Defensible10

ISAUnited
INSTITUTE OF SECURITY ARCHITECTURE UNITED

Defensible 10

Annex C (Normative): D03-Compute, Platform & Workload Security Architecture

Technical Standard

Standards Committee
12-5-2025

About ISAUnited

The Institute of Security Architecture United is the first dedicated Standards Development Organization (SDO) focused exclusively on cybersecurity architecture and engineering through security-by-design. As an international support institute, ISAUnited helps individuals and enterprises unlock the full potential of technology by promoting best practices and fostering innovation in security.

Technology drives progress; security enables it. ISAUnited equips practitioners and organizations across cybersecurity, IT operations, cloud/platform engineering, software development, data/AI, and product/operations with vendor-agnostic standards, education, credentials, and a peer community—turning good practice into engineered, testable outcomes in real environments.

Headquartered in the United States, ISAUnited is committed to promoting a global presence and delivering programs that emphasize collaboration, clarity, and actionable solutions to today's and tomorrow's security challenges. With a focus on security by design, the institute champions the integration of security at every stage of architectural and engineering practice, ensuring robust, resilient, and defensible systems for organizations worldwide.

Disclaimer

ISAUnited publishes the ISAUnited Defensible 10 Standards Technical Guide to provide information and education on security architecture and engineering practices. While efforts have been made to ensure accuracy and reliability, the content is provided “as is,” without any express or implied warranties. This guide is for informational purposes only and does not constitute legal, regulatory, compliance, or professional advice. Consult qualified professionals before making decisions.

Limitation of Liability

ISAUnited - and its authors, contributors, and affiliates - shall not be liable for any direct, indirect, incidental, consequential, special, exemplary, or punitive damages arising from the use of, inability to use, or reliance on this guide, including any errors or omissions.

Operational Safety Notice

Implementing security controls can affect system behavior and availability. First, validate changes in non-production, use change control, and ensure rollback plans are in place.

Third-Party References

This guide may reference third-party frameworks, websites, or resources. ISAUnited does not endorse and is not responsible for the content, products, or services of third parties. Access is at the reader’s own risk.

Use of Normative Terms (“Shall,” “Should,” “Must”)

- Must / Shall: A mandatory requirement for conformance to the standard.
- Must Not / Shall Not: A prohibition; implementations claiming conformance shall not perform the stated action.
- Should: A strong recommendation; valid reasons may exist to deviate in particular circumstances, but the full implications must be understood and documented.

Acceptance of Terms

By using this guide, readers acknowledge and agree to the terms in this disclaimer. If you disagree, refrain from using the information provided.

For more information, please visit our Terms and Conditions page.

License & Use Permissions

Standards Development & Governance Notice

This standard is one of the ten Parent Standards in the Defensible 10 Standards (D10S) series. Each Parent Standard is governed by ISAUnited's Standards Committee, peer-reviewed by the ISAUnited Technical Fellow Society, and maintained in the Defensible 10 Standards GitHub repository for transparency and version control.

Contributions & Collaboration

ISAUnited maintains a public GitHub repository for standards development. Practitioners may view and clone materials, but contributions require:

- ISAUnited registration and vetting
- Approved Contributor ID
- Valid GitHub username

All Sub-Standard contributions must follow the Defensible Standards Submission Schema (D-SSF) and are peer-reviewed by the Technical Fellow Society during the annual Open Season.

Abstract

The ISAUnited Defensible 10 Standards provide a structured, engineering-grade framework for implementing robust and measurable cybersecurity architecture and engineering practices. The guide outlines the frameworks, principles, methods, and technical specifications required to design, build, verify, and operate reliable systems.

Developed under the ISAUnited methodology, the standards align with modern enterprise realities and integrate Security by Design, continuous technical validation, and resilience-based engineering to address emerging threats. The guide is written for security architects and engineers, IT and platform practitioners, software and product teams, governance and risk professionals, and technical decision-makers seeking a defensible approach that is testable, auditable, and scalable.

This document includes a series of Practitioner Guidance, Cybersecurity Students & Early-Career Guidance, and Quick Win Playbook callouts.	
	**Practitioner Guidance-** Actionable steps and patterns to apply the technical standards in real environments.
	**Cybersecurity Student & Early-Career Guidance-** Compact, hands-on activities that turn each section's ideas into a small, verifiable artifact.
	**Quick Win Playbook-** Immediate, evidence-driven actions that improve posture now while reinforcing good engineering discipline.

Together, these elements help organizations translate intent into engineered outcomes and sustain long-term protection and operational integrity.

Foreword

Message from ISAUnited Leadership

Cybersecurity is at a turning point. As digital systems scale, reactive and checklist-driven practices do not keep pace with adversaries. The ISAUnited position is clear: security must be practiced as engineered design, grounded in scientific principles, structured methods, and defensible evidence. Our mission is to professionalize cybersecurity architecture and engineering with standards that are actionable, testable, and auditable.

ISAUnited Defensible 10 Standards: First Edition is a practical framework for that shift. The standards in this book are not theoretical. They translate intent into measurable specifications, controls, and verification, and enable teams to design and operate resilient systems at enterprise scale.

About This First Edition

This edition publishes 10 Parent Standards, one for each core domain of security architecture and engineering. Sub-standards will follow in subsequent editions, contributed by ISAUnited members and reviewed by our Technical Fellow Society, to provide focused, technology-aligned detail. Adopting the Parent Standards now positions organizations for seamless integration of Sub Standards as they are released on the ISAUnited annual update cycle.

Why “Defensible Standards”

Defensible means the work can withstand technical, operational, and adversarial scrutiny. These standards are designed to be demonstrated with evidence, featuring clear architecture, measurable specifications, and verification, so that practitioners can confidently stand behind their designs.

Contents

Annex C (Normative): D03-Compute, Platform & Workload Security Architecture

ISAUnited's Defensible 10 Standards
Parent Standard: D03-Compute, Platform, & Workload Security Architecture
Document: ISAU-DS-CPW-1000
Last Revision Date: December 2025
Peer-Reviewed By: ISAUnited Technical Fellow Society
Approved By: ISAUnited Standards Committee

Section 1. Standard Introduction

The Compute, Platform & Workload Security Architecture (ISAU-DS-CPW-1000) establishes the engineering baseline for securing the compute plane end to end: platform control planes, hypervisors and orchestrators, host operating systems, virtual machines, containers and their registries, serverless functions, and the automation that provisions and operates them. As a Parent Standard, it defines common terminology, scope, requirements (inputs), technical specifications (outputs), and verification and validation expectations that all sub-standards inherit. The standard remains vendor-neutral and implementation-agnostic, aligns with recognized foundational frameworks (NIST, ISO/IEC), and provides normative, testable specifications. The goal is a defensible, measurable, and auditable posture for platform and workload security across on-premises, cloud, and hybrid environments.

