental to high availability. A resilient architecture is built on duplicate components and seamless failover processes: if one component fails, another instantly takes over to maintain continuity. Partners will contribute solutions which support technologies such as clustered deployments, multi-zone or multi-region distribution, and active-active load balanced services to the framework. The CSF will map these approaches for instance, identifying which critical services use active-passive versus active-active clustering to ensure each workflow has appropriate backup instances ready. By harmonizing failover mechanisms across the Multi-Provider Cloud-Edge Continuum, the framework ensures that localized outages or hardware failures do not cascade into overall service downtime. This coordinated approach means that even in the face of component failures, the system "fails over" gracefully, preserving uptime with minimal disruption to users.

8.3 Disaster Recovery Planning

Even with robust real-time failover, organizations must prepare for wider disasters that could knock out multiple components or entire sites. Disaster recovery (DR) planning focuses on restoring services after catastrophic events, within agreed recovery objectives. This includes setting clear Recovery Time Objectives (RTOs) for how quickly services must be restored and Recovery Point Objectives (RPOs) for how much data loss is acceptable. Key partner contributions in this area include offsite data backups, cross-region replication, and standby infrastructure that can be activated in emergencies. The CSF will unify these measures into a cohesive DR strategy, ensuring that each critical component has a documented recovery procedure and backup environment. Regular backups and geo-redundant data storage are essential to protect against data loss. The framework will highlight any single points of failure in recovery plans and mandate regular drills or simulations to test restoration procedures, so that when a real disruption strikes, all participants can rapidly bring services back online. By coordinating DR planning across the consortium, the CSF ensures that even a large-scale failure can be overcome with minimal downtime and aligned processes for continuity.

8.4 Chaos Engineering

Resilience is not just planned but also proactively tested. Chaos engineering has emerged as a practice of deliberately introducing failures into systems to validate their robustness. In the context of the CSF, partners will adopt similar chaos testing techniques to continually probe the limits of their services. This may involve simulating node outages, network latency spikes, or other failure modes in a controlled manner. By doing so, hidden weaknesses such as a dependency on a component that wasn't obvious can be exposed and fixed before they cause real incidents. The CSF will incorporate guidelines and tools for chaos engineering across the multi-provider environment, building confidence that the combined infrastructure can self-heal and meet its availability targets even when things go wrong. In essence, "breaking things on purpose" under controlled conditions leads to stronger systems that are far less likely to break unexpectedly in production.

In a federated cloud-edge ecosystem, no single provider's measures are sufficient on their own. Resilience must be coordinated across multiple platforms. A multi-provider continuity strategy ensures that if one cloud or edge provider experiences an outage, others can seamlessly pick up the slack. This involves distributing critical services across different vendors or regions, implementing multi-cloud load balancing and data replication, and agreeing on mutual failover arrangements between partners. The CSF will document and classify how each partner's services interconnect and back each other up. Concretely, if one partner's component goes down, the framework will delineate how an equivalent component from another provider can replace or bypass it, maintaining service delivery. By distributing workloads across multiple cloud providers, the risk of a total outage is significantly reduced. The framework will also track interdependencies to prevent cascading failures for instance, ensuring that a network failure in one domain does not isolate dependent services elsewhere. Through this federated approach, resilience becomes a shared responsibility: all providers collectively contribute to and benefit from a more robust, interconnected continuity plan.

This not only improves technical fault tolerance but also builds trust that the entire system can endure stresses that no single provider could handle alone.

By integrating these diverse resilience and availability measures, the CSF strengthens the overall reliability of the multi-provider environment. The project moves beyond isolated uptime solutions toward a collaborative continuity strategy in which all partners' contributions are aligned. This unified approach means the ecosystem can uphold its uptime commitments even amid unexpected disruptions, whether they be cyberattacks, component failures, or natural disasters. In summary, resilience and availability are woven into the very fabric of the architecture through coordination and continuous improvement. Critical cloud-edge services continue to operate within agreed service levels even under the most adverse conditions, ensuring users and applications experience a dependable platform at all times.

Topic 9: Privacy & User Control

9.0 General Description

Privacy and user control is dedicated to protecting personal data and upholding user rights across the system. It includes enforcing privacy-by-design principles, managing user consent and data preferences, implementing data minimization, and techniques like differential privacy for data analysis. The CSF will deliver a consolidated view of how each partner's solutions address privacy concerns, classifying these measures and ensuring they are embedded into the architecture from end to end. All partner components will be mapped against privacy requirements, for instance: whether they properly handle consent or anonymize user data, and the CSF will identify any gaps where additional controls are needed. By collaboratively developing a privacy questionnaire and baseline aligned with regulations like GDPR, the framework guarantees that user data is handled transparently and that users maintain control over their information. This unified stance on privacy fosters user trust and compliance with legal mandates throughout the IPCEI-CIS platform.

9.1 Privacy-by-design Principles

to be added in a future version of the document

9.2 Consent Management

to be added in a future version of the document

9.3 Differential Privacy

to be added in a future version of the document

9.4 Data Minimization Practices

to be added in a future version of the document

Topic 10: Emerging & Future Threats

10.0 General Description

This forward-looking topic addresses the need to anticipate and guard against evolving security challenges. It covers preparation for threats on the horizon such as quantum-computing attacks and the need for quantum-resistant encryption, AI-driven or autonomous attacks, vulnerabilities in novel technologies like smart contracts or satellite/mesh networks, and even potential bio/neuro-technology risks. The CSF will coordinate research and development efforts among partners to identify and classify solutions for these emerging threats, ensuring that the security architecture remains adaptive. Concretely, the CSF will map partner contributions like experimental quantum-safe cryptography implementations or AI-based threat detection tools into the framework. By doing so, it delivers a strategy for continuous innovation in security: as new threats are identified, the framework can evolve by incorporating new controls or best practices. This collaborative approach means the project is not just reacting to current threats but actively preparing for future challenges as a unified front.

10.1 Quantum-resilient Encryption

Quantum-resilient encryption will be one of the defining challenges of the next decade. Once quantum computing becomes capable of breaking current encryption schemes, critical infrastructure, personal data, and industrial secrets could be exposed retroactively. The CSF will therefore prioritize coordination among partners to classify and evaluate emerging quantum safe cryptographic methods. By aligning future research and adoption strategies, the framework ensures that the 8ra ecosystem is equipped to transition smoothly and securely when post-quantum standards mature.

10.2 AI-driven Threats

AI-driven threats represent another major frontier, with artificial intelligence enabling adversaries to design malware, automate large scale campaigns, or conduct deepfake based disinformation operations at unprecedented speed and scale. These capabilities shift the balance of power by lowering the barrier of entry for sophisticated attacks. The CSF will encourage the development of adaptive defenses that harness AI for detection and response, while also ensuring that ethical and explainable safeguards are considered in countering AI misuse.

10.3 Autonomous Agents

Autonomous agents and smart contracts bring new risks as decentralized systems and self-executing code become more widespread. Vulnerabilities in smart contracts have already led to significant financial and operational damages, and future autonomous systems could be manipulated to act against their intended purpose. The CSF will classify solutions that secure autonomous logic, establish trust in decentralized ecosystems, and prevent cascading failures triggered by compromised automated agents.

10.4 DDoS and large-scale Network Attacks

DDoS and large-scale network attacks continue to be among the most visible and persistent threats, and they are evolving in scope and sophistication. Future attack campaigns are expected to leverage billions of connected devices, AI-driven botnets, and amplification vectors capable of exceeding terabit scale traffic volumes. These attacks have the potential to disrupt critical European digital services and cloud-edge operations. NBIP, as topic lead for this domain, will deliver a federated detection and mitigation capability that strengthens resilience against such events. By embedding DDoS defense mechanisms into the CSF framework, partners will benefit from both collective situational awareness and practical mitigation measures, ensuring that availability remains protected even under extreme stress.

10.5 Satellite and Mesh Network Security

Satellite and mesh network security will grow in importance as new communications infrastructures are deployed to extend coverage and enable resilient, distributed connectivity. These systems introduce unique vulnerabilities, from signal interference and spoofing to routing manipulation across loosely governed networks. The CSF will capture and classify approaches for securing satellite and mesh networks, ensuring that they can become reliable elements of Europe's digital backbone without creating exploitable weak points.

10.6 Biological and neural Interfaces

Biological and neural interfaces present longer term, but critical risks as human machine integration accelerates. From medical implants to brain computer interfaces, future digital systems will increasingly interact directly with biological processes. The potential impact of compromise in this domain is profound, raising not only privacy and safety concerns but also ethical considerations. The CSF will track developments in this field to ensure that cybersecurity is embedded from the earliest stages of design and standardization.

10.7 Hybrid Threats

Cross domain hybrid threats, such as supply chain manipulation and combined cyber physical attacks, are expected to intensify as digital dependencies deepen. These threats often span multiple sectors and infrastructures, exploiting trust relationships and interconnections across ecosystems. The CSF will enable partners to collectively classify, share, and anticipate hybrid threat models, ensuring that lessons learned in one sector can be applied proactively across the broader 8ra community.

By coordinating across these diverse yet interconnected domains, the CSF ensures that the 8ra ecosystem is not only resilient against today's risks but also actively prepared for the disruptive security challenges of the future. NBIP, as the lead partner in *Emerging & Future Threats,* will guide this effort, fostering collaboration across projects and partners to ensure that proactive threat intelligence, forward-looking research, and adaptive defense mechanisms remain embedded at the core of Europe's cloud-edge security architecture.

Establishing a Baseline Security Compliance Posture

To establish a baseline security compliance posture, the CSF will define a core set of security requirements that all components must meet. This will be based on regulatory standards, industry best practices, and threat intelligence. This baseline will be developed in collaboration with all 8ra partners to ensure alignment with diverse operational and compliance needs. A structured framework for assessing compliance will be implemented. The CSF will also introduce continuous monitoring mechanisms to ensure that security postures remain up to date in response to evolving developments and threats. By establishing a unified compliance framework, the CSF will facilitate interoperability while maintaining high-security standards across all partner components.

Integration Language Standardization

Based on 8ra reference architecture an API blueprint will be developed to provide a uniform method for accessing security functionalities. This blueprint will define well-structured, standardized APIs for both internal and external consumers. It will also act as a translation layer, allowing seamless integration of diverse security modules supplied by different partners.

This translation layer will ensure interoperability between security components from different partners as a objective. The CSF will establish mechanisms for seamless API adaptation, enabling plug-and-play integration of security modules. This approach will enhance flexibility for partners while maintaining consistency in security operations.

Testbed/Pilot Infrastructure for Validation

A testbed/pilot infrastructure needs to be developed to validate the compatibility of each Security Building Block with the standardized API translation layer. This testbed will also be used to assess compliance with the baseline security posture and verify that security modules function correctly within the broader infrastructure.

Annexes

Building Block analyses

Separate excel spreadsheet

Technical details topic 1 IAM

to be added in a future version of the document

Technical details topic 2 Network security

to be added in a future version of the document

Technical details topic 3 Data security

3.1. TLS Security and Compliance

To deploy a webserver which is secure against known attacks but also compliant with NIS 2 related guidelines, it has to be configured in the following way:

Protocol – a server **must** offer TLS 1.3 and **should** support TLS 1.2. The server **must not** support TLS version lower than 1.2, or SSL 2.0/3.0.

Cipher suites (TLS 1.3) - when using TLS 1.3, a server **must** offer at least one of the following cipher suites (and **should** enable the others):

TLS_AES_128_GCM_SHA256

TLS_AES_256_GCM_SHA384

TLS_CHACHA20_POLY1305_SHA256

TLS_AES_128_CCM_SHA256

Cipher suites (TLS 1.2) - when using TLS 1.2, a server **must** offer at least one of the following cipher suites (preferring ECDHE over DHE, and ECDSA over RSA) and **should** enable the others in the same order they are listed:

TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_ECDSA_WITH_AES_128_CCM

TLS_ECDHE_ECDSA_WITH_AES_256_CCM

TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256

TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

TLS_DHE_RSA_WITH_AES_128_CCM

TLS_DHE_RSA_WITH_AES_256_CCM

TLS Extensions – along with the ones mandated by the respective RFCs, the following extensions

must be enabled when using TLS 1.2

encrypt_then_mac

extended_master_secret

must NOT be enabled

heartbeat

truncated_hmac

early_data

Supported groups – a server **must** always support X25519 / Curve25519. **should** support the following curves:

brainpoolP256r1

brainpoolP384r1

brainpoolP512r1

and, if DHE cipher suites are offered, it **must** support at least one of the following groups:

ffdhe3072

ffdhe4096

ffdhe6144

ffdhe8192

Signature algorithms – a server **must** support the following signature algorithms:

ed25519

rsa_pss_rsae_sha256

rsa_pss_rsae_sha384

rsa_pss_rsae_sha512

rsa_pss_pss_sha256

rsa_pss_pss_sha384

rsa_pss_pss_sha512

and **should** support the following:

ecdsa_brainpoolP256r1tls13_sha256

ecdsa_brainpoolP384r1tls13_sha384

ecdsa_brainpoolP512r1tls13_sha512

Technical details topic 4 Application security

4.1 Secure Coding Practices

4.1.1 Open-Source Tools for Implementation

SonarQube Community Edition – Open-source code quality and security platform that identifies vulnerabilities, code smells, and security hotspots across 30+ programming languages. Integrates with CI/CD pipelines and provides detailed remediation guidance for developers.

Semgrep – Lightweight, fast static analysis tool with customizable rules for detecting security anti-patterns. Open-source with support for multiple languages, excels at finding custom security issues specific to your codebase.

Bandit – Python-focused security linter that finds common security issues in Python code. Designed to be easy to integrate into development workflows with minimal false positives and clear vulnerability descriptions.

ESLint with security plugins – JavaScript/TypeScript linting tool with security-focused plugins (eslint-plugin-security, eslint-plugin-no-unsanitized). Catches XSS vulnerabilities, unsafe regex patterns, and other JavaScript-specific security issues.