Objective

The objective of ISAU-DS-CPW-1000 is to secure the entire compute stack—from platform control planes and host operating systems to virtual machines, containers, serverless functions, and cloud-native infrastructure components—against evolving threats. It equips cybersecurity architects and engineers with a structured, defensible way to engineer security into compute and platform environments that support enterprise applications, back-end services, and orchestrated workloads.

Emphasis is placed on:

- Hardening control planes, hypervisors, orchestrators, and workloads against compromise.
- Implementing runtime controls that detect, contain, and respond to threats across the stack.
- Enforcing segmentation and Zero Trust between platform components, workloads, and external services.
- Instrumenting hosts, control planes, and workloads to produce actionable telemetry for continuous monitoring, forensic readiness, and automated response.
- Automating threat detection, policy enforcement, and compliance validation through infrastructure as code and policy as code.
- Attesting supply-chain integrity, image provenance, and artifact trust so unverified components cannot enter production.

By integrating these engineering-focused capabilities, the standard provides a measurable, defensible framework for securing compute, platform, and workload environments across hybrid, cloud-native, and on-premises architectures.

Justification

Enterprise compute and platform ecosystems now span virtualization, containers, serverless computing, orchestration platforms, and cloud-native control planes. These capabilities deliver agility and scale, yet they expand the attack surface beyond what perimeter-focused or compliance-only approaches address. The distributed, dynamic, and ephemeral nature of modern workloads, coupled with the criticality of platform control planes, demands engineering-grade controls.

Adversaries exploit hypervisor and orchestrator flaws, attempt container breakouts and privilege escalation, abuse control-plane APIs, and introduce malicious artifacts through compromised supply chains. Weak workload or platform identity, poor segmentation between control planes and workloads, and insufficient runtime protections create gaps that can be exploited. Misconfigurations in orchestration policy, secrets management, and service-to-service trust boundaries remain leading causes of breaches.
The velocity of DevOps and platform engineering further amplifies risk. Secure provisioning, configuration enforcement, and validation must be automated and embedded from the platform layer down to individual workloads. This requires a shift to engineering-led, vendor-neutral practices that enforce controls-as-code and continuously validate them.

ISAU-DS-CPW-1000 addresses these realities by combining platform hardening, workload security, identity and access control, runtime detection and response, telemetry generation, and supply-chain validation into a cohesive architecture. Foundational alignment with NIST and ISO/IEC is maintained at the Parent level, while detailed control mappings to CSA CCM, CIS, and OWASP appear in Section 9 (Security Controls) and in sub-standards. With structured requirements, measurable outputs, and rigorous validation, the standard enables practitioners to reduce exploitability, prevent unauthorized access, and control configuration drift across hybrid, cloud-native, and on-premises deployments.

Section 2. Definitions

Admission — The control-plane step where a workload specification (for example, pod, function, VM template) is evaluated before it is allowed to run.

Admission Controller — A policy gate in the platform control plane (for example, Kubernetes) that validates or mutates workload specifications at admission time to enforce security and compliance before runtime.

API Gateway — A control surface that mediates API traffic and enforces authentication, authorization, rate limiting, schema validation, and logging for platform/administrative and workload endpoints.

Artifact Integrity — The property that deployable items (images, functions, packages, VM templates) are unchanged from a verified publisher and build process; enforced through signing and attestation.

Artifact Signing — Applying a cryptographic signature to a build artifact (image, package, function) to prove integrity and publisher identity.

Attestation — Cryptographically bound metadata (for example, build provenance, SBOM digest, policy results) asserting how and by whom an artifact was produced.

BAS (Breach and Attack Simulation) — Automated or semi-automated execution of adversary-inspired techniques to continuously test whether controls (segmentation, admission/verify-before-start, detection/response) work as intended in production-like conditions.

Bastion (Bastion Host) — A hardened administrative access point that mediates privileged connections to control planes or hosts and enforces MFA, JIT elevation, and session recording.

Behavioral Analytics — Machine-learning or statistical techniques that model normal platform/workload behavior to detect meaningful deviations indicative of threats.

Container Runtime Security — Controls applied while containers/pods execute, including image integrity verification, syscall/process constraints, least-privilege runtime configuration, network policy, and continuous behavior monitoring.

Control Plane (Platform Control Plane Security) — Orchestration and management layers (for example, hypervisors, Kubernetes API/etcd, serverless control plane, management consoles) protected against unauthorized access, misconfiguration, and exploitation.

Control-Plane Audit (Audit Logging) — Authoritative logging of control-plane administrative and API activity for investigation, correlation, and evidence.

CSPM (Cloud Security Posture Management) — Continuous assessment of cloud services and configurations for misconfigurations and policy violations; in CPW, paired with WSPM to cover platform/workload posture.

Data Plane (Workload/Data Path) — The execution and traffic path for workloads and services; subject to segmentation, identity-based policy, and transport encryption.

Default-Deny — A policy stance where connections or actions are denied unless explicitly allowed by rule (applies to network, admission, or API gateways).

Drift (Configuration Drift) — Divergence between the intended, version-controlled configuration and the running state; must be detected, reconciled, and evidenced.
East–West Traffic — Lateral traffic between workloads or platform components within a trust zone or data center/cloud; subject to micro-segmentation and identity-based policy.

Egress Allowlist — An explicit set of permitted outbound destinations or services for a workload, namespace, or zone; all other outbound traffic is denied.

Evidence Pack (EP) — A curated, immutable collection of artifacts (policies, logs, scans, attestations, reports) tied to a specific requirement or test. For this annex, EP-03 is the Parent-level pack; sub-items use dot suffixes (for example, EP-03.6).

Golden Image (Approved Base Image) — A pre-approved, hardened, and signed base image (host OS, container base, VM template) tracked with SBOM and patch cadence.