Brakeman - Static analysis security scanner specifically designed for Ruby on Rails applications. Scans for SQL injection, XSS, command injection, and Rails-specific vulnerabilities without requiring a running application.

4.1.2 Implementation

Input Validation and Sanitization

Validate and sanitize all user inputs to prevent injection attacks and ensure data integrity.

Implementation Examples:

Whitelist validation: Define allowed characters, formats, and lengths for each input field. For email fields, use regex patterns like ^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ and reject anything that doesn't match.

Parameterized queries: Use prepared statements with bound parameters instead of string concatenation for database queries. In Java: PreparedStatement ps = conn.prepareStatement("SELECT * FROM users WHERE id = ?"); ps.setInt(1, userId);

Context-aware output encoding: Encode data based on where it's used (HTML, JavaScript, URL). Use libraries like OWASP Java Encoder or DOMPurify for JavaScript to prevent XSS attacks.

Principle of Least Privilege

Grant only the minimum permissions necessary for applications, services, and users to perform their functions.

Implementation Examples:

- **Service accounts:** Create dedicated service accounts with restricted permissions. For a web application database connection, grant only SELECT, INSERT, UPDATE on specific tables, not DROP or ALTER permissions.
- **File system permissions:** Set restrictive file permissions (chmod 600 for config files, 755 for executables) and use separate user accounts for each service to prevent lateral movement in case of compromise.
- **RBAC:** Implement granular roles like "viewer," "editor," "admin" with specific permissions. Use middleware to check roles before allowing access to sensitive operations.

Secure Authentication and Session Management

Implement robust authentication mechanisms and protect session data from hijacking and fixation attacks.

Implementation Examples:

- **Multi-Factor Authentication (MFA):** Implement TOTP-based 2FA using libraries like Google Authenticator or hardware tokens. Require MFA for privileged accounts and sensitive operations.
- **Secure session handling:** Generate cryptographically random session IDs (minimum 128 bits), set secure cookie flags (HttpOnly, Secure, SameSite=Strict), implement absolute and idle timeouts (30 minutes idle, 12 hours absolute).
- **Password security:** Enforce strong password policies (minimum 12 characters, complexity requirements), use bcrypt or Argon2 for hashing with appropriate work factors (bcrypt cost 12+), implement account lockout after 5 failed attempts.

Secure Error Handling and Logging

Prevent information leakage through error messages while maintaining comprehensive audit trails.

Implementation Examples:

- **Generic error messages:** Show users' generic messages like "An error occurred. Please try again." while logging detailed error information server-side with correlation IDs for troubleshooting.
- **Structured logging:** Use logging frameworks (Log4j, Winston, Python logging) with appropriate levels (DEBUG, INFO, WARN, ERROR). Log security events like failed logins, privilege escalations, and data access with timestamp, user, IP, and action.
- **Log sanitization:** Filter sensitive data (passwords, credit cards, PII) from logs using regex patterns or dedicated sanitization libraries before writing to log files or SIEM systems.

Secrets Management

Protect sensitive credentials, API keys, and cryptographic keys from exposure in code or configuration.

Implementation Examples:

- **Environment variables and vaults:** Store secrets in environment variables or dedicated secret management systems. Access them programmatically at runtime.
- **Secret rotation:** Implement automated rotation policies for API keys and credentials (90-day rotation). Use versioned secrets so old credentials remain valid during rotation windows.
- **Encryption at rest:** Encrypt configuration files containing sensitive data using tools like ansible-vault or git-crypt. Use Key Management Service (KMS) for managing encryption keys separately from encrypted data.

Secure Cryptographic Practices

Use strong, modern cryptographic algorithms and proper implementation patterns.

Implementation Examples:

- **Modern algorithms:** Use AES-256-GCM for symmetric encryption, RSA-2048+ or ECDSA-256+ for asymmetric operations, and SHA-256+ for hashing. Avoid deprecated algorithms like MD5, SHA-1, DES, or RC4.
- **Proper key generation:** Generate cryptographic keys using secure random number generators (SecureRandom in Java, crypto.randomBytes in Node.js). Never hardcode keys or use weak seeds.
- **TLS/SSL configuration:** Enforce TLS 1.2+ with strong cipher suites, implement certificate pinning for mobile apps, and use HSTS headers to prevent downgrade attacks.

Secure Dependency Management

Monitor and update third-party libraries and dependencies to address known vulnerabilities.

Implementation Examples:

- **Dependency scanning:** Integrate tools like npm audit, pip-audit, or OWASP Dependency-Check into CI/CD pipelines. Fail builds when critical vulnerabilities are detected.
- **Version pinning:** Pin exact dependency versions in package.json, requirements.txt, or pom.xml to prevent unexpected updates. Use lock files (package-lock.json, Pipfile.lock) for reproducible builds.
- **Regular updates:** Schedule monthly dependency reviews, prioritize security patches, test updates in staging environments, and maintain a policy for addressing vulnerabilities within SLA timeframes (critical: 7 days, high: 30 days).

Secure Data Storage and Transmission

Protect sensitive data both at rest and in transit using encryption and secure protocols.

Implementation Examples:

- **Database encryption:** Enable Transparent Data Encryption (TDE) for databases, encrypt specific columns containing PII using application-level encryption, and store encryption keys separately in KMS.
- **Secure transmission:** Enforce HTTPS for all communications, use certificate validation, implement certificate pinning for mobile applications, and ensure API endpoints reject HTTP connections.
- **Data classification:** Classify data by sensitivity (public, internal, confidential, restricted) and apply appropriate protection measures. Implement data masking for non-production environments.

Security Code Reviews and Pair Programming

Establish peer review processes to catch security issues before code reaches production.

Implementation Examples:

- **Security-focused reviews:** Create security checklists for code reviews covering input validation, authentication, authorization, error handling, and cryptography. Require security approval for changes touching sensitive areas.
- **Automated PR checks:** Configure GitHub/GitLab to require passing security scans (SAST, secret detection, dependency checks) before allowing merge. Use branch protection rules to enforce reviews.
- **Training and knowledge sharing:** Conduct regular secure coding training sessions, share OWASP Top 10 examples, perform threat modeling exercises, and maintain internal security documentation.

Secure Configuration Management

Ensure applications and infrastructure use secure default configurations and hardened settings.

Implementation Examples:

- **Security baselines:** Apply CIS benchmarks or vendor hardening guides for operating systems, databases, and web servers. Disable unnecessary services, remove default accounts, and change default passwords.
- **Configuration as code:** Store infrastructure configuration in version control (Terraform, Ansible), implement peer review for changes, and use policy-as-code tools like OPA to validate configurations before deployment.
- **Secure defaults:** Configure applications with security-first defaults (HTTPS only, secure session cookies, restrictive CORS policies, disabled directory listing) and require explicit action to relax security controls.

4.2 Static and dynamic analysis

4.2.1 Open-Source Tools for Implementation

OWASP ZAP (Zed Attack Proxy) – Comprehensive DAST tool for finding vulnerabilities in web applications during development and testing. Features automated scanners, passive scanning, fuzzing capabilities, and a robust API for CI/CD integration.

SpotBugs (formerly FindBugs) – Static analysis tool for Java bytecode that identifies potential bugs and security vulnerabilities. Works with plugins like Find Security Bugs to detect security-relevant issues like SQL injection and weak cryptography.

Grype – Vulnerability scanner for container images and filesystems that works independently or with SBOMs. Fast, accurate matching against multiple vulnerability databases with minimal false positives.

Trivy – Comprehensive security scanner for containers, filesystems, IaC, and Kubernetes configurations. Detects vulnerabilities in OS packages and language-specific dependencies, misconfigurations, and exposed secrets.

PMD – Source code analyzer that finds common programming flaws across multiple languages (Java, JavaScript, Apex, etc.). Includes security-focused rule sets and integrates with major IDEs and build tools.

Nikto – Web server scanner that performs comprehensive tests against web servers for dangerous files, outdated software versions, and server configuration issues. Fast, regularly updated with new vulnerability checks.

4.2.2 Implementation

SAST

Analyze source code without execution to identify security vulnerabilities early in development.

Implementation Examples:

- **CI/CD integration:** Integrate SonarQube or Semgrep into your build pipeline to scan every commit. Configure quality gates to fail builds when critical vulnerabilities are detected (CVSS 7.0+).
- **IDE plugins:** Install security-focused linters and extensions (ESLint with security plugins, Bandit for Python, Brakeman for Rails) that provide real-time feedback as developers write code.
- **Incremental scanning:** Configure tools to scan only changed files during development for faster feedback, and perform full codebase scans nightly or before releases to catch issues in untouched legacy code.

Dynamic Application Security Testing

Test running applications to identify vulnerabilities that only manifest during execution.

Implementation Examples:

Automated scanning: Deploy OWASP ZAP or Nikto in staging environments to scan applications after deployment. Configure authentication (session tokens, API keys) to test authenticated functionality.

- **Scheduled penetration tests:** Run weekly or monthly automated DAST scans against staging and production environments. Configure scans during low-traffic periods to minimize performance impact.
- **Attack simulation:** Use tools to simulate OWASP Top 10 attacks (SQL injection, XSS, CSRF) against your application. Test with both valid and malicious payloads to verify security controls.

Software Composition Analysis

Scan third-party libraries and dependencies for known vulnerabilities and license issues.

Implementation Examples:

- **Automated dependency scanning:** Integrate OWASP Dependency-Check or Trivy into CI/CD pipelines to scan package manifests (package.json, pom.xml, requirements.txt) and fail builds for high-severity vulnerabilities.
- **Real-time monitoring:** Enable GitHub Dependabot or Snyk to automatically create pull requests when vulnerabilities are discovered in dependencies, with upgrade recommendations and fix information.
- **License compliance:** Use tools like FOSSA or FOSSology to scan dependencies for license conflicts, GPL contamination risks, and ensure compliance with organizational policies.

Interactive Application Security Testing

Combine SAST and DAST approaches using instrumentation for more accurate vulnerability detection.

Implementation Examples:

- **Agent instrumentation:** Deploy IAST agents (like Contrast Security Community Edition) in QA environments that instrument the application runtime to observe code execution paths during testing.
- **Test coverage mapping:** Integrate with automated test suites (JUnit, pytest, Jest) so the IAST agent analyzes security during functional tests, achieving higher code coverage than standalone DAST.
- **Real-time feedback:** Configure IAST tools to provide immediate vulnerability alerts during development and testing, with exact code locations and data flow analysis for faster remediation.

Container and Infrastructure Scanning

Analyze container images and infrastructure-as-code for security misconfigurations and vulnerabilities.

Implementation Examples:

- **Image scanning pipeline:** Integrate Trivy or Clair into container build processes to scan Docker images before pushing to registries. Reject images with critical CVEs or require security approval for exceptions.
- **IaC security scanning:** Use Checkov or tfsec to analyze Terraform, CloudFormation, and Kubernetes manifests for misconfigurations (exposed ports, missing encryption, weak IAM policies) before deployment.
- **Registry scanning:** Enable continuous scanning in container registries (Docker Hub, ECR, Harbor) to detect new vulnerabilities in stored images and alert teams when patches are needed.

Fuzz Testing

Use automated input generation to discover crashes, memory leaks, and security vulnerabilities.

Implementation Examples:

- **Coverage-guided fuzzing:** Deploy AFL++ or libFuzzer to generate test inputs that maximize code coverage. Run fuzzing campaigns for at least 24-48 hours on file parsers, protocol implementations, and input handlers.
- **API fuzzing:** Use tools like RESTler or Fuzz-lightyear with OpenAPI specifications to generate malformed requests, boundary values, and unexpected data types to test API robustness.
- **Crash analysis:** Configure fuzzing tools to save crash-inducing inputs, integrate with debuggers (GDB, LLDB) for root cause analysis, and track unique crash signatures to prioritize fixes.

Mobile Application Security Testing

Analyze mobile applications for platform-specific vulnerabilities and insecure data storage.

Implementation Examples:

- **Automated scanning:** Use MobSF to analyze Android APK and iOS IPA files for insecure data storage, weak cryptography, hardcoded secrets, and dangerous permissions before release.
- **Runtime analysis:** Deploy Frida or Objection to hook into running mobile applications, intercept API calls, bypass SSL pinning, and analyze runtime behavior during security assessments.
- **Backend API testing:** Test mobile backend APIs separately using OWASP ZAP or Burp Suite, focusing on authentication bypass, excessive data exposure, and business logic flaws.

Infrastructure Vulnerability Scanning

Regularly scan networks, systems, and services for known vulnerabilities and misconfigurations.

Implementation Examples:

- **Network scanning:** Deploy OpenVAS or Nmap with NSE scripts to scan internal and external networks weekly. Identify exposed services, outdated software versions, and missing security patches.
- **Configuration auditing:** Use Lynis for Unix/Linux systems or ScoutSuite for cloud environments to audit configurations against security benchmarks (CIS, NIST) and generate hardening recommendations.
- **Continuous monitoring:** Implement Nuclei with custom templates to continuously scan for newly disclosed vulnerabilities (CVEs), exposed sensitive endpoints, and infrastructure changes.

Database Security Testing

Identify SQL injection vulnerabilities, weak authentication, and configuration issues in databases.

Implementation Examples:

- **SQL injection testing:** Use SQLMap to test web applications for SQL injection vulnerabilities, including blind SQL injection, time-based attacks, and union-based extraction techniques.

- **Database configuration scanning:** Run dbprotect or custom scripts to check for default credentials, missing encryption, overly permissive user privileges, and disabled audit logging.
- **Access control testing:** Verify that database users have minimum required privileges, test for privilege escalation paths, and ensure sensitive data access is properly logged and monitored.

Security Regression Testing

Ensure previously fixed vulnerabilities don't reappear and security controls remain effective over time.