Immutable Infrastructure — An operating model where compute instances and images are replaced rather than modified in place, reducing drift and improving auditability.

Image Provenance — Cryptographic evidence (signing plus provenance/attestation) demonstrating an image's source, build process, and integrity.

Image Registry — A repository that stores and distributes signed images for containers or VMs and enforces trust policy (for example, allowed publishers, signature/attestation, immutable tags).

Infrastructure as Code (IaC) — Managing, provisioning, and configuring compute, platform, and workload resources via code under version control to ensure repeatability and enforce baselines.

Interface Control Document (ICD) — A structured specification that defines an interface's contracts: authentication/authorization model, identity type, data classification, rate/flow limits, error handling, telemetry, and security invariants.

JIT (Just-in-Time) Access — Time-bounded elevation of privilege with approval and session capture to minimize standing administrator access.

KEV (Known Exploited Vulnerabilities) — Catalogue of vulnerabilities known to be exploited in the wild; used to drive mandatory gating and remediation.

Key Management Service (KMS) — Centralized cryptographic key management that enforces access controls, automated rotation, auditability, and cryptographic agility for platform/workload use.

mTLS (Mutual TLS) — A transport security mode where both client and server present and verify certificates to authenticate and encrypt service-to-service or administrative paths.

MTTD (Mean Time to Detect) — The average time to detect a security-relevant event or condition.

MTTC (Mean Time to Contain) — The average time to contain or isolate an identified incident or policy violation.

MTTR (Mean Time to Recover) — The average time to restore a service or workload to acceptable operating conditions after an incident, rollback, or remediation.

Namespace (Platform Namespace/Scope) — A logical isolation boundary for workloads and policies (for example, Kubernetes namespace) used to apply default-deny, egress allowlists, and per-team controls.

Network Policy (Namespace/Container Network Policy) — Declarative rules that govern allowed ingress/egress between pods/services within or across namespaces; used to implement default-deny and allowlists.

PaC (Policy as Code) — Defining and enforcing security and compliance policies as code so validation occurs automatically in pipelines and at admission/runtime.

PKI (Public Key Infrastructure) — The certificate issuance and trust system used to manage identities and mTLS within platforms.

Privileged Access Enforcement — Restricting and monitoring elevated operations through least-privilege role design, JIT elevation, MFA, and session recording.

Quarantine (Security Quarantine) — Automated isolation of a workload, artifact, or registry namespace upon policy violation or high-severity finding, pending rollback or remediation.

RBAC / ABAC — Role-Based Access Control and Attribute-Based Access Control; authorization models used to enforce least privilege for humans, services, and workloads.

Rollback (Security Rollback) — Automated restoration to a last-known-good signed image or configuration after a failed gate, policy violation, or incident.

Runtime Protection — Active controls during workload or platform execution that detect anomalies, prevent exploitation, and contain threats (for example, isolate, kill, or restart).

Secrets Management — Secure storage, rotation, scoped access, and auditing of credentials, tokens, and keys used by workloads and platform services; secrets are never embedded in code or images.

Serverless Function (FaaS) — A managed compute substrate where functions execute under platform control; in CPW, subject to admission/verify-before-deploy, identity scoping, private networking, and egress allowlists.

Service Mesh — A communication fabric that provides mTLS, workload identity, policy, and telemetry for service-to-service traffic.

SIEM / XDR / SOAR — Security Information and Event Management; Extended Detection and Response; Security Orchestration, Automation, and Response—platforms used for correlation, detection, and automated response.

SLO (Service Level Objective) — A target value or range for a service metric used to guide promotion/rollback decisions and operational gates (for example, error rate, latency, policy-denial rate).

Software Bill of Materials (SBOM) — A manifest of packages and components in an artifact (image, VM, function) used for provenance, license, and vulnerability analysis.

Threat-Model Delta — A concise PR-level note describing how a proposed change affects the system's threat model (new trust boundary, interface, dependency), plus how the change will be tested and evidenced.

Traffic-Contract Test — A positive/negative test that proves a declared service-to-service path is permitted while unauthorized paths are blocked, with results captured as evidence.

Verify-before-Start — A pre-runtime enforcement in which the platform validates signatures, attestations, provenance, and policy conformance immediately before the workload starts; non-compliant artifacts are denied.

Verify-on-Pull — A deployment-time enforcement that rejects artifacts lacking valid signatures/attestations or failing policy checks when they are retrieved from a registry.

Workload Detection and Response (WDR) — Continuous monitoring and protection for workloads (VMs, containers, serverless) using behavioral analytics and runtime enforcement to detect, mitigate, and respond to threats in real time.

Workload Identity — Short-lived, cryptographically verifiable identities assigned to workloads/services for authenticated, authorized, and encrypted interactions within and across platform boundaries.

Workload Security Posture Management (WSPM) — Continuous assessment and enforcement of configuration baselines and controls for workloads and platforms to prevent drift and misconfigurations.

WORM (Write Once, Read Many) Storage — An immutable storage mode used to preserve logs, audits, and evidence so records cannot be altered or deleted within the retention window.

Zero Trust Platform & Workload Architecture (ZTPWA) — An architectural pattern that applies Zero Trust to platform control planes and workloads, enforcing continuous identity verification and policy at every hop.

Zero Trust Platform & Workload Security — A security approach that requires continuous authentication, authorization, and validation for all entities (human, service, workload, platform component) with no implicit trust.

Section 3. Scope

Compute, platform, and workload environments now operate across highly dynamic, distributed, and interconnected infrastructures. ISAU-DS-CPW-1000 covers the full compute stack—from platform control planes and host operating systems to virtual machines (VMs), containers and their orchestrators, serverless functions, and orchestrated workloads—deployed in on-premises, cloud, and hybrid environments, including single-cloud, multi-cloud, hybrid-cloud, and traditional data-center architectures.

The standard sets architectural expectations, engineering practices, and technical guardrails required to achieve measurable resilience and defensibility across compute and platform ecosystems. It enables practitioners to anticipate and mitigate misconfiguration, enforce identity and trust boundaries, validate supply-chain integrity, and counter evolving threats, while advancing automation, immutable infrastructure, and cloud-native operations.

Applicability

- **All Compute and Platform Service Models:** Applies to workloads, platforms, and services implemented as VMs, containers, serverless functions, hypervisors, container-orchestration platforms, and managed compute services.
- **Enterprise, Government, and Academic Environments:** Intended for security, platform, and infrastructure teams across public and private sectors, research institutions, and organizations advancing compute and platform security practices.

- **Hybrid, Multi-Cloud, and On-Premises Architectures:** Addresses the security challenges of integrating and protecting workloads and platforms across diverse providers, data centers, and deployment models.

Key Focus Areas

- **Platform Control-Plane Security:** Protections for orchestration layers, hypervisors, and management APIs to prevent unauthorized access, abuse, and misconfiguration.
- **Identity and Access Controls:** Mechanisms to secure workload, service, and platform identities; enforce least privilege; and apply Zero Trust across all layers.
- **Workload and Platform Segmentation:** Isolation strategies, micro-segmentation, and trust-boundary enforcement to limit lateral movement and blast radius.
- **Compute- and Platform-Native Security Models:** Secure DevOps practices, immutable infrastructure, and automation to maintain security at scale.
- **Supply-Chain Integrity and Image Provenance:** Deployment of only verified, trusted artifacts, with controls that detect and block unverified or malicious components.
- **Encryption, Data Protection, and API Security:** Robust cryptographic controls, secure data-lifecycle management, and resilient API protection.
- **Continuous Monitoring, Telemetry, and Incident Response:** Real-time observability, automated detection, and rapid response tailored to platform and workload contexts.

Outcomes

By defining this scope, the standard ensures that compute, platform, and workload security architectures are:

- **Defensible:** Built with enforceable boundaries, engineered controls, and measurable security assurances.
- **Measurable:** Anchored to outputs that can be validated through continuous assessment and testing.
- **Adaptive:** Designed to evolve with technology advances, threat landscapes, and operational requirements.
- **Aligned:** Consistent with organizational policy, regulatory mandates, and industry practices for platform and workload security.

This scope establishes the foundation for resilient, secure, and defensible compute, platform, and workload environments that protect critical assets, maintain operational integrity, and enable organizational agility.

Section 4. Use Case

A robust compute, platform, and workload security standard must demonstrate practical applicability in complex, real-world environments. The following consolidated use case presents a technical scenario typical of modern enterprises operating across hybrid and multi-cloud architectures. It highlights common architectural weaknesses, maps them to targeted technical solutions based on Zero Trust Platform & Workload Architecture (ZTPWA), and defines measurable outcomes. This integrated approach ensures that engineering teams can directly align actions with defensible security objectives.

Table C-1:

Use Case Name	**Securing Hybrid & Multi-Cloud Compute, Platform, and Workload Environments with Zero Trust Platform & Workload Architecture**
Objective	Implement ZTPWA to protect platform control planes, workloads, and associated services from unauthorized access, misconfigurations, and lateral movement attacks.
Scenario	A global enterprise operates across on-premises data centers, AWS, Azure, and Google Cloud. The organization experiences inconsistent platform and workload access policies, frequent misconfigurations, and limited visibility across environments. Security teams detect privilege escalation attempts in control planes, unverified images entering production, and unmonitored lateral movement between workloads.
Actors	Platform Security Architect, Compute Engineer, IAM & Access Governance Team, DevOps/Platform Engineering Team, SOC Analysts
Challenges Identified	• Overprivileged Access: Excessive permissions for platform and service accounts; misconfigured IAM roles allowing broad access to control planes and workloads. • Misconfigurations & Vulnerable Artifacts: Inconsistent baselines across VMs, containers, and orchestrators; deployment of unverified or outdated images. • Insufficient Segmentation: Flat trust boundaries between workloads, platforms, and network segments permit excessive lateral movement. • Telemetry Gaps: Fragmented logging from hosts, control planes, and workloads, with no unified monitoring or correlation capability.
Technical Solution	• Identity & Access Hardening: Enforce RBAC/ABAC at platform and workload levels; require JIT elevation and MFA for privileged operations.

	• Posture & Configuration Management: Deploy automated, code-based policy enforcement for baseline compliance across all compute and platform layers. • Micro-Segmentation & Zero Trust Enforcement: Apply software-defined perimeters, mutual workload/platform authentication, and strict • east-west traffic controls. • Supply Chain Integrity Controls: Require SBOM verification, signed images, and artifact provenance checks before deployment. • Centralized Telemetry & Threat Detection: Aggregate platform, workload, and network telemetry into a unified SIEM/XDR pipeline with automated response playbooks.
Expected Outcome	• Reduction of platform and workload attack surface by 60% through least-privilege and JIT access controls. • 75% reduction in misconfiguration exposure via automated detection and remediation. • Segmentation prevents unauthorized lateral movement between workloads and platforms. • Enhanced visibility enabling SOC teams to detect and respond to threats 40% faster. • 100% enforcement of artifact verification before production deployment.

Key Takeaways

- Direct mapping of identified risks to engineering solutions enables actionable and defensible security objectives.
- Zero Trust Platform & Workload Architecture ensures consistent enforcement of access controls, segmentation, and monitoring across heterogeneous environments.
- Measurable outcomes provide validation points for continuous optimization and operational assurance.

This consolidated use case and summary table provide technical teams with an explicit, actionable reference for aligning compute, platform, and workload security controls with enterprise risk management and resilience objectives.

Section 5. Requirements (Inputs)

A defensible compute, platform, and workload security architecture is grounded in clearly defined, actionable inputs. These requirements establish the technical, procedural, and policy conditions that must be present before implementation begins. By setting these preconditions, ISAUnited's Defensible Standards ensure organizations are prepared for disciplined, engineering-driven security integration.

5.1 Zero Trust Platform & Workload Security

- All access requests—regardless of user, platform component, workload, device, or location—must be continuously verified before permission is granted.
- Implement ongoing authentication, dynamic authorization, and continuous trust validation across platform and workload layers.
- Enforce least privilege by default, and require multi-factor authentication (MFA) for all privileged actions at both platform and workload levels.

5.2 Platform & Shared Responsibility Model Compliance

- Align security controls explicitly with the shared responsibility model for each compute platform, orchestrator, or cloud provider.
- Delineate provider-managed versus organization-managed security functions.
- Maintain procedures to verify the effectiveness of both provider and organization controls to prevent gaps.