Implementation Examples:

- **Automated security test suites:** Maintain a collection of security-specific test cases (authentication bypass attempts, injection payloads, authorization checks) that run with every build or deployment.
- **Vulnerability re-testing:** After fixing security issues, add specific test cases that verify the fix and prevent regression. Store these tests alongside functional tests in your test suite.
- **Baseline comparison:** Establish security baselines with initial SAST/DAST scans and compare subsequent scans to identify new vulnerabilities or degraded security posture over time.

4.3 Application Firewall

4.3.1 Open-Source Tools for Implementation

Coraza – Modern WAF framework compatible with ModSecurity rules but written in Go for improved performance. Designed as a library that can be embedded in applications or used as a standalone proxy.

ModSecurity – Leading open-source WAF engine that can be deployed with Apache, Nginx, and IIS web servers. Highly flexible with extensive rule-writing capabilities and support for the OWASP Core Rule Set (CRS).

Shadow Daemon – Web Application Firewall that uses blacklists and whitelists to detect and prevent attacks. Features a web interface for managing rules and analyzing threats with minimal false positives.

NAXSI (Nginx Anti XSS & SQL Injection) – Third-party Nginx module providing WAF capabilities with a focus on simplicity and low resource usage. Uses a whitelist-based approach to detect and block malicious requests.

Vulture – Complete application delivery and security platform combining reverse proxy, WAF (based on ModSecurity), load balancing, and authentication portal. Provides a web interface for centralized management of security policies.

4.3.2 Implementation

OWASP Top 10 Protection

Deploy WAF rules to defend against the most critical web application security risks.

Implementation Examples:

- **Core Rule Set deployment:** Install ModSecurity with OWASP CRS (Core Rule Set) that provides pre-configured rules for SQL injection, XSS, remote file inclusion, and other OWASP Top 10 threats.
- **Paranoia level tuning:** Start with OWASP CRS paranoia level 1 for basic protection, gradually increase to level 2-3 after tuning false positives, balancing security and application functionality.
- **Custom rule development:** Create application-specific rules for known attack patterns. For example, if your app doesn't accept file uploads, block all multipart/form-data requests to sensitive endpoints.

Rate Limiting and DDoS Mitigation

Protect applications from abuse, brute force attacks, and denial-of-service attempts.

Implementation Examples:

- **Request rate limiting:** Configure nginx rate limiting to allow maximum 10 requests per second per IP address for login endpoints, 100 requests per second for API endpoints, with burst allowances for legitimate traffic spikes.
- **Connection limiting:** Implement Fail2Ban to monitor WAF logs and automatically ban IP addresses that trigger multiple security rules within 5 minutes, with escalating ban durations (1 hour, 24 hours, permanent).
- **Geographic filtering:** Use GeoIP2 modules to block or challenge traffic from countries where you don't operate, or apply stricter rate limits to regions with historically high attack volumes.

Bot Detection and Management

Identify and control automated traffic to prevent scraping, credential stuffing, and spam.

Implementation Examples:

- **User-agent filtering:** Create ModSecurity rules to block known malicious bot user-agents (scrapers, vulnerability scanners) while allowing legitimate bots based on verified IP ranges.
- **Behavioral analysis:** Implement JavaScript challenges or CAPTCHA for suspicious behavior patterns (rapid page requests, sequential URL enumeration, no cookie support) while allowing normal users seamless access.
- **Bot reputation lists:** Integrate threat intelligence feeds containing known bot IPs and update WAF blocklists hourly using automated scripts that fetch and apply updated IP ranges.

Virtual Patching

Apply temporary protection rules while waiting for application patches to be developed and deployed.

Implementation Examples:

- **CVE-specific rules:** When CVE-2024-XXXX is announced affecting your CMS, immediately create ModSecurity rules blocking the specific attack pattern (e.g., malicious URL parameter) until the vendor patch is applied.
- **Rapid response rules:** Maintain a virtual patching workflow: within 4 hours of vulnerability disclosure, analyze the exploit, create blocking rules, test in staging, and deploy to production WAF.
- **Rule retirement:** Track virtual patches in a spreadsheet with application version, CVE number, deployment date, and planned removal date (after application is patched and verified).

Request and Response Filtering

Inspect HTTP traffic for malicious content and prevent data leakage.

Implementation Examples:

- **Request body inspection:** Configure ModSecurity to inspect POST request bodies up to 128KB for injection attacks, limiting inspection size to balance security and performance impact.
- **Response data masking:** Create rules to detect and block responses containing sensitive patterns (credit card numbers matching regex \d{4}[-\s]?\d{4}[-\s]?\d{4}[-\s]?\d{4}, Social Security numbers, API keys).
- **File upload restrictions:** Implement rules restricting upload file types to whitelisted extensions (.jpg, .png, .pdf), maximum sizes (10MB), and block executable content (.exe, .sh, .php) in upload directories.

SSL/TLS Security Enforcement

Ensure secure encrypted communications and prevent protocol downgrade attacks.

Implementation Examples:

- **Strong cipher configuration:** Configure nginx or Apache to use only TLS 1.2+ with strong cipher suites (ECDHE-RSA-AES256-GCM-SHA384), disable weak ciphers (RC4, DES, 3DES), and prioritize forward secrecy.
- **HSTS implementation:** Add Strict-Transport-Security header with max-age=31536000; includeSubDomains; preload to force HTTPS and submit domain to browser HSTS preload lists.
- **Certificate validation:** Implement OCSP stapling for certificate validation, configure automatic certificate renewal with Let's Encrypt or other providers, and monitor certificate expiration 30 days in advance.

API-Specific Protection

Apply specialized WAF rules for REST and GraphQL API security.

Implementation Examples:

- **API rate limiting:** Implement token-bucket rate limiting per API key (1000 requests/hour for free tier, 100,000/hour for premium), with separate limits for expensive operations (search, reporting).
- **Schema validation:** Deploy OpenAPI-aware WAF rules that validate requests match API specification (required fields, data types, enum values) and reject malformed requests before reaching application.
- **GraphQL query depth limiting:** Implement rules limiting GraphQL query depth to 5 levels and complexity scoring to prevent resource exhaustion from deeply nested or circular queries.

Logging, Monitoring, and Alerting

Maintain visibility into attacks and security events for incident response.

Implementation Examples:

- **Centralized logging:** Configure ModSecurity to send audit logs to ELK Stack or Graylog with structured JSON format including timestamp, source IP, rule ID, payload, and action taken (block/alert).
- **Real-time alerting:** Set up alerts in Grafana or Prometheus when attack thresholds are exceeded (>100 blocked requests from single IP in 5 minutes, >1000 SQL injection attempts per hour).
- **Dashboard creation:** Build security dashboards displaying blocked attacks by type, top attacker IPs/countries, blocked requests over time, and false positive rates for continuous WAF tuning.

False Positive Management

Minimize legitimate traffic disruption while maintaining security effectiveness.

Implementation Examples:

- **Baseline learning:** Run WAF in detection-only mode for 1-2 weeks to understand normal traffic patterns, identify frequently triggered rules on legitimate requests, and build tuning exceptions.
- **Rule tuning:** Create rule exceptions for specific paths or parameters. For example, if CRS rule 942100 (SQL injection) blocks legitimate rich text editor content, disable it only for /api/comments endpoint.
- **Gradual enforcement:** Start with permissive settings (log-only), progressively enable blocking for high-confidence rules (paranoia level 1), then carefully add more aggressive rules while monitoring user impact.

Geographic and IP Reputation Filtering

Control access based on geographic location and IP address reputation.

Implementation Examples:

- **GeoIP blocking:** Use MaxMind GeoIP2 database with nginx or ModSecurity to block traffic from high-risk countries while allowing VPN/proxy detection to identify circumvention attempts.
- **IP reputation integration:** Subscribe to threat intelligence feeds (AbuseIPDB, Spamhaus) and automatically update WAF blocklists every 6 hours with IPs flagged for malicious activity in last 30 days.
- **Whitelist management:** Maintain IP whitelists for known good sources (corporate offices, partner networks, monitoring services) that bypass geographic and reputation filters while still applying other WAF rules.

4.4 API Security

4.4.1 Open-Source Tools for Implementation

Kong Gateway (Open-Source) – Powerful API gateway with extensive security plugin ecosystem including authentication (JWT, OAuth2, API keys), rate limiting, request validation, and IP restriction. Highly scalable and cloud-native.

OWASP ZAP – Besides web application testing, ZAP excels at API security testing with OpenAPI/Swagger support, automated API scanning, and custom API attack scripts. Can import API definitions and perform comprehensive security testing.

Tyk Gateway (Open-Source) – API gateway providing authentication, rate limiting, quotas, and API versioning. Features include analytics, webhooks, and comprehensive access control policies.

Apisix – Cloud-native API gateway with dynamic routing, authentication, rate limiting, and comprehensive observability. Built on Nginx and supports plugins for various security requirements with low latency.

42Crunch REST API Static Security Testing – Free community edition that performs security audits on OpenAPI specifications. Identifies security issues in API design before implementation with detailed remediation advice.

Metlo – Open-source API security platform providing API inventory discovery, sensitive data detection, and security testing. Analyzes API traffic to identify vulnerabilities and compliance issues automatically.

4.4.2 Implementation

Strong Authentication Mechanisms

Implement robust identity verification for API consumers using modern protocols.

Implementation Examples:

- **OAuth 2.0 implementation:** Deploy Keycloak or ORY Hydra as authorization server, implement authorization code flow for web applications, client credentials for service-to-service, and require refresh token rotation.
- **JWT token validation:** Issue short-lived access tokens (15 minutes), validate token signatures using RS256 algorithm, verify issuer/audience claims, check expiration, and implement token revocation lists for compromised tokens.
- **API key management:** Generate cryptographically random API keys (minimum 32 bytes), hash keys before storing (using bcrypt), implement key rotation policies (90 days), and support multiple keys per client for zero-downtime rotation.

Fine-Grained Authorization

Control access to API resources and operations based on user identity and permissions.

Implementation Examples:

- **RBAC implementation:** Define roles (admin, editor, viewer) with specific permissions, implement middleware checking user roles before allowing operations, and enforce least privilege by default denying all access.
- **Attribute-Based Access Control (ABAC):** Deploy Open Policy Agent (OPA) to evaluate policies based on user attributes, resource properties, and environmental context. Example: "Allow if user.department==resource.department AND time.hour>=9 AND time.hour<=17".
- **Scope-based authorization:** Implement OAuth scopes (read:users, write:orders) granularly, validate required scopes in API endpoints, and document scope requirements in API specifications for client developers.

Comprehensive Rate Limiting

Protect APIs from abuse, resource exhaustion, and ensure fair usage across consumers.

Implementation Examples:

- **Tiered rate limiting:** Implement Kong or Tyk with multiple tiers (free: 100 req/hour, basic: 1000 req/hour, enterprise: 100,000 req/hour), using Redis for distributed rate limit counters across multiple API gateway instances.
- **Endpoint-specific limits:** Apply different limits based on computational cost: 10 requests/minute for expensive search operations, 1000 requests/minute for simple CRUD operations, unlimited for health checks.

- **Sliding window algorithm:** Use sliding window counters instead of fixed windows to prevent traffic spikes at window boundaries, distributing load more evenly and providing smoother user experience.

Input Validation and Schema Enforcement

Validate all API inputs against defined schemas to prevent injection and data integrity issues.

Implementation Examples:

- **OpenAPI validation:** Use Express OpenAPI Validator or similar middleware to automatically validate requests against OpenAPI 3.0 specifications, rejecting requests with missing required fields, wrong types, or out-of-range values.
- **JSON Schema validation:** Implement AJV (Another JSON Schema Validator) to validate request payloads, enforce maximum string lengths (255 chars for names), numeric ranges (age: 0-150), and regex patterns for emails/phones.
- **Query parameter sanitization:** Whitelist allowed query parameters, validate data types (integers for pagination), implement maximum values (limit: 100), and reject requests with unexpected or dangerous characters.

Secure Data Transmission

Ensure API communications are encrypted and protected from interception.

Implementation Examples:

- **TLS enforcement:** Configure API gateways to require TLS 1.2+ only, use strong cipher suites (ECDHE with AES-GCM), implement HSTS headers, and redirect all HTTP requests to HTTPS with 301 status codes.
- **Certificate pinning:** For mobile applications, implement certificate pinning to prevent man-in-the-middle attacks, pin to intermediate CA certificates (not leaf certificates) for flexibility, and include backup pins.
- **mTLS for sensitive APIs:** Implement mutual TLS authentication for B2B APIs or microservice communication, requiring client certificates for connection establishment, validating certificates against trusted CA.

API Gateway Deployment

Centralize security controls, monitoring, and traffic management through API gateways.

Implementation Examples:

- **Kong Gateway setup:** Deploy Kong as central API gateway, configure plugins for authentication (JWT, OAuth), rate limiting, request/response transformation, and CORS handling across all APIs.
- **Service mesh integration:** Implement Istio or Linkerd for microservices, providing mutual TLS between services, traffic management, distributed tracing, and policy enforcement at the sidecar proxy level.

- **Multi-region deployment:** Deploy API gateways in multiple geographic regions for low latency, implement health checks and automatic failover, synchronize rate limit counters and authentication state across regions.

API Versioning and Deprecation

Maintain backward compatibility while evolving APIs securely and deprecating vulnerable versions.

Implementation Examples:

- **URL versioning:** Implement version in URL path (/api/v1/users, /api/v2/users), maintain security patches for N-1 version, announce deprecation 6 months before removal with response headers Sunset: Sat, 31 Dec 2024 23:59:59 GMT.
- **Header-based versioning:** Use custom headers (API-Version: 2024-01-15) for versioning, maintain security updates for all versions released in last 12 months, force upgrade for critical vulnerabilities.
- **Deprecation workflow:** Mark deprecated endpoints in OpenAPI specs, return Deprecation: true headers, log usage metrics, email affected consumers, and provide migration guides before removal.

API Security Testing

Regularly assess API security through automated and manual testing approaches.

Implementation Examples:

- **OWASP API Top 10 testing:** Use OWASP ZAP with API scanning profiles to test for broken authentication, excessive data exposure, injection flaws, and security misconfigurations against staging APIs.
- **Fuzzing campaigns:** Deploy RESTler or similar fuzzing tools with OpenAPI specifications to generate unexpected inputs, boundary values, malformed JSON, and observe API behavior for crashes or errors.
- **Penetration testing:** Conduct quarterly security assessments focusing on authentication bypass, authorization flaws, business logic vulnerabilities, and rate limit circumvention techniques.

API Monitoring and Anomaly Detection

Detect suspicious patterns, performance issues, and security incidents in real-time.

Implementation Examples:

- **Metrics collection:** Implement Prometheus exporters collecting API metrics (request rates, error rates, latency percentiles, authentication failures), visualize in Grafana dashboards with alerts for anomalies.
- **Behavioral analysis:** Deploy machine learning models or rule-based systems detecting unusual patterns (geographic anomalies, off-hours access, excessive failed authentication, rapid endpoint enumeration).

- **Distributed tracing:** Implement Jaeger or Zipkin for request tracing across microservices, identifying performance bottlenecks, debugging failures, and correlating security events across service boundaries.

API Documentation and Developer Security

Provide secure API documentation and educate developers on proper implementation.

Implementation Examples:

- **Secure documentation portals:** Use tools like Redoc or Swagger UI with authentication required for internal APIs, document security requirements (authentication methods, required scopes, rate limits) prominently.
- **Security guidelines:** Publish API security best practices documentation covering authentication implementation, error handling, data validation, and common vulnerabilities with code examples for multiple languages.
- **SDK security:** Provide official SDKs implementing security best practices (automatic token refresh, request signing, certificate validation), regularly update for vulnerabilities, and discourage custom implementations.

4.5 Software Bill of Materials

4.5.1 Open-Source Tools for Implementation

CycloneDX CLI – Official tool for generating and manipulating CycloneDX SBOMs across multiple languages and package managers. Supports conversion between formats, merging SBOMs, and validation.

Syft (by Anchore) – Comprehensive CLI tool that generates SBOMs for container images, filesystems, directories, and archives. Supports multiple output formats (SPDX, CycloneDX, JSON) and integrates easily into CI/CD pipelines.

Dependency-Track – Comprehensive component analysis platform that ingests SBOMs (CycloneDX, SPDX) and correlates components with vulnerability databases (NVD, GitHub, OSS Index). Provides continuous monitoring, risk scoring, and policy violation alerts.

Tern – Container inspection tool that generates SBOMs specifically for container images by analyzing layers. Particularly useful for understanding the composition of base images and identifying package origins.

SPDX Tools – Official SPDX utilities for creating, validating, and converting SPDX documents. Includes libraries for Java, Python, and Go to programmatically work with SPDX-formatted SBOMs.

OSS Review Toolkit (ORT) – Comprehensive toolkit for automating SBOM generation, license compliance checks, and vulnerability scanning. Supports numerous package managers and provides policy-based evaluation of dependencies.

4.5.2 Implementation

Automated SBOM Generation

Create comprehensive software bills of materials during build processes.

Implementation Examples:

- **Container image SBOMs:** Integrate Syft into Docker build pipelines with command syft packages container-image:latest -o cyclonedx-json > sbom.json, generating SBOMs for every container image before pushing to registries.
- **Language-specific generation:** Use CycloneDX Maven Plugin for Java projects (mvn cyclonedx:makeAggregateBom), cyclonedx-bom for Node.js (npx @cyclonedx/bom), or CycloneDX Python for Python (cyclonedx-py), running as part of CI/CD builds.
- **Multi-format support:** Generate SBOMs in both SPDX and CycloneDX formats for interoperability, storing both formats alongside release artifacts for different tools and customer requirements.

Vulnerability Tracking and Monitoring

Continuously monitor SBOM components against vulnerability databases for security issues.

Implementation Examples:

- **Dependency-Track deployment:** Deploy Dependency-Track as central SBOM management platform, configure API upload from CI/CD pipelines, enable automatic vulnerability scanning against NVD, GitHub Advisories, and OSS Index databases.
- **Automated alerts:** Configure Dependency-Track to send email/Slack notifications when new CVEs affect components with CVSS scores >7.0, including affected versions, recommended upgrades, and exploitability information.
- **Dashboard monitoring:** Create dashboards showing vulnerability trends over time, top vulnerable components, percentage of components with known vulnerabilities, and mean time to remediation metrics.

License Compliance Management

Track open-source licenses to ensure legal compliance and avoid license conflicts.

Implementation Examples:

- **License policy enforcement:** Configure Dependency-Track or FOSSA with organizational policies prohibiting GPL licenses in proprietary products, flagging AGPL licenses, and requiring legal review for dual-licensed components.
- **License compatibility checking:** Implement automated checks ensuring license combinations are compatible (e.g., MIT+Apache-2.0 compatible, but GPL+proprietary incompatible), blocking builds with violations.

- **License obligation tracking:** Maintain database of license obligations (attribution requirements, source code disclosure), generate attribution files automatically from SBOMs for product documentation.

Supply Chain Transparency

Document component provenance and verify integrity throughout the software supply chain.

Implementation Examples:

- **Package signature verification:** Verify npm package signatures using npm audit signatures, validate PyPI package checksums against published hashes, and verify Maven artifact GPG signatures before including in builds.
- **SBOM signing:** Sign generated SBOMs using GPG or Sigstore cosign (cosign sign-blob sbom.json > sbom.json.sig), providing cryptographic proof of SBOM authenticity and detecting tampering.
- **Provenance tracking:** Include SBOM metadata documenting component sources (public registries, private repositories, vendor-supplied), download URLs, checksums, and timestamps for audit trails.

Component Inventory and Discovery

Maintain accurate inventories of all software components across applications and environments.

Implementation Examples:

- **Automated discovery:** Deploy Syft or Tern to scan existing container images in registries, generating SBOMs for legacy applications without CI/CD integration, identifying unknown or shadow dependencies.
- **Runtime analysis:** Use tools scanning running containers or file systems to discover components not captured during build (dynamically loaded libraries, plugins, embedded dependencies).
- **Centralized inventory:** Aggregate SBOMs from all applications into Dependency-Track, providing organization-wide visibility into component usage, identifying redundant dependencies, and tracking component proliferation.

SBOM Integration in CI/CD

Embed SBOM generation and validation as automated steps in development pipelines.

Implementation Examples:

- **Pipeline integration:** Add SBOM generation stage in Jenkins/GitLab CI after build: syft packages dir:. -o cyclonedx-json=sbom.json, upload to Dependency-Track via API, and fail pipeline if critical vulnerabilities detected.
- **Quality gates:** Implement security gates checking SBOM against policies (no GPL licenses, no critical CVEs, all components from approved sources) before allowing deployment to production.

- **Artifact association:** Store SBOMs alongside release artifacts in artifact repositories (Nexus, Artifactory), container registries (as separate layer or annotation), or release management systems.

Third-Party SBOM Collection

Request and manage SBOMs from vendors and third-party component providers.

Implementation Examples:

- **Vendor requirements:** Include SBOM delivery requirements in procurement contracts (SPDX or CycloneDX format, updated with each release, within 48 hours of vulnerability disclosure).
- **SBOM validation:** Validate vendor-supplied SBOMs for completeness (all components listed, version numbers included, license information present), verify against known component lists from scanning tools.
- **SBOM aggregation:** Combine application SBOMs with vendor-supplied SBOMs for third-party components, creating comprehensive system-level SBOMs covering entire application stack including COTS products.

Vulnerability Response Workflow

Establish processes for responding to vulnerabilities discovered through SBOM analysis.

Implementation Examples:

- **Triage process:** When CVE alerts received, security team assesses exploitability (public exploit available?), reachability (vulnerable code path executed?), and impact (data exposure? service disruption?) within 24 hours.
- **Remediation tracking:** Create Jira tickets for vulnerabilities requiring action, assign severity-based SLAs (critical: 7 days, high: 30 days, medium: 90 days), track remediation status in dashboards.
- **Patch deployment:** Prioritize patches based on risk scoring combining CVSS, exploitability, and business impact, test patches in staging environments, deploy to production during maintenance windows, and update SBOMs post-deployment.

SBOM Format Standardization

Adopt and maintain consistent SBOM formats across the organization for interoperability.

Implementation Examples:

- **Format selection:** Standardize on CycloneDX 1.5+ for detailed vulnerability and licensing data, or SPDX 2.3+ for broader tool compatibility, documenting format choice in security policies.
- **Schema validation:** Implement automated validation using official schema validators (cyclonedx-cli validate --input-file sbom.json) in CI/CD pipelines to ensure SBOMs conform to specifications before storage.

- **Conversion tools:** Deploy SBOM conversion utilities for legacy systems producing different formats, using tools like sbom-utility or CycloneDX CLI to convert between SPDX, CycloneDX, and SWID formats.

SBOM Distribution and Sharing

Make SBOMs available to stakeholders while protecting sensitive information.

Implementation Examples:

- **Customer delivery:** Publish SBOMs on product download pages or customer portals, include SBOMs in software packages (in /usr/share/doc/ or .sbom directory), or provide via API for automated consumption.
- **Internal distribution:** Store SBOMs in centralized repositories (Artifactory, S3 buckets) with appropriate access controls, enable security and compliance teams to query SBOM database for risk assessments.
- **Sensitive data redaction:** Remove internal URLs, private repository references, and proprietary component details from customer-facing SBOMs while maintaining complete internal versions for vulnerability management.

Technical details topic 5 Endpoint & Device Security

5.1. Architectural Requirements

The following requirements constitute the non-negotiable blueprint for implementing a Zero Trust Architecture at the hardware level for all endpoint and edge devices participating in the 8ra ecosystem. These mandates are the technical materialization of the "never trust, always verify" principle, creating a resilient, defensible, and auditable foundation by moving beyond obsolete perimeter-based security models. This approach ensures that trust is never implicitly granted but is continuously verified based on cryptographic proof of platform integrity, beginning at the silicon level.

5.1.1 Hardware Root of Trust (HRoT) Mandate

Primary Mandate (TPM 2.0): All devices capable of supporting it must be equipped with a hardware root of trust compliant with the Trusted

Platform Module (TPM) 2.0 specification (ISO/IEC 11889). This serves as the immutable anchor for measurement and reporting.

Constrained & Legacy Devices (DICE): We acknowledge that a significant portion of the current IIoT landscape ("Brownfield") and ultra-low-power sensors cannot support full TPM 2.0 implementations due to cost or energy constraints. For these devices, the DICE architecture is an accepted alternative standard. DICE provides a lightweight, hardware-based root of trust suitable for microcontrollers (MCUs) while maintaining the requirement for a cryptographically unique device identity.

Trusted Execution Environment (TEE): Where workloads require strong isolation from privileged system software, devices shall utilize a TEE (e.g., SGX, SEV-SNP, TrustZone) to protect runtime code and data.

5.1.2 Secure and Measured Boot Process

All devices must implement a Measured Boot process to ensure a verifiable and secure startup sequence. This process begins with an immutable Core Root of Trust for Measurement (CRTM), which measures the initial firmware (e.g., UEFI/BIOS). This measurement hash is securely stored within the TPM's Platform Configuration Registers (PCRs). This process must create an unbroken cryptographic chain of trust, where each subsequent component – from bootloader to operating system kernel – is measured before it is executed, with each measurement being extended into the PCRs.

For runtime integrity, devices shall leverage the Integrity Measurement Architecture (IMA) or an equivalent kernel-level module. IMA extends the measured boot chain beyond the kernel and into the application layer. It is responsible for measuring all executed binaries, loaded libraries, scripts, and configuration files, extending these measurements into designated TPM PCRs to provide a comprehensive and continuous record of the system's runtime state.

5.1.3 Firmware Integrity and Resilience

All device firmware and subsequent updates must be cryptographically signed by the manufacturer or another trusted authority. The device's bootloader shall validate this signature using a public key stored securely on the device before allowing the firmware to execute.

While raw public keys are permissible for signature validation, the 8ra ecosystem shall move toward the prevalent use of certificate-based secure boot. As detailed in contemporary research on key management for long-lived assets, certificates significantly improve the agility and scalability of rotating firmware signing keys – a critical capability for managing the security lifecycle of Industrial IoT (IIoT) and edge devices deployed for a decade or more.

Devices must implement robust anti-rollback protection mechanisms. This is typically achieved by enforcing a monotonic Security Version Number (SVN), which prevents an attacker from downgrading firmware to an older, vulnerable version. This requirement is a core principle of platform resiliency as outlined in NIST SP 800-193, Platform Firmware Resiliency Guidelines.

5.1.4 Cryptographic Identity and Key Binding

Device identity must be cryptographically bound to its hardware. The recommended standard for this is IEEE Std 802.1AR (DevID), wherein the device’s unique private key is generated, stored within, and protected by the TPM. This provides a strong, unforgeable identity rooted in silicon.

The cryptographic keys used for signing TPM Quotes (Attestation Keys, or AKs) must be distinct from the device's identity key (DevID). This functional separation ensures that the act of attesting to platform state is cryptographically distinct from proving device identity, preventing the misuse of a

device's core identity for routine attestation signing and ensuring the principle of least privilege is applied to cryptographic keys.

These technical mandates provide the verifiable foundation for security, which directly aligns with and supports formal compliance with European standardization efforts.

5.1.5 Standardization Baseline (EUCS Alignment)

A core strategic objective of the IPCEI-CIS is to align with key European standards to ensure interoperability, security, and digital sovereignty. Adherence to the architectural requirements detailed in the previous section is the primary mechanism for demonstrating compliance with the EUCS, particularly for services aiming to achieve the high assurance level. These mandates directly translate into the technical evidence required to satisfy EUCS security objectives.