5.3 Automated Security Enforcement

- Use WSPM, infrastructure-as-code (IaC) policy engines, policy-as-code (PaC) frameworks, and runtime security agents to enforce consistent security across environments.
- Automate detection and remediation of misconfigurations, policy violations, and compliance gaps.
- Integrate automated security workflows into CI/CD pipelines.
- Enforce admission controls and verify-on-pull/verify-before-start in pre-production and production; policy-as-code bundles include registry allowlists/denylists and artifact signature/attestation checks.
- Pipelines fail closed on unsigned or unattested images, or on failing admission policies.
- Admission/verification policies cover serverless functions: only signed and attested packages/layers are deployable; build-time dependencies are pinned; pipelines fail closed on unsigned or policy-failing functions.
- EP-03.3: policy bundle, function-signing/attestation config, failed-gate logs.

5.4 Segmentation & Trust Boundary Enforcement

- Design and implement segmentation at platform and workload layers using software-defined networking, micro-segmentation, and identity-based access controls.
- Enforce isolation between workloads, services, and control planes to prevent lateral movement.
- Apply software-defined perimeters and context-aware access controls based on identity and risk.
- Define default-deny namespace network policies for containers, with explicit egress allowlists (including DNS); platform management endpoints are not reachable from workload namespaces.

- Serverless functions integrated with private networking; outbound traffic restricted by egress allowlists; function URLs or public endpoints require strong authentication and do not bypass policy.
- EP-03.4: function-to-service contract tests, egress-deny events, allowlist map.

5.5 Data Encryption & Compliance

- Enable encryption by default for all data at rest and in transit (for example, AES-256, TLS 1.3).
- Align encryption, key management, and data handling with data-residency, privacy, and compliance requirements.
- Implement centralized KMS with automated key rotation and strict access controls.
- Deliver secrets via workload identity and a dedicated secrets store; prohibit secrets in images and plaintext environment variables.

5.6 Supply-Chain Integrity & Artifact Trust

- Require all artifacts (for example, VM images, container images, code packages) to be signed, verified, and vulnerability-scanned before deployment.
- Maintain a software bill of materials (SBOM) for all deployable artifacts to support provenance and compliance.
- Enforce verify-on-pull or verify-before-start so only signed and attested artifacts with valid policy checks can execute.
- Maintain an approved registry/namespace list; permit only immutable, approved tags in production (no “latest” or other floating tags).
- Achieve 100 % SBOM coverage for all deployable images and functions; validate signatures and attestations at pull or start.
- Function environment variables never store plaintext secrets; secrets are injected at runtime from a dedicated store and protected by KMS; ephemeral storage encryption is enabled where supported.
- EP-03.5: secret-access policy, rotation logs, env-scan results.

5.7 Administrative Access & Privileged Operations

- Use bastioned administrative access with MFA and just-in-time (JIT) elevation; prohibit standing administrator roles on platforms and workloads.
- Enable session recording and command logging for privileged actions on control planes and hosts.
- Define emergency access procedures with approval, time bounds, and post-use review.
- Perform cluster/control-plane administration (for example, virtualization and container-orchestration consoles) only via a bastion with MFA and JIT; no standing cluster-admin.

- Allow direct “exec” into production workloads only with JIT, session capture, and ticketed justification.

5.8 Baseline & Hardening Standards

- Adopt hardened baselines for hosts/OS, orchestrators, control planes, and images; define drift-prevention policy and remediation windows.
- Enforce admission control (where applicable) to block non-conformant workloads at deploy/admission time.
- Maintain a register of approved base images and golden artifacts tied to SBOMs and signatures.
- Require container baselines: non-root execution with non-zero UID, read-only root filesystem, minimal capabilities, and defined syscall/capability profiles; disallow privileged containers and host mounts without approved exceptions.
- Maintain a golden image catalog for base images with SBOMs, signatures, and a documented patch cadence.

5.9 Telemetry, Logging & Evidence Readiness

- Define required telemetry (host, control-plane, workload, network) and ensure immutable log storage with policy-aligned retention.
- Standardize event schemas so SIEM/XDR and §12 V&V can correlate platform and workload events.
- Establish Evidence Pack conventions (IDs, locations) for scans, policies, signatures/attestations, and test results referenced by this standard.
- Collect container lifecycle events (create/start/stop), image-pull provenance, admission denials, and network-policy denials; retain control-plane audit logs and correlate with workload identity.
- Evidence Packs include registry policy, admission policy, signing/attestation reports, SBOM coverage, and namespace network-policy maps.

Practitioner Guidance:

Use these requirements as readiness gates before implementing §6 and scheduling tests in §12:

- Map each §5.x item to one artifact and one §12 test (for example, 5.6 → Signing Policy + verify-on-pull test).
- Keep single sources of truth: platform diagrams, PaC/IaC repository, and golden-image catalog (with SBOM and signatures).
- Assign ownership per gate and record it in the CPW register.
- Baseline before go-live: standing admins (= 0), JIT usage, east–west default-deny coverage, admission-policy pass rate, SBOM coverage (= 100 %), KMS rotation cadence.
- Fail closed: block deployments for missing MFA/JIT (5.1/5.7), unsigned or unattested images (5.6), admission violations (5.8), or absent segmentation/KMS bindings (5.4/5.5).

	• Enforce admission-time checks and log denials with Evidence Pack IDs; revalidate gates after major CPW changes and at least quarterly.

Section 6. Technical Specifications (Outputs)

Technical specifications define the concrete, defensible outputs required to satisfy this standard. Each output area translates policy into measurable, actionable security outcomes, establishing a robust foundation for secure compute, platform, and workload protection across cloud-native, hybrid, and enterprise environments.

Outputs must be:

- **Measurable:** validated by scans, logs, audits, or tests
- **Actionable:** implementation-ready, not policy slogans
- **Aligned:** traceable to §5 Requirements and sub-standards

6.1. Identity & Access Security for Platforms and Workloads

- **Multi-Factor Authentication (MFA):** MFA SHALL be enforced for all privileged and administrative accounts at platform and workload levels. Prefer phishing-resistant factors for privileged actions. Privilege elevation SHALL be Just-in-Time (JIT) with session recording.
- **RBAC/ABAC:** Roles and attributes SHALL implement least privilege for humans, services, and workloads. Standing administrator roles SHALL NOT exist; elevation is time-bound with approval and logging.
- **Native IAM Integration:** Platform-native IAM and workload-identity mechanisms SHALL issue short-lived credentials and enforce policy centrally.
- **Privileged Access Operations:** Administrative access SHALL traverse a hardened path (for example, a bastion) with MFA and JIT; commands and sessions are captured and retained per retention policy.
- **Access Reviews:** Access reviews SHALL be automated on a defined cadence; stale and orphaned identities SHALL be removed within 7 days of detection.