Key Standards Alignment:

EUCS: Meets CS-EL4 mandates for Platform Integrity and Automated Compliance Monitoring.

NIST: Aligns with NIST SP 800-207 (Zero Trust Architecture) Section 3.1.2 regarding the inspection and assessment of asset state.

IETF: Adheres to IETF RATS (Remote Attestation Procedures) architecture.

The following table explicitly maps the CSF's architectural principles to the corresponding EUCS security objectives, clarifying how technical implementation delivers regulatory compliance.

EUCS Security Objective	Architectural Realization and Compliance Impact
Hardware Root of Trust	This objective is met by the mandate for TPM 2.0. The TPM provides the secure, immutable hardware anchor for all trust-related operations, including secure key storage, cryptographic measurements, and signed reporting.
Platform Integrity	This is achieved through the combined and mandatory implementation of Measured Boot, runtime IMA measurements, and cryptographically signed firmware with anti-rollback protection, creating a verifiable and unbroken chain of trust from power-on to runtime.
Trusted Execution & Strong Isolation	This is fulfilled by the requirement for TEEs in devices handling sensitive workloads. The TEE provides a hardware-enforced boundary that protects code and data from compromised host operating systems or hypervisors, ensuring confidentiality and integrity in use.

Achieving the EUCS 'High' assurance level requires not just the presence of these mechanisms, but verifiable evidence of their continuous monitoring and automated enforcement. The attestation and verification flows described in the subsequent sections are the means by which this evidence is

generated, collected, and appraised, thereby providing the demonstrable proof of compliance required for high-assurance certification.

5.1.6 Implementation Strategy

The security of the 8ra Cloud-Edge Continuum is a shared responsibility, built upon the specialized capabilities provided by ecosystem partners. The CSF does not seek to reinvent foundational technologies but rather to ensure that partner-provided building blocks integrate seamlessly to form a coherent, interoperable, and verifiable trust architecture. This collaborative approach leverages the strengths of the European technology sector to build a secure-by-design digital infrastructure.

The foundation of this security domain will be built upon components and devices provided by partners that adhere to the architectural requirements specified in section 5.1.2. This includes contributions from:

Silicon & Hardware Providers providing processors with integrated TPM 2.0 and TEE capabilities, which form the hardware anchor for platform trust.

Device Manufacturers and OEMs implementing Measured Boot, firmware signing, and anti-rollback protections in compliance with established standards such as NIST SP 800-193.

Network Equipment Vendors providing devices with strong, hardware-anchored identity mechanisms, such as IEEE 802.1AR DevIDs, to ensure the authenticity of network infrastructure components.

Higher-level integrity verification and continuous monitoring will be enabled by partners who provide the software and services necessary to operationalize this trust architecture at scale. **This includes solutions for:**

Centralized Attestation Services capable of challenging a large, heterogeneous fleet of devices and appraising the cryptographic evidence they provide.

Continuous Integrity Monitoring Tools that leverage the hardware roots of trust to provide ongoing assurance of device health and compliance. The telemetry and evidence appraised by these services are critical inputs for the Security Operations (SecOps) domain, feeding SIEM and SOAR platforms to enable ecosystem-wide threat detection and response.

These partners must support the standardized attestation flow detailed below, which is the core process for verifying device integrity across the ecosystem.

5.1.7 Attestation and Verification Flow (Sequence Description)

A standardized remote attestation flow is the core mechanism by which the 8ra ecosystem computationally verifies the integrity of a device before granting it access to resources. This process is a "challenge-response" protocol where a device (the Attester) must cryptographically prove its

current state to a trusted entity (the Verifier). This flow transforms platform integrity from a policy statement into a verifiable, machine-readable fact.

Based on the IETF Remote Attestation Procedures (RATS) architecture, the process involves three key roles:

Attester: The device whose integrity is being verified (the Target).

Verifier: The trusted service (the Appraiser) that challenges the Attester, receives evidence of its state, and appraises that evidence against established policies.

Reference Value Provider: The trusted source (e.g., the device manufacturer or software developer) that provides the "known-good" measurements (the golden configuration) in a signed, machine-readable format.

The attestation sequence proceeds as follows:

Challenge: The Verifier initiates the process by sending a unique, unpredictable challenge (a nonce) to the Attester. This nonce is critical for ensuring the freshness of the attestation report and preventing replay attacks, where an adversary reuses old, valid evidence.

Evidence Generation: Upon receiving the nonce, the Attester instructs its TPM to generate a Quote. A Quote is a data structure containing a selection of PCR values, the received nonce, and other platform state information. This entire structure is cryptographically signed by a private Attestation Key (AK), which is securely stored within and protected by the TPM. In parallel, the Attester compiles the detailed Attestation Log (e.g., the TCG Canonical Event Log) which contains the sequence of individual measurements that produced the PCR values in the Quote.

Evidence Transmission: The Attester sends the signed TPM Quote and the corresponding Attestation Log back to the Verifier as evidence of its current integrity state.

Appraisal (Verification): The Verifier receives the evidence and performs a series of cryptographic checks to appraise its validity:

First, it verifies the signature on the Quote using the Attester's public AK. This confirms that the evidence originated from a genuine TPM and has not been tampered with in transit.

Next, it checks that the nonce contained within the signed Quote matches the one it originally sent, proving the freshness of the report.

It then cryptographically replays the events from the Attestation Log, calculating the expected final PCR values. If the calculated values match the PCR values signed in the Quote, it proves that the log is authentic and accurately reflects the boot and runtime history of the device.

Finally, it compares the individual measurements from the now-verified log against a set of approved values contained in a Reference Integrity Manifest (RIM). This RIM is itself

cryptographically signed by a trusted Reference Value Provider, serving as the "golden configuration" for that device.

If all of these checks pass, the Verifier gains high cryptographic assurance that the device has booted and is running an exact, authorized software configuration. This successful appraisal is the prerequisite for the device to be granted access to IPCEI-CIS resources, forming a foundational control in a Zero Trust Architecture.

5.1.8 Partner Building Blocks

The 8ra ecosystem leverages specialized partner capabilities to deliver the architectural requirements defined in this section. The following table maps partner building blocks to their primary contributions within the Device Trust, Platform Integrity & Attestation domain.

Hardware Root of Trust & Secure Storage

Partner	Building Block	Contribution
SAP SE	KMS	Provides HSM-backed root key management, ensuring cryptographic keys are stored in secure hardware components.
SAP SE	Secrets Management	Delivers HSM integration for encryption and signing operations, including auto-unsealing via HSM.
secunet Security Networks AG	Secure Keymanagement	Implements hardware-assisted (DPU, SmartNIC) separation of key material from tenant and management workloads.
Atos Eviden	Zero-Touch Onboarding	Integrates with secure elements (SE) as a form of Hardware Root of Trust for device provisioning.
Oktawave	KMS	Researches Intel SGX and AMD SEV-SNP as foundations for virtual HSM (vHSM) implementations.

Trusted Execution Environments & Workload Isolation

Partner	Building Block	Contribution
Deutsche Telekom AG	Confidential Containers	Enables execution of Kubernetes Pods on edge-cloud infrastructure with enhanced security guarantees, preventing cloud operator manipulation or eavesdropping.
secunet Security Networks AG	Secure Tenant Separation	Develops a platform with minimal Trusted Computing Base (TCB) for strict, secure tenant separation through virtualization.
E-Group	Secure Processing Framework	Incorporates production-ready TEE solutions for secure workload processing.

Oktawave	KMS	Leverages Intel SGX and AMD SEV-SNP research to create a virtualization-proof key management system.

Measured Boot & Runtime Integrity

Partner	Building Block	Contribution
Deutsche Telekom AG	Zero Trust Networking for Kubernetes	Implements continuous verification of device integrity as a core ZTNA function, directly dependent on measured boot chains.
Atos Eviden	Zero-Touch Onboarding	Utilizes PKI for digital identity provisioning and integrates with secure elements that store the CRTM and perform PCR measurements.
Adva Network Security	Automated Trust and Secrets Management	Automates trust relationship establishment, requiring devices to demonstrably start in a trusted (measured boot) state.
Oktawave	KMS	Builds on Intel SGX and AMD SEV-SNP technologies that require measured launch and strict integrity checks.

Firmware Integrity & Crypto-Agility

Partner	Building Block	Contribution
SAP SE	Crypto Agility	Enables seamless switching between cryptographic algorithms, protocols, and implementations for long-term resilience.
Adva Network Security	Highly Secure L2 Network Access Device	Combines Post-Quantum Cryptography with crypto-agility for long-term security and anti-downgrade protection.
Atos Eviden	Zero-Touch Onboarding	Leverages PKI to deliver X.509 certificates for certificate-based secure boot and firmware signature validation.
Adva Network Security	Automated Trust and Secrets Management	Provides certificate management capabilities essential for firmware signing and bootloader validation.
secunet Security Networks AG	Secure Software Dev and Deploy Pipeline	Implements Software Bill of Materials (SBoM) concepts for firmware supply chain transparency.
E-Group	Quantum-safe Cryptography Layer	Integrates quantum-safe SSL/TLS components and develops quantum-resistant algorithms for long-term signature resilience.

Cryptographic Identity & Key Binding

Partner	Building Block	Contribution
Atos Eviden	Zero-Touch Onboarding	Establishes digital device identities (X.509 certificates) with secure element integration, implementing the DevID principle.
SAP SE	KMS	Manages encryption keys with HSM-backing, ensuring device keys are protected within immutable hardware anchors.
SAP SE	Secrets Management	Securely stores secrets and certificates with HSM integration, providing the foundation for unforgeable identity.
secunet Security Networks AG	Secure Keymanagement	Implements hardware-assisted separation of key material for functional key isolation.
Oktawave	KMS	Develops virtual HSM (vHSM) based on SGX and SEV-SNP for secure storage of device identity and attestation keys.

Remote Attestation (RATS Architecture Roles)

Partner	Building Block	RATS Role	Contribution
Deutsche Telekom AG	Zero Trust Networking for Kubernetes	Verifier	Implements continuous device integrity verification against established policies.
Adva Network Security	Automated Trust and Secrets Management	Verifier	Provides continuous monitoring and security compliance auditing of device state.
Atos Eviden	Zero-Touch Onboarding	Attester	Delivers cryptographically signed state data via secure element integration.
SAP SE	Zero Trust	Verifier	Supports Zero Trust Architecture by enabling trust boundary configuration and attestation-based access decisions.
secunet Security Networks AG	Secure Software Dev and Deploy Pipeline	Reference Value Provider	Provides Software Bill of Materials (SBoM) as known-good measurements in machine-readable format.
Oktawave	KMS	Attester	Researches TEE technologies (SGX/SEV-SNP) requiring measured launch and attestation for integrity proof.

5.2. Architectural Requirements

The following technical specifications are mandatory for establishing a secure and manageable device lifecycle within the IPCEI-CIS framework. These requirements are organized by lifecycle phase and are essential for mitigating risks from supply chain to end-of-life.

5.2.1 Secure Onboarding and Enrollment

Zero-Touch Provisioning (Greenfield): All new edge and IIoT devices must be onboarded using the FIDO Device Onboard (FDO) specification. This mitigates supply chain risk via cryptographic ownership transfer.

Legacy Onboarding (Brownfield): We acknowledge that the vast majority of existing devices do not support FDO. For these "Brownfield" devices, a transitional onboarding process shall be implemented. This process must place newly connected legacy devices into a quarantined network segment. Full network access is granted only after a successful (manual or agent-based) integrity scan and registration with the management platform.

Initial Integrity Verification: Upon first connection, all hardware must undergo a comprehensive firmware integrity scan against a Reference Integrity Manifest (RIM).

5.2.2 Policy-Based Configuration and Hardening

Unified Endpoint Management (UEM): All devices, regardless of function or location, must be enrolled in and managed by a central UEM or Mobile Device Management (MDM) platform. This platform will serve as the single, authoritative point for policy enforcement, configuration deployment, and continuous monitoring.

Configuration Baselines: All devices shall be hardened according to standardized security profiles.

Systems requiring EUCS High Assurance must be configured in alignment with the CIS Level 2 Profile, which provides a stringent security posture suitable for mission-critical infrastructure. For all other systems, the CIS Level 1 Profile is the minimum acceptable baseline.

Configuration Control & Drift Prevention: The integrity of device configurations must be continuously meuc onitored. Adhering to ANSI/EIA-649 principles, any modification to a eucschange control process. All proposed changes must be tracked, reviewed, approved, and logged. This shall be enforced via Policy-as-Code (PaC) during deployment and continuous run-time monitoring (SIEM/MDR) to automatically detect and remediate deviations (drift).

Principle of Least Functionality: Devices must be configured with either a deny-by-exception (application blacklisting) or an application whitelisting policy. This control prevents the execution of unauthorized software and strictly limits the device’s functionality to only what is essential for its intended purpose.

5.2.3 Secure Configuration Storage and Credential Management

Federated Configuration Database (CMDB): To address the scalability challenges of managing millions of dynamic edge devices, the CMDB architecture shall be federated. Instead of a single monolithic database, a hierarchical model must be used where local controllers manage their specific edge domains and report only aggregated compliance status (not raw telemetry) to the central IPCEI-CIS governance layer. This ensures privacy and operational scalability.

Secrets Management: All secrets – including API keys, private certificates, passwords, and other credentials – must not be hard-coded. They shall be managed and retrieved at runtime from a centralized secrets management solution.

Cryptographic Key and Certificate Lifecycle: To support long operational lifetimes, a certificate-based key rotation strategy is mandatory.

5.2.4 Credential Revocation

Revocation Mechanisms: The system shall support a robust and efficient credential revocation mechanism. In line with the move to a certificate-based trust model, revocation will be certificate-based. For resource-constrained devices, this can be implemented through simple but permanent hardware-level actions, such as a single-byte increment in a device's One-Time Programmable (OTP) memory to permanently retire past keys.