6.2. Platform & Workload Network Security & Segmentation

- **Software-Defined Segmentation:** L3–L7 segmentation SHALL be expressed as policy-as-code (peer-reviewed, staged rollout) to isolate workloads, control planes, and services across environments.
- **Private Endpoints & Zero Trust:** Sensitive platform/workload communications SHALL use private endpoints or software-defined perimeters. Service-to-service traffic on sensitive paths SHALL require authenticated encryption (for example, mTLS) and identity-based authorization.

- **Network Firewalls:** Platform-native firewalls with layer-7 controls SHALL enforce default-deny at trust boundaries. Management planes SHALL be isolated; administrative access occurs only via a hardened bastion with MFA and JIT.
- **Micro-Segmentation:** East–west traffic controls SHALL be identity/context aware and scoped to least privilege (for example, namespace/app/role), with egress allowlists for workloads.
- **Network Access Controls:** Security groups, ACLs, and route policies SHALL tightly control ingress and egress per workload; changes are version-controlled and validated pre-deployment.
- **Container Network Policy:** Workload namespaces SHALL enforce identity- or label-based network policies for ingress and egress; default-deny east–west with explicit egress allowlists (including DNS). Platform management endpoints SHALL NOT be reachable from workload namespaces.

6.3. Data Protection & Encryption

- **Encryption Standards:** Data at rest SHALL use AES-256 (or stronger); data in transit SHALL use TLS 1.3 (or stronger) with approved cipher suites. Certificates and keys SHALL be rotated automatically per policy.
- **Centralized Key Management:** A centralized KMS SHALL control key generation, access, rotation, and auditing with separation of duties.
- **Data Classification & Handling:** Automated tagging/classification and lifecycle policies SHALL be applied to workloads and platform storage.
- **Compliance Alignment:** Cryptographic configurations, retention, and residency SHALL conform to applicable regulatory and organizational requirements.
- **Platform Integrity (Boot/Runtime Measurements):** Where supported, secure/verified boot and measured boot SHALL be enabled for hosts and control planes; integrity events SHALL be forwarded to centralized telemetry and used as gates for workload admission.

6.4. API & Runtime Security

- **API Gateways:** External and inter-service APIs SHALL be fronted by gateways that enforce authentication, authorization, rate limiting, schema/validation, and logging. Anonymous access to sensitive APIs is prohibited.
- **Modern Protocols:** OAuth 2.0, OpenID Connect, and JWT-based models SHALL be used for token-based access where applicable; token lifetimes are short-lived and scoped.
- **Runtime Controls (VMs/Containers/Serverless):**
    - **Artifact Integrity:** Artifacts SHALL be signed and attested; verify-on-pull/verify-before-start SHALL block unsigned, unattested, or policy-failing artifacts.
    - **Registry & Admission Policy:** Only approved registries and namespaces SHALL be allowed; admission/verify-before-start

SHALL block artifacts from unapproved sources or with failing policy checks.
    - **Vulnerability Management:** Images and functions SHALL be vulnerability-scanned pre-deploy and on a defined cadence; critical policy failures block promotion.
    - **Least-Privilege Execution:** Workloads SHALL run with least privilege (for example, no privileged containers; read-only root filesystem; minimal capabilities; syscall/process/network policies).
    - **Serverless Runtime Baseline:** Functions SHALL execute with least privilege (execution role scoped to function), short timeouts and concurrency limits, minimal permissions to event sources and destinations, and no public unauthenticated “function URLs” unless explicitly approved with expiry.
    - **Function Package Integrity:** Deployed function code and layers SHALL be **signed/attested**; verify-before-deploy SHALL block unsigned or policy-failing packages; third-party layers are quarantined until verified.
    - **Egress and Interface Policy:** Functions running inside private networking SHALL use identity-based policies and egress allowlists; direct internet egress is denied unless explicitly approved with expiry.
    - **Secrets and Configuration:** Secrets SHALL be provided at invocation via a secrets store and short-lived tokens; configuration SHALL avoid embedding secrets in environment variables; ephemeral /tmp use is minimized and not trusted for persistence.
    - **Telemetry:** Capture per-invocation logs and metrics (invocations, duration, errors, throttles, cold starts) and correlate with identity and admission decisions in §6.5.
    - **EP-03.6 / EP-03.21:** function-signing policy, verify logs, allowed-registry/source list, denial events; per-invocation metrics in SIEM/XDR.

- **Container Runtime Security (normative subset):**
    - **Image Baselines:** Only approved base images with SBOMs SHALL be used; tags SHALL be immutable; “latest” or other floating tags are prohibited in production.
    - **Allowed Registries:** Only approved registries and namespaces SHALL be permitted; unapproved sources SHALL be blocked at admission.
    - **User & Isolation:** Containers SHALL run as non-root with a non-zero UID; privileged containers, hostPID/hostNetwork, and hostPath mounts are prohibited unless formally approved with compensating controls and expiration.
    - **Kernel/Capability Profiles:** System-call and capability profiles SHALL be enforced; dangerous capabilities (for example,

NET_RAW) SHALL be dropped; writable device access is prohibited by default.
    - **Filesystem & Secrets:** Root filesystem SHALL be read-only; writable volumes are scoped to necessity. Secrets SHALL be delivered via workload identity or a dedicated secrets store—never baked into images or stored as plain environment variables.
    - **Resource Limits & Quotas:** CPU and memory requests/limits SHALL be set to constrain blast radius and support reliable autoscaling and eviction behavior.
    - **Continuous Scanning:** Deployed images SHALL be rescanned on a defined cadence; high-severity findings trigger automated quarantine/rollback workflows.
    - **Runtime Detection:** Policy SHALL detect and respond to container-escape attempts, crypto-mining, reverse shells, unexpected outbound beacons, and tampering with runtime policy.
- **Secrets Management:** Secrets, keys, and tokens SHALL be stored in a dedicated vault, issued short-lived to workloads via identity, and rotated automatically. Secrets SHALL NOT be embedded in code or images.
- **API Threat Detection:** Abuse and anomaly detection (for example, authorization bypass, injection, enumeration) SHALL be monitored with alerting and response tied to §6.5.