5.2.5 Secure Decommissioning

Data Sanitization: Upon retirement, all devices containing storage media must undergo a formal data sanitization process. This process shall use industry-standard methods, such as NIST SP 800-88 compliant wiping protocols, to ensure that all sensitive data is permanently and irrecoverably destroyed.

Asset Retirement Protocol: The decommissioning of any device must be accurately documented. This includes updating the device's status to "retired" in the CMDB, formally recording the method and date of data sanitization, and revoking all associated credentials and access rights.

5.2.6 Standardization Baseline

The architectural requirements detailed above map directly to specific control families within the EUCS framework (such as Asset Management [AM] and Operations Security [OPS]), providing a clear path to compliance. The following table links the key lifecycle requirements to their corresponding EUCS alignment areas.

Key Standards Alignment:

EUCS: OPS-02 (Configuration Management) and OPS-03 (Asset Management).

NIST: NIST SP 800-128 (Security-Focused Configuration Management).

IEC: ISA/IEC 62443-4-1 (Secure Product Development Lifecycle).

Architectural Requirement	EUCS Alignment Area
Secure Onboarding and Initial Integrity Verification	Secure Onboarding/Decommissioning
Policy-Based Configuration & Hardening	Secure Configuration, Hardening
Centralized CMDB and Asset Tracking	Asset Management
Secure Decommissioning and Data Sanitization	Secure Onboarding/Decommissioning

Adherence to these requirements and their underlying controls is fundamental to demonstrating the continuous compliance necessary to achieve and maintain EUCS High Assurance certification.

5.2.7 Implementation Strategy

The implementation of these requirements within the 8ra ecosystem will rely on the integration of specialized partner capabilities into a unified framework. While the security framework does not mandate a single vendor for every function, it defines the overarching process that unifies partner contributions. For example, physical security measures for far-edge cloud modules, provided by a specialized partner, will be complemented by centrally managed logical controls for configuration, patching, and secure zero-touch onboarding. This federated approach ensures that best-in-class solutions are integrated into a cohesive, end-to-end process that is consistently applied across the entire device lifecycle.

5.2.8 Sequence Description: Zero-Touch Onboarding Flow

The following sequence describes the technical workflow for a zero-touch device onboarding process using the FDO protocol. This flow enables devices to be provisioned securely and automatically without manual intervention.

Manufacturing & Ownership Transfer: The device is manufactured with a unique cryptographic identity. The manufacturer generates a corresponding 'Ownership Voucher' containing the public key and securely transfers this voucher to the target organization (the device owner).

Initial Power-On: The device is deployed in its intended operational environment and powered on for the first time. It contains no pre-programmed credentials, network-specific configurations, or management platform details.

Rendezvous Server Connection: Using its factory configuration, the device automatically connects to a publicly known FDO Rendezvous Server on the internet.

Ownership Redirection: The device presents its identity to the Rendezvous Server. The server validates the device against its database of Ownership Vouchers and securely redirects the device to the owner’s specific management platform (e.g., its UEM/MDM service).

Mutual Authentication & Trust Establishment: The device connects to the owner's management platform. The platform uses the Ownership Voucher to validate the device's authenticity, while the device validates the platform, establishing a mutually authenticated and encrypted communication channel.

Secure Provisioning: Through the secure channel, the management platform sends the device its initial configuration payload. This includes all necessary network settings, security policies (e.g., CIS baselines), software, and cryptographic certificates needed for operation.

Operational State: The device applies the received configuration, completes its formal enrollment in the UEM, and transitions into a fully managed, secure operational state. It is now ready to perform its designated function within the 8ra Cloud-Edge Continuum without requiring any manual setup or on-site technical intervention.

This automated onboarding process is a critical enabler for the secure and scalable management of devices throughout the 8ra ecosystem, providing a trusted foundation for subsequent security controls.

5.3 Architectural Requirements

The following architectural mandates define the non-negotiable standards for securing the software supply chain and update mechanisms.

5.3.1 Secure Update Pipeline & Code Signing

The integrity of the update process is a primary target for supply chain attacks. A compromised update mechanism can deliver malicious code to thousands of devices simultaneously.

Certificate-Based Signing: All firmware and software updates shall be signed using a certificate-based system. The use of raw public keys stored directly in device memory is deprecated in favor of a chain of trust model. This allows signing keys to be rotated by the provider's KMS without physical modification of the device, enabling crypto-agility.

Fail-Safe Installation: All devices, particularly in OT environments, must implement a fail-safe update mechanism (e.g., dual-bank firmware). This prevents devices from being rendered inoperable ("bricked") due to power loss or network failure during an update.

Anti-Downgrade Protection: All updatable components shall implement anti-downgrade mechanisms. A SVN, stored in tamper-proof non-volatile memory (e.g., OTP or TPM NVRAM), must be checked before applying updates. The device must reject any firmware image with an SVN lower than the currently stored value. For critical infrastructure, a "Delayed SVN Commit" policy shall be used, where the SVN is only updated after the new firmware successfully passes self-tests.

5.3.2 Supply Chain Transparency and Integrity

Verifiable transparency is a non-negotiable prerequisite for any hardware or software vendor in the 8ra ecosystem.

Software Bill of Materials (SBOM): All software acquisitions must be accompanied by a machine-readable SBOM compliant with CycloneDX or SPDX standards. This enables automated vulnerability correlation and license management.

Hardware Bill of Materials (HBOM): Hardware acquisitions must include an HBOM to identify sub-components from untrusted or high-risk suppliers.

Vulnerability Exploitability eXchange (VEX): Vendors shall provide VEX documents alongside their SBOMs to clarify the exploitability status of known vulnerabilities, reducing false positives in vulnerability reports.

Trusted Source: Third-party software must be sourced exclusively from a trusted, internal read-only registry after undergoing security verification (NIST SSDF PO.2, PS.1).

5.3.3 Vulnerability Management and Response

The strategic goal is to shrink the window of opportunity for adversaries by implementing aggressive, automated vulnerability management.

Continuous Scanning: All assets must be subject to continuous vulnerability scanning.

Risk-Based Prioritization: Remediation must be prioritized based on a correlation of vulnerability severity (CVSS), threat intelligence, and asset criticality from the CMDB.

Service Level Agreements (SLAs): Formal SLAs for patch deployment shall be enforced based on severity tiers (e.g., Critical, High).

Automated Patching: Automated patch management solutions must be used to orchestrate the lifecycle from detection to deployment for remote and hybrid endpoints.

5.3.4 Crypto-Agility and PQC Readiness

To address the threat of quantum computing (Shor's algorithm), the architecture must be designed for crypto-agility.

Key Rotation: Regular rotation of signing keys is mandatory. Certificate-based secure boot is the required architecture to facilitate this without hardware changes.

PQC Migration: The system must support the seamless replacement of cryptographic primitives (e.g., migrating from RSA/ECDSA to Dilithium or Kyber) via modular cryptographic libraries and protocol negotiation, ensuring long-term resilience.

5.3.5 Standardization Baseline (EUCS Alignment)

The architectural requirements defined in this chapter directly map to the controls mandated by the EUCS.

Key Standards Alignment:

Regulation: EU Cyber Resilience Act (Security by Design, Vulnerability Handling).

EUCS: VUL-01 (Vulnerability Management), CS-08 (Supply Chain Security), CFG (Configuration Management).

Framework: The Update Framework (TUF) for secure metadata and content delivery.

Architectural Requirement	Relevant EUCS Control Family
Vulnerability & Patch Management	VUL-01: Mandates processes for identifying and remediating vulnerabilities.
Secure Update Pipelines	SUP-02: Requires verifying integrity and provenance of updates.
Anti-Downgrade Protection	CFG: Aligns with NIST SP 800-171 ensuring configuration baselines.
Supply Chain Transparency (SBOM)	APP: Aligns with controls requiring transparency of components.
Crypto-Agility	CRY: Encompasses controls for robust key management and PQC preparation.

5.3.6 Implementation Strategy & Partner Building Blocks

The implementation of secure update pipelines and supply chain security relies on specialized partner capabilities.

Secure Update Pipeline & Code Signing:

Atos Eviden (Zero-Touch Onboarding): Delivers X.509 certificates for secure boot and signature validation.

Adva Network Security (Automated Trust): Provides certificate management for firmware signing key lifecycles.

SAP SE (Crypto Agility): Enables seamless algorithm transitions for update signing.

Supply Chain Transparency:

secunet (Secure Software Pipeline): Implements SBOM concepts for component verification and provenance tracking.

Crypto-Agility & Post-Quantum Readiness:

Adva Network Security: Combines Post-Quantum Cryptography with crypto-agility for network devices.

E-Group: Integrates quantum-safe SSL/TLS components.

5.3.7 Illustrative Sequence Flow: Certificate-Based Secure Firmware Update

Update Generation: Provider generates firmware, signs the hash with a private key (K2), and packages it with a public certificate (C2) and new SVN.

Notification & Retrieval: Edge device receives notification and retrieves the signed package via TLS.

Verification: Bootloader verifies the signature using the device's trust anchor and validates the certificate chain.

Anti-Downgrade Check: Device compares the update's SVN against the stored SVN in OTP memory. If New_SVN > Stored_SVN, it proceeds.

Installation: Firmware is written to the secondary memory bank (A/B update).

Activation & Attestation: Device reboots into the new firmware. Upon connection to the cloud, it performs remote attestation, providing a TPM Quote of the new PCR measurements to prove its updated, valid state.

5.4. Architectural Requirements

The strategic importance of codifying precise architectural requirements cannot be overstated. These mandates translate high-level security principles into concrete, verifiable engineering specifications for all IPCEI-CIS partners and solutions. Adherence to these requirements is not optional; it forms the baseline for ensuring interoperability, establishing a unified security posture, and building mutual trust across the entire 8ra ecosystem. All requirements specified herein are mandatory.

5.4.1 Data-at-Rest Encryption

Mandate Universal Encryption: All non-volatile storage on edge and IoT devices (e.g., flash, disk) that contains operational data, configuration data, or tenant workloads must be protected with strong, authenticated encryption.

Specify Cryptographic Standards: Encryption mechanisms shall utilize FIPS 140-3 validated cryptographic modules. For resource-constrained devices, as identified in IETF RFC 7228, implementations may use lightweight, high-performance ciphers. These shall include options like ChaCha20, which is optimized for high-speed software execution on low-power processors that may lack AES hardware acceleration, or AES-128 in a NIST-standardized mode such as Counter (CTR) mode, provided they meet the necessary performance and security constraints.

Enforce Data Sanitization at Decommissioning: Upon device retirement or decommissioning, a formal data sanitization process must be executed. This process shall use industry-standard methods, such as NIST wiping standards, to ensure that all sensitive data is permanently and irrecoverably erased. This practice is critical to preventing data breaches from legacy assets.

5.4.2 Hardware-Backed Key & Secret Storage

Require Hardware Root of Trust: All cryptographic keys and sensitive secrets, such as API tokens and device identity credentials, must be generated, stored, and managed within a hardware-backed keystore. This keystore must be anchored by an immutable hardware root of trust.

Specify Approved Technologies: This requirement shall be met using either a Trusted Platform Module (TPM) 2.0 compliant module or a certified Hardware Security Module (HSM).

Enforce Access Control Policies: Keys and secrets stored within the hardware root of trust must be governed by strict access control policies enforced by the hardware itself. At a minimum, these policies shall restrict key usage based on user authentication, defined cryptographic purposes (e.g., signing only), and the measured integrity state of the device, as recorded during the secure boot process.

5.4.3 Centralized Secrets Management and Lifecycle

Prohibit Hard-Coded Credentials: The practice of hard-coding secrets in source code or configuration files is strictly forbidden.

Mandate Secure Vaulting: All secrets must be managed through a centralized secrets management solution.

Offline Operation Support: We acknowledge that critical IIoT systems often operate with intermittent or no cloud connectivity. For these scenarios, devices shall support a "Cached Secrets" model. Secrets may be cached locally if and only if they are encrypted using a key protected by the device's hardware root of trust (TPM/DICE). This ensures availability without compromising security during offline periods.

Enforce Automated Key Rotation: A formal policy must be established for the regular rotation of keys.

5.4.4 Workload and Tenant Isolation

Mandate Confidential Computing for Sensitive Workloads: For all workloads that process sensitive or regulated data, all processing (data-in-use) must occur within a hardware-isolated TEE, also known as an enclave.

Explain the Isolation Guarantee: The TEE shall be implemented to protect the confidentiality and integrity of both code and data from all privileged software, including the host operating system, hypervisor, and any other workloads running on the device.

Specify Tenant Separation: In multi-tenant edge environments, strong logical isolation between different tenant workloads, data, and network traffic must be enforced. An example of a valid architectural pattern for achieving this is the use of distinct and separate Microsoft Entra directories to ensure that users and administrators of one tenant directory cannot access data in another.

These architectural mandates provide the technical foundation upon which a verifiable security posture can be built, directly aligning with key European standards.

5.4.5 Standardization Baseline: EUCS Alignment

The architectural requirements defined in this framework are not arbitrary. They are directly mapped to European regulatory standards to ensure a verifiable, certifiable, and unified security posture across the IPCEI-CIS. This section provides the direct alignment between the previously defined requirements and the controls within the EUCS, demonstrating our commitment to compliance and high assurance.

Key Standards Alignment:

EUCS: ENC-01 (Encryption) and KMS-01 (Key Management).

Standard: FIPS 140-3 (Security Requirements for Cryptographic Modules).

Regulation: GDPR (Data Protection by Design).

The table below maps the core architectural requirements for local data protection to their corresponding EUCS focus areas and the minimum assurance level they are designed to support.

Architectural Requirement	EUCS Focus Area	Implied EUCS Assurance Level
Data-at-Rest Encryption	Data-at-Rest Protection	High
Hardware-Backed Key Storage	Secure Key Storage	High
Workload Isolation via TEE	Tenant Isolation	High

Assurance Level Implications

The EUCS framework specifies three distinct assurance levels – Basic, Substantial, and High – which have direct implications for implementation and validation:

The Basic and Substantial assurance levels can typically be met by implementing foundational security configurations, such as those defined in the CIS Level 1 Profile. These controls establish a fundamental security posture with minimal impact on system performance.

Achieving the High assurance level is mandatory for services supporting critical infrastructure and managing highly sensitive data. This level must be supported by implementing stricter, more robust security guidelines, such as those defined in the CIS Level 2 Profile.

Crucially, the High assurance level also requires continuous, automated monitoring to prove the ongoing effectiveness of security controls. This mandate directly corresponds to EUCS controls like AM-01.6, which requires the automated monitoring of asset inventories, and OPS-05.3, which requires the automated monitoring of system configurations.

This alignment ensures that our technical architecture not only meets but can demonstrably prove its compliance with the highest levels of European cybersecurity standards, paving the way for practical implementation.

5.4.6 Implementation Strategy and Reference Architectures

This section provides concrete, actionable guidance on implementing the mandated security controls. While specific building blocks from IPCEI-CIS partners will integrate into these architectures, the following reference flows describe the standardized processes that all solutions must follow to ensure consistent security and interoperability. Partner-specific implementation details are not available in this version of the document.

KMS Partners: Provide centralized Key Management Systems that manage the lifecycle of certificates and push policies to local agents.

Device Software Partners: Implement "TPM-TSS" stacks (e.g., using tpm2-tss libraries) to interface the OS encryption layers (like LUKS/BitLocker) with the hardware.

5.4.7 Reference Flow: Device Secrets Retrieval

The following sequence describes the standardized process for an application on an edge device to securely retrieve a secret from a central vault.

Boot: Device powers on. Trusted Boot measures firmware/OS components into PCRs.

Request: The application requests a required secret from a local agent running on the device.

Attestation: The agent authenticates the application and its host environment. This process shall leverage the device's TPM to attest to the integrity state of the device by providing cryptographic

evidence (a TPM "quote," which is a digitally signed statement of the PCR values) of the PCRs captured during the Measured Boot process.

Secure Channel: Upon successful attestation, the local agent establishes a secure, mutually authenticated, and encrypted channel to the central secrets management service.

Authorization: The central service validates the device's identity and confirms its authorization to access the requested secret based on pre-defined policies. The secret is then securely transmitted to the agent.

Delivery: The secret is delivered directly to the application for in-memory use and is never written to persistent storage on the device, minimizing its exposure.

5.4.8 Reference Flow: Confidential AI Model Fine-Tuning

This high-level flow outlines the process for fine-tuning an AI model with a private dataset within a TEE, ensuring the confidentiality of both the model and the data.

A deployment order is created, specifying a base AI model and the private dataset that will be used for the fine-tuning process.

A confidential virtual machine, protected by a hardware-based TEE, is instantiated on a compliant edge node. An execution controller operating inside the TEE verifies the integrity and authenticity of the deployment order.

The TEE's trusted loader securely downloads the base model and the encrypted private dataset into the protected enclave.

An AI training engine, also running within the TEE, decrypts the private dataset and executes the fine-tuning workload. At no point during this process are the proprietary model or the plaintext private data exposed to the host operating system, the hypervisor, or the node operator.

Upon completion, the newly fine-tuned model layer is encrypted within the TEE and securely returned to the tenant's designated storage location.

These reference flows illustrate the practical application of our security requirements, turning architectural mandates into secure operational processes.

5.4.9 Partner Building Blocks

The following IPCEI-CIS partner building blocks provide concrete, field-tested capabilities that directly support the requirements of 5.4 Local Data Protection & Key Handling on Devices. Their expertise covers three core areas: per-tenant/per-customer encryption of data at rest, hardware-backed storage of secrets and cryptographic keys, and strong isolation of workloads and data-in-use on edge devices.

Data-at-Rest Encryption & Tenant-Specific Protection

Partner	Building Block	Contribution
SAP SE	KMS	Ensures that data at rest is encrypted individually per customer and supports HSM-backed root key management, including the disablement of root keys to prevent access to encrypted data, enabling secure wipe workflows.
secunet Security Networks AG	Block Encryption	Provides per-tenant encryption of data at rest with transparent encryption outside the client infrastructure and protects keys using suitable hardware mechanisms or physical separation.
secunet Security Networks AG	Secure Tenant Separation	Delivers strict, secure tenant separation through encryption of data in the network and data storage, combined with virtualization for workload isolation on edge platforms.

Hardware-Backed Secrets & Key Management

Partner	Building Block	Contribution
SAP SE	KMS	Implements HSM-backed root key management that fulfills hardware-protected key storage requirements and underpins secure key lifecycle management on edge devices.
SAP SE	Secrets Management	Provides secure storage for credentials, secrets, and API keys with planned HSM integration, aligning with TPM/TEE-backed secure storage requirements.
secunet Security Networks AG	Secure Keymanagement	Uses hardware-assisted (DPU, SmartNIC) separation of key material from tenant and management workloads to preserve key confidentiality on edge nodes.
Oktawave	KMS	Develops a key management system where keys are stored so that the cloud operator has no access, leveraging Intel SGX and AMD SEV-SNP as a basis for virtual HSM (vHSM) implementations.

Workload Isolation & Data-in-Use Protection on Edge Devices

Partner	Building Block	Contribution
secunet Security Networks AG	Secure Tenant Separation	Provides strict, secure tenant separation through encryption and virtualization, directly addressing isolation requirements between workloads and containers on edge devices.
Deutsche Telekom AG	Confidential Containers	Enables execution of Kubernetes Pods on remote edge-cloud infrastructure with strong security guarantees such that even the remote cloud operator cannot manipulate or eavesdrop on customer workloads.

Deutsche Telekom AG	Micro-segmentation for Kubernetes	Defines an architecture for dynamic micro-segmentation to isolate workloads based on their security posture, reducing lateral movement risk and protecting sensitive data during processing and transmission.
Oktawave	Processing of Encrypted Data (PoED)	Develops technology for processing encrypted data in the cloud without decryption, providing a beyond-state-of-the-art mechanism for protecting data in use on edge and cloud nodes.
E-Group	Secure Processing Framework	Considers the use of production-ready TEE solutions as the primary hardware-backed method for workload isolation and data-in-use protection.

5.5 Architectural Requirements

The following architectural mandates are applicable to all IPCEI-CIS endpoints. The keywords must and shall are used to delineate strict compliance requirements.

5.5.1 Host Security Baselines & Hardening

Baseline Enforcement: All endpoints, including servers, user devices, and network equipment, must be hardened according to a centrally managed and enforced security baseline covering the operating system, applications, and firmware.

EUCS Assurance Alignment: Systems designated for services requiring EUCS High Assurance must be configured to a baseline compliant with the CIS Level 2 Profile. All other systems shall adhere to a minimum baseline compliant with the CIS Level 1 Profile.

Least Functionality: The principle of least functionality must be enforced by disabling all services, applications, protocols, and ports not explicitly required for business operations.

Application Control: Endpoints must implement application control policies to prevent the execution of unauthorized or malicious software. This shall be achieved through application whitelisting (deny-by-default) where operationally feasible, particularly for EUCS High environments. As a minimum baseline, a deny-by-exception (blacklisting) policy must be enforced. This mitigates the risk of attackers using their own tools after gaining initial access, a common "living-off-the-land" technique.

Configuration Management: A formal CM process, consistent with standards such as ITIL or ISO 10007, must be implemented to ensure the integrity and consistency of all endpoints. This process is the procedural backbone that prevents the erosion of security posture over time. It shall encompass the four core disciplines of CM: Configuration Identification (defining and baselining all controlled components), Configuration Control (managing all changes through a formal review and approval workflow), Configuration Status Accounting (recording and reporting on the state of all components), and Configuration Verification & Audit (independently confirming that components

conform to their documented baselines). All changes to endpoint configurations must be tracked, reviewed, approved, and logged to prevent unauthorized "configuration drift."

5.5.2 Telemetry Collection and Logging

Mandatory Log Generation: All hosts and networking equipment must generate and collect security logs that cover the operating system, running services, and all applications.

Log Content: Log entries must contain, at a minimum, the User ID, event date and time, and terminal identity (e.g., hostname and/or IP address).

Structured Logging: The use of structured logging formats is mandatory to enhance parsing efficiency and enable precise filtering and redaction of sensitive data, thereby supporting GDPR compliance.

Log Retention: A unified log retention policy must be established. This policy shall adhere to the "highest common denominator" of all applicable regulations (e.g., 7 years for SOX), ensuring logs are available for forensic investigation and compliance audits.

Time Synchronization: All system clocks on all devices must be synchronized with an authoritative time source using a secure protocol on a periodic basis (at least once per day) to ensure the integrity and correlation of time-stamped audit records.

5.5.3 Health & Integrity Monitoring

Hardware-Anchored Trust: The integrity of all endpoints must be anchored in a hardware root of trust. Devices must utilize a Trusted Platform Module (TPM) to support secure boot and integrity measurement.

Continuous Integrity Verification: A chain of trust must be established from the immutable CRTM through the boot process (Measured Boot) and extended into the runtime environment. The IMA or an equivalent mechanism shall be used to continuously measure the integrity of executed binaries, scripts, and configuration files.

Security Agent Health: The health and status of security agents (e.g., Endpoint Detection and Response – EDR) must be continuously monitored. Key metrics, including fleet-wide agent coverage, data pipeline latency, and agent tampering alerts, must be reported to a central monitoring dashboard.

5.5.4 Threat Detection & Response

EDR/XDR (Tiered Requirement): An EDR solution must be deployed, but requirements are tiered by device class to address resource constraints:

Gateways & Servers: Must run a full EDR agent with real-time behavioral analysis.

Constrained Sensors (Low-Power): Instead of a heavy agent, these devices shall be monitored via Network-based Intrusion Detection Systems (N-IDS) at the gateway level and report lightweight "heartbeat" telemetry to minimize resource consumption.

SIEM Integration: All collected telemetry, logs, and security alerts must be forwarded to a central SIEM platform.

Standardized Log Formats: All audit logs must be transmitted using a standardized format, such as CEF or LEEF.

5.5.5 Standardization Baseline

The architectural requirements defined in this domain are not arbitrary; they are specifically designed to align with and satisfy key control families within the EUCS framework. The following table provides a direct mapping, demonstrating how IPCEI-CIS mandates translate to auditable EUCS compliance.

Key Standards Alignment:

EUCS: LOG-02 (Log Generation) and MON-01 (Monitoring).

Concept: Continuous Diagnostics and Mitigation (CDM).

IPCEI-CIS Requirement Area	EUCS Control Family Alignment	Rationale and Key Mandate
Host Security Baselines & Hardening	Host Security Baselines	Aligns with the EUCS mandate for standardized, hardened configurations (e.g., CIS Level 2) and rigorous change management to prevent security posture degradation.
Telemetry Collection and Logging	Logging & Monitoring (LOM)	Fulfills the explicit EUCS requirement to generate, protect, and retain comprehensive audit logs from all system components for forensic and compliance purposes.
Health & Integrity Monitoring	Event Collection (EVT)	Addresses the need for continuous, automated monitoring of security configurations (e.g., EUCS OPS-05.3) and agent health to guarantee the fidelity of collected telemetry.
Threat Detection & Response	Logging & Monitoring (LOM), Event Collection (EVT)	Implements the analytical framework (EDR/XDR/SIEM) required to process collected logs and events, enabling the detection of, and response to, security incidents.

5.5.6 Implementation Strategy

The implementation strategy for this domain relies on pervasive automation and centralized management to achieve and maintain a consistent security posture at scale across the heterogeneous Cloud-Edge Continuum.

Host Hardening will be implemented using a Policy-as-Code (PaC) methodology. Security baselines, such as the CIS Level 2 profile, will be defined as version-controlled configuration files. These baselines will be automatically applied during device provisioning and throughout the device lifecycle. This automated approach ensures that all endpoints – from powerful cloud servers to resource-constrained edge devices like the NVIDIA Jetson and ruggedized outdoor hardware – are deployed in a verifiably hardened state and prevents "configuration drift" by design.

Monitoring and Detection will be operationalized through a three-tiered data processing architecture:

Tier 1 (Endpoint): An EDR/XDR agent resides on each endpoint. It is responsible for collecting rich system telemetry, performing local analysis, and executing automated response actions, such as isolating a compromised host from the network. The health and operational status of this agent are continuously monitored to ensure the integrity of the data pipeline.

Tier 2 (Correlation): An XDR platform ingests and correlates telemetry from all endpoint agents with security signals from other domains, including network traffic, cloud workloads, and identity providers. This provides a unified view of complex, multi-stage attack chains, which significantly improves threat detection accuracy and reduces analyst workload.

Tier 3 (Aggregation & Compliance): A central SIEM solution serves as the definitive aggregation point for all security-relevant data. It ingests standardized CEF or LEEF logs from the XDR platform and all other infrastructure components. While XDR is optimized for real-time threat detection and response, the SIEM is the primary system for demonstrating EUCS compliance, facilitating long-term log retention, and enabling unrestricted threat hunting across the entire digital estate.

5.5.5 Conceptual Flow: Runtime Integrity Attestation and Secure Telemetry

The process of verifying an endpoint's integrity and the trustworthiness of its telemetry is anchored in hardware. The flow begins when a central Verifier, such as a component of the XDR platform, issues an integrity challenge to an Endpoint. In response, the Endpoint's TPM generates a cryptographically signed Quote over its PCRs. These PCRs contain an immutable, cryptographic record of all software components loaded during the Measured Boot process. This record is continuously updated at runtime by the IMA, which extends the chain of trust by measuring hashes of all executed code, scripts, and configuration files.