6.5. Monitoring, Detection & Incident Response

- **Centralized SIEM/XDR:** Host, control-plane, workload, and network telemetry SHALL be normalized and correlated centrally. Time synchronization and schema standards are required for correlation.
- **Detection Engineering:** Behavioral, heuristic, and rules-based detections (optionally ML-assisted) SHALL cover control-plane abuse, lateral movement, privilege escalation, and runtime policy violations.
- **Automated Response:** SOAR/playbooks SHALL isolate or restart compromised workloads, revoke credentials and keys, quarantine artifacts/registries, and open tracked incidents with evidence links.
- **Immutable Audit Trails:** Logs and audit trails SHALL be tamper-evident (for example, WORM or equivalent) with retention aligned to policy; access is monitored and auditable.
- **Continuous Posture Assessment:** WSPM/CSPM/IaC-as-code checks SHALL continuously assess compliance and drift, with auto-reconciliation or time-bounded remediation per severity.
- **Container Telemetry:** Collect container lifecycle events (create/start/stop), image-pull provenance, policy denials at admission, syscall/capability violations, and namespace-level network-policy denials; retain control-plane audit logs and correlate with workload identity.
- **Quarantine & Rollback:** On policy violation or high-severity finding, orchestrate automated isolation of the affected pod or workload, revoke tokens and keys, and roll back to a last-known-good signed image; record an Evidence Pack ID.

- **Serverless Telemetry:** Collect per-invocation metrics (invocations, errors, throttles, duration, cold starts) and bind them to function identity, source package digest, and admission decision; alert on anomalous spikes, unusual egress, or policy-denial rates.
- **EP-03.9**: telemetry schema sample, correlation queries (admission → invocation → egress), alert runbooks.

Practitioner Guidance:

- Begin with a gap analysis against §6 outputs; prioritize control-plane hardening, identity, segmentation, and artifact trust.
- Encode these outputs as policy-as-code and infrastructure-as-code; enforce fail-closed gates in CI/CD and at admission/verify-before-start.
- For every §6 control, pair a §12 test and an evidence artifact (for example, policy exports, attestation reports, segmentation maps, SIEM queries).
- Use staged rollout (canary) and peer review for policy changes; track drift MTTR, and failed-gate rates as quality signals.
- Train operators on privileged path operations (bastion + JIT + session capture) and rehearse IR playbooks for control-plane and runtime compromise scenarios.

Quick Win Playbook:

Title: Verify-Before-Start and Approved Registries (Production)

Objective: Prevent untrusted or tampered artifacts from entering runtime by enforcing signature and attestation at admission and restricting workloads to approved registries and namespaces.

Target: Enforce verify-on-pull/verify-before-start and approved registries for production workloads (§6.4).

Component/System: Admission controller + image registry + workload runtime (VMs/containers/serverless).

Protects: Control planes and workloads from untrusted or tampered artifacts entering runtime (supply-chain compromise).

Stops/Detects: Unsigned or unattested images; artifacts from unapproved registries/namespaces; mutable “latest”/floating tags; signature mismatches at deploy/start.

Action: Enable signature and attestation enforcement at admission/verify-before-start; allow only approved registries/namespaces; prohibit floating tags; run a smoke deploy that attempts (1) a signed and attested image from an approved

	registry (allow) and (2) an unsigned or unapproved-registry image (deny). Ensure deny/verify events export to centralized telemetry; record owner and review cadence. **Proof:** Policy-as-code commit/diff, admission/verification deny logs, registry allowlist/denylist, digest list of running images, and signature/attestation reports → Evidence Pack EP-03.10. Reference Table C-5 (row 5.6). **Metric:** 100 % of unsigned, unattested, or unapproved-registry images are denied; 100 % of running images use approved registries and immutable digests; deny/verify events appear in SIEM/XDR within target MTTD. **Rollback:** Revert the admission/policy bundle to the previous commit; if required, issue a time-bounded exception with owner and expiry; archive new artifacts under EP-03.10 as superseded.

Section 7. Cybersecurity Core Principles

The following ISAUnited Cybersecurity Core Principles are foundational to the design, implementation, and ongoing management of secure compute, platform, and workload security architectures. Each principle guides architectural decisions, technical controls, and operational practices, ensuring that environments are resilient, measurable, and engineered to withstand real-world threats.

Table C-2:

Principle Name	Code	Applicability to Compute, Platform & Workload Security Architecture
Least Privilege	ISAU-RP-01	Scope platform and administrative roles, and workload/service identities, to the minimum required; prohibit standing administration; enforce JIT elevation with session capture and time bounds.
Zero Trust	ISAU-RP-02	Continuously verify humans, services, workloads, and platform components; require identity-based policy for service-to-service calls (for example, mTLS) and default-deny east–west within clusters.
Defense in Depth	ISAU-RP-04	Layer controls across control planes, hosts/OS, registries, images, runtime policy, network segmentation, and telemetry so a single failure cannot compromise workloads.

Principle Name	Code	Applicability to Compute, Platform & Workload Security Architecture
Secure by Design	ISAU-RP-05	Encode guardrails as IaC/PaC; enforce admission/verify-before-start; treat images and policies as versioned artifacts with peer review and staged rollout.
Minimize Attack Surface	ISAU-RP-06	Remove unused services, modules, and capabilities; run containers as non-root with read-only root filesystem; disallow privileged and host mounts; restrict administrative paths behind bastions.
Secure Defaults	ISAU-RP-10	Default to deny (network and admission), encrypt by default (at rest and in transit), and enforce signed and attested artifacts; any relaxation requires approved, time-bounded exceptions.
Resilience & Recovery	ISAU-RP-14	Design for fault tolerance and rapid rollback: quarantine non-compliant workloads, automatically roll back to a last-known-good signed image, and practice control-plane recovery.
Evidence Production	ISAU-RP-15	Generate immutable, correlated logs from hosts, control planes, and workloads; capture admission denials, image provenance, and privileged sessions for audit and forensics.
Cryptographic Agility	ISAU-RP-17	Use centralized KMS with automated rotation; support cipher and protocol upgrades (for example, TLS 1.3+) and re-issue identities/keys without downtime.
Protect Confidentiality	ISAU-RP-18	Enforce strong encryption, scoped access to secrets via identity-bound tokens, and provenance-aware image handling to prevent sensitive data exposure.
Protect Availability	ISAU-RP-20	Ensure high availability for control planes and critical workloads; apply resource limits and quotas to contain noisy neighbors and preserve capacity during incidents.