The Endpoint returns this signed TPM Quote along with the corresponding Attestation Log (formatted according to a standard like the Canonical Event Log Format) to the Verifier. The Verifier first validates the Quote's signature using the Endpoint's trusted identity. It then cryptographically replays the events in the Attestation Log to ensure the recalculated PCR values match those in the signed Quote, which proves the log has not been tampered with. Finally, the Verifier appraises these verified measurements against a known-good baseline, such as a Reference Integrity Manifest (RIM) provided by the trusted software authority (e.g., the device manufacturer or software publisher).

Concurrently, the EDR agent on the Endpoint securely streams its operational telemetry, such as process creations and network connections, to the SIEM/XDR platform. This is performed using a standardized format like CEF over a secure TLS channel. A successful integrity attestation provides high assurance that this telemetry is trustworthy and originates from a non-compromised host whose integrity has been cryptographically verified from the hardware up.

Having established a robust framework for continuous monitoring and host security, the next section will detail the requirements for securing the network communications between these trusted endpoints.

5.5.7 Partner Building Blocks

The following IPCEI-CIS partner building blocks provide concrete capabilities that directly support the requirements of 5.5 Endpoint Telemetry, Health Monitoring & Host Security Controls. Their expertise spans telemetry aggregation and SIEM/SOAR integration, distributed threat detection and EDR/XDR capabilities, and device posture reporting, host hardening, and compliance auditing.

Telemetry Aggregation & SIEM/SOAR Integration

Partner	Building Block	Contribution
SAP SE	Audit logging via OpenTelemetry	Extends OpenTelemetry to integrate audit logs into central SIEM platforms using standard formats, directly supporting the requirement to feed endpoint and application logs into SIEM/SOAR and establish a common telemetry baseline.
Engineering Ingegneria Informatica (ENG)	Integrated Security Monitoring & Analysis (ISMA)	SIEM and SOAR to proactively detect and respond to security incidents based on aggregated endpoint and network telemetry.

Threat Detection & EDR/XDR Capabilities

Partner	**Building Block**	**Contribution**
Engineering Ingegneria Informatica (ENG)	Specific Anomaly Detection & Response (SADR)	Provides core EDR, Extended Detection & Response (XDR), and Advanced Threat Protection (ATP) functionality for endpoints, directly addressing the EDR/XDR expectations defined in this domain.
Deutsche Telekom AG	AI-based threat detection	Develops distributed AI algorithms for real-time threat detection in edge data centers, enabling rapid identification of policy violations and emerging threats close to where data is generated.
FBK (Fondazione Bruno Kessler)	LUCID (Lightweight, Usable CNN in DDoS Detection)	Offers a deep-learning framework for cyber threat and anomaly detection in resource-constrained edge and IoT environments, providing an EDR/XDR-equivalent capability for constrained devices.

NBIP	Anomaly Detection	Analyzes network and application traffic logs to detect and classify anomalies, forming a foundational component of continuous detection and monitoring.
NBIP	Threat Intelligence	Operates a centralized monitoring and threat intelligence service that derives intelligent insights into the evolving threat landscape, improving resilience and deviation detection across the ecosystem.
FBK (Fondazione Bruno Kessler)	Decepto (cyber deception)	Generates realistic decoy assets in production environments, serving as an active host security control and effective mechanism for detecting intrusions and lateral movement.

Device Posture Reporting & Host Security Controls

Partner	Building Block	Contribution
Deutsche Telekom AG	Zero Trust Networking for Kubernetes	Defines a blueprint for continuous verification of device integrity, including secure boot and patch state, providing the basis for device posture reporting in Kubernetes-based environments.
Deutsche Telekom AG	Micro-segmentation for Kubernetes	Enables dynamic isolation of workloads through micro-segmentation, reducing the attack surface and acting as a host-centric network hardening and firewall control.
Adva Network Security	Automated Trust and Secrets Management	Includes continuous monitoring and security compliance auditing; the resulting security compliance scores indicate adherence to hardening and security requirements, fulfilling device posture reporting and compliance assessment needs.
FBK (Fondazione Bruno Kessler)	TLSAssistant	Performs vulnerability and configuration analysis of TLS deployments, identifying misconfigurations and verifying alignment with technical standards, thereby supporting verification of the hardened state of endpoints.

Collectively, these building blocks support the achievement of EUCS High assurance by enabling automated monitoring of assets, configurations, and security posture. They operationalize requirements such as OPS-21.3, OPS-05.3, IAM-03.11, and IAM-06.3/6, making telemetry, compliance auditing, and anomaly detection mandatory capabilities rather than optional enhancements.

5.6. Architectural Requirements

The following architectural requirements are mandatory for all edge and far-edge devices deployed within the IPCEI-CIS framework. To ensure economic viability without compromising security, these requirements are applied based on a Risk-Based Tiering Model:

Tier 1 (Critical Infrastructure & Unattended Far-Edge): Full compliance with ruggedization and active tamper detection.

Tier 2 (Standard Industrial Edge): Baseline compliance with environmental resilience and tamper-evident designs.

5.6.1 Tamper Protection and Detection

Hardware Root of Trust: All devices must incorporate an immutable hardware root of trust (TPM 2.0 or DICE for constrained devices).

Secure Boot and Measured Boot: The entire boot process shall be protected by a secure boot mechanism.

Physical Enclosures (Tier 1 Only): Critical devices deployed in unsupervised environments must be housed in secure, ruggedized enclosures specifically designed to provide protection against unauthorized physical access.

Intrusion Detection (Tier 1 Only): Critical devices must be equipped with active physical intrusion detection sensors (e.g., detecting enclosure breach, shock, or temperature anomalies). Tier 2 devices shall utilize passive tamper-evident seals.

Continuous Integrity Monitoring: The runtime integrity of the device's operating system and critical files must be continuously monitored via IMA.

5.6.2 Environmental Resilience

Standardized Durability Testing: Devices intended for outdoor deployment must be certified against MIL-STD-810H. Indoor IIoT devices (Tier 2) shall meet industrial IP ratings (e.g., IP65) appropriate for their specific operating environment.

Moisture and Corrosion Protection: Internal electronics shall be protected from moisture and corrosion.

Thermal Management: All devices must include a thermal management solution appropriate for their deployment environment.

Electromagnetic Interference (EMI) Protection: Devices shall incorporate EMI protection appropriate for their specific environment.

5.6.3 Secure Maintenance and Lifecycle Management

This section addresses the physical security implications of device lifecycle management. For detailed logical control requirements (Configuration Management, Provisioning, Updates), refer to Section 5.2 and Section 5.3.

Physical Protection during Maintenance: Maintenance procedures must not degrade the physical security posture of the device. Physical ports (e.g., USB, JTAG, UART) used for maintenance must be logically disabled by default and physically protected (e.g., behind locked panels). Access shall require strong authentication and, where possible, a cryptographic proof of authorization (e.g., a signed authorization token).

Physically Secure Provisioning: While Section 5.2.2.1 mandates the logical FDO protocol, the physical environment during provisioning must be controlled to prevent the injection of malicious hardware or pre-boot malware before the ownership transfer is complete.

Decommissioning & Destruction: When a device is decommissioned, in addition to the data sanitization requirements of Section 5.2.2.5, the physical destruction of the storage media s4hall be considered for high-security environments to strictly prevent any possibility of forensic data recovery.

These technical requirements are grounded in a solid foundation of European policy and international standards, ensuring a verifiable and compliant security posture.

5.6.4 Standardization Baseline

The architectural requirements outlined above are directly mapped to European regulations and established international standards. This alignment ensures a consistent, verifiable, and compliant security posture across all IPCEI-CIS deployments, facilitating interoperability and trust.

Key Standards Alignment:

EUCS: PHY-01 (Physical Security Perimeters) and PHY-02 (Physical Entry Controls).

Directive: CER Directive (Resilience of Critical Entities).

Requirement Category	Applicable Standard / Regulation
Physical Security & Resilience	EUCS Control Families: Physical Security, Environmental Resilience, Protection of Critical Assets.
Operational Availability	Critical Entities Resilience (CER) Directive: Mandates that critical entities must be able to prevent, resist, absorb, and recover from disruptive incidents, including natural hazards and sabotage.
Maintenance & Integrity	NIST SP 800-171 / SP 800-53: Maps to control families for Physical and Environmental Protection, Maintenance, and Configuration Management.
Cryptographic Modules	FIPS 140-3: Referenced for hardware security modules (TPMs, HSMs) used for key storage and cryptographic operations, which are integral to tamper protection.

By grounding our architecture in this baseline, we ensure that the solutions developed by project partners are not only technically robust but also demonstrably compliant with key European and international security frameworks.

5.6.5 Implementation Strategy: Partner Building Blocks

The IPCEI-CIS framework leverages specialized partner building blocks to implement the physical and environmental controls defined in this section. For 5.6 Physical & Environmental Protection for edge/far-edge devices, expertise is concentrated around physical hardening and tamper detection, environmental resilience and availability, and secure maintenance and decommissioning of edge modules.

Tamper Protection and Detection (Physical Hardening & Indicators)

Partner	Building Block	Contribution
Lindner BiGreen	BiGreen Advanced Physical Security for Far-Edge Cloud Modules	Provides purpose-built physical security for far-edge cloud modules, including ruggedized enclosures and tamper-evident construction tailored to decentralized, unattended environments.
Lindner BiGreen	Module Cryptographic Sealing Process	Defines a cryptographic sealing process that detects and evidences physical tampering attempts on modules, directly supporting tamper protection and integrity verification requirements.
Lindner BiGreen	Perfect Root of Trust	Establishes an immutable hardware trust anchor within the physical module, forming the basis for any tamper detection scheme by enabling the platform to cryptographically prove its integrity before it is trusted.

Environmental Resilience & Operational Continuity

Partner	Building Block	Contribution
Lindner BiGreen	BiGreen Advanced Physical Security for Far-Edge Cloud Modules	Ensures the physical integrity and protection of modules deployed in decentralized, unattended edge and far-edge locations, contributing directly to environmental resilience and long-term availability.
NBIP	Mitigation	Provides volumetric DDoS mitigation at network and application layers, protecting edge nodes against large-scale flooding attacks and supporting operational continuity under external availability threats.
Adva Network Security	Highly Secure L2 Network Access Device	Combines post-quantum cryptography with crypto-agility to guarantee long-term cryptographic resilience for network access and key exchange over the extended lifecycle of edge devices.

Secure Maintenance and Lifecycle Management (Secure Decommissioning)

Partner	Building Block	Contribution
SAP SE	KMS	Enables disablement of root keys to permanently prevent access to encrypted data, providing the foundation for crypto-erase and secure wipe during decommissioning.
SAP SE	Secrets Management	Manages the lifecycle of secrets, including credential renewal and revocation, which is critical for revoking access before device retirement or transfer.
secunet Security Networks AG	Secure Keymanagement	Addresses the handling of compromised keys and key lifecycle governance, a critical element of secure maintenance and revocation processes.
Lindner BiGreen	Module Cryptographic Sealing Process	Supports controlled on-site maintenance by ensuring that any physical intervention on the module can be detected and that integrity can be re-verified after servicing.

Collectively, these building blocks operationalize the physical, environmental, and lifecycle controls required to achieve EUCS High assurance for edge and far-edge deployments, complementing the logical controls from other domains with verifiable hardware and facility-level protections.

5.6.6 Reference Flow: Physical Tamper Detection and Response

The following sequence describes the high-level operational flow for detecting and responding to a physical tampering event or a critical runtime integrity failure on an edge device. This automated process ensures a swift, consistent, and verifiable response to threats at the physical layer.

Trigger Event: An event is triggered by either a physical sensor detecting a breach (e.g., an enclosure door is opened) or a software-based integrity check failing (e.g., the IMA detects an unauthorized modification to a critical system file).

State Change & Measurement: The device's firmware and operating system immediately react to the trigger. The event is cryptographically measured, and the hash is extended into the appropriate TPM PCR, permanently altering the device's integrity state. The device may also enter a pre-defined locked-down or minimal-functionality state to prevent further compromise.

Alert and Attestation: The device establishes a secure connection to a designated management or Security Operations (SecOps) platform. It generates and transmits a TPM Quote, which is a digitally signed attestation of its current PCR values. This quote is sent along with the device's integrity measurement logs and data describing the specific trigger event (e.g., sensor alert type and timestamp).

Remote Verification (Appraisal): The remote platform, acting as a Verifier, performs a series of checks. First, it validates the device's identity using its hardware-backed certificate (e.g., an IEEE 802.1AR DevID). Second, it verifies the digital signature on the TPM Quote. Finally, it appraises the

integrity state by reconstructing the PCR values from the provided logs and appraising them against the established 'known-good' baseline to verify the integrity deviation.

Incident Response: Upon successful verification of the tamper event, the platform automatically triggers a predefined incident response playbook. This response shall be orchestrated via automated playbooks and must include the following minimum actions:

Logging the incident in the SIEM for correlation and auditing.

Generating a high-priority alert for security personnel.

Revoking the device's network credentials and access tokens.

Isolating the device from the network to prevent any potential lateral movement.

Technical details topic 6

to be added in a future version of the document

Technical details topic 7

to be added in a future version of the document

Technical details topic 8

to be added in a future version of the document

Technical details topic 9

to be added in a future version of the document

Technical details topic 10

to be added in a future version of the document

Standards

Standardization

National and international security standards can help operators protect their critical infrastructures by defining state-of-the-art cybersecurity measures and structured risk management in the ISMS. The regulation for e.g. KRITIS and also NIS2 does not stipulate any exact standards that operators must implement. The use of industry standards generally facilitates the implementation and verification of the state of the art. The following table provides an overview of standardization and its areas of application:

Standard	Area of application
ISO 27001:2022	Data security
ISO 27011	Telecommunication
ISO 27017 / BSI C5	Cloud
ISO 27018 / BSI C5	Data privacy in cloud environments
ISO 27019	Energy
IEC 62443	Industrial facilities
ETSI *	Telecommunication, E-Health, Data security, AI security