Practitioner Guidance:

- Integrate these principles into §6 outputs (e.g., RP-01 ↔ §6.1 JIT/MFA; RP-06 ↔ §6.4 container least-privilege; RP-15 ↔ §6.5 immutable logs).

	• Validate adherence via §12 tests: privilege-escalation simulations, admission/verify-before-start denials, east–west block tests, and rollback drills. • Teach “principle → control → evidence” mapping in runbooks so teams can show defensibility on demand.

Section 8. Foundational Standards Alignment

The Compute, Platform & Workload Security Architecture (ISAU-DS-CPW-1000) must align closely with globally recognized foundational standards to ensure interoperability, regulatory compliance, and a consistent risk management approach. While ISAUnited Defensible Standards provide detailed technical guidance and engineering rigor, alignment with established frameworks is critical for auditability, industry acceptance, and seamless integration into existing security and compliance programs.

Table C-3. Foundational standards relevant to this Parent Standard:

Framework	Standard ID	Reference Focus
NIST	CSF 2.0	Cybersecurity Framework core outcomes are organized into Govern, Identify, Protect, Detect, Respond, and Recover, with a governance overlay for risk and program alignment.
NIST	SP 800-53 Rev. 5	Baseline security and privacy controls relevant to platforms, workloads, identity, boundary protection, and continuous monitoring in enterprise and hybrid environments.
NIST	SP 800-190	Containerized application security: image hardening, verification, orchestrator configuration, and runtime protection for containers and platforms.
NIST	SP 800-207	Zero Trust architecture principles and reference models for continuous verification, least privilege, identity-centric policy, and segmentation across platforms and workloads.
ISO/IEC	27001:2022	ISMS requirements that underpin governance and assurance; CPW implementations should not conflict with the organization’s ISMS controls and risk treatment.
ISO/IEC	27002:2022	

Framework	Standard ID	Reference Focus
		Information security controls (93 controls across organizational, people, physical, and technological themes) are used to implement ISMS controls in practice.
ISO/IEC	27017	Cloud services security guidance, including shared-responsibility alignment and platform/workload control expectations for cloud and hybrid deployments.
ISO/IEC	27033 (series)	Network security concepts and design guidance supporting east–west and north–south protections for platform/workload segmentation and secure communications.

NOTE: ISAUnited Charter Adoption of Foundational Standards.

Per the ISAUnited Charter, the institute formally adopts the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) and the National Institute of Standards and Technology (NIST) as its foundational standards bodies, consistent with their public encouragement of organizational adoption. Parent Standards align to ISO/IEC and NIST for architectural grounding and auditability, and this alignment flows down to Sub-Standards as invariants and minimum requirements that may be tightened but not weakened. ISAUnited does not restate or speak on behalf of ISO/IEC or NIST; practitioners shall consult the official publications and terminology of these organizations, verify scope and version currency against the latest materials, and implement controls in a manner consistent with ISAUnited security invariants and the requirements of this standard.

Future Sub-Standard Integration

As detailed sub-standards are developed under ISAU-DS-CPW-1000, specific mappings to NIST and ISO/IEC controls will be incorporated at the control level. Mappings to other frameworks (e.g., CSA CCM, CIS) will appear in §9 for clarity and in sub-standards to guide implementation in diverse technical contexts.

Practitioner Guidance:

ISAUnited Defensible Standards are designed to complement and extend globally recognized standards. Practitioners should:

- Map sub-standard technical controls and §6 specifications to NIST/ISO for foundational alignment and audit traceability (including CSF 2.0's Govern function for program governance).

	• Use NIST/ISO as authoritative references during architecture reviews, risk assessments, and compliance evaluations. • Maintain alignment as part of continuous improvement so architecture remains defensible, measurable, and adaptable to evolving risks and regulatory demands. • Place CSA CCM/CIS mappings in §9 (Security Controls) and within sub-standards to keep foundational versus control frameworks clearly separated.

Section 9. Security Controls

This section identifies the technical control families and specific control references directly supported or enforced by the Compute, Platform & Workload Security Architecture (ISAU-DS-CPW-1000). These controls explicitly link architectural and engineering guidance to recognized cybersecurity frameworks, ensuring traceability, auditability, and consistent implementation across diverse environments.

Purpose and Function

Security controls translate the standard's architectural intent into actionable, measurable safeguards. They provide the tactical foundation to enforce confidentiality, integrity, availability, authentication, authorization, and auditability across compute, platform, and workload environments.

By explicitly mapping to widely accepted frameworks such as the CSA Cloud Controls Matrix (CCM), CIS Controls v8, and OWASP standards, ISAUnited ensures:

- Clear alignment with recognized industry best practices and regulatory frameworks.
- Interoperability across varied operational contexts.
- Consistency and reusability of controls in sub-standards aligned to this Parent Standard.

These mappings also enable engineers and auditors to measure and validate the defensibility of implementations aligned with this standard.

Implementation Guidance

Standard developers and practitioners must:

- Reference at least three technical controls from authoritative cybersecurity frameworks.

- Provide framework name, acronym, control ID, and a concise, actionable description.
- Align selected controls to the technical specifications (§6 outputs) and core principles (§7).
- Select concrete, implementation-level controls rather than high-level statements.

Table C-4. Control Mappings for Compute, Platform & Workload Security

Framework	Control ID	Control Name / Description
CIS Controls v8	13.1	Protect Network Infrastructure — Enforce identity-based segmentation for platforms/workloads; default-deny east–west; restrict admin paths to bastions per §6.2.
CIS Controls v8	14.4	Centralize Security Event Logging — Aggregate host, control-plane, and workload telemetry in SIEM/XDR; include admission denials and image-verify events per §6.5.
CIS Controls v8	4.3	Secure Configuration of Assets and Software — Apply hardened baselines to hosts, orchestrators, and images; prevent drift with PaC/IaC gates per §6.2/§6.4.
CIS Controls v8	5.1	Inventory of Service Accounts — Maintain and constrain platform/workload service identities; rotate credentials; remove orphans per §6.1.
CIS Controls v8	6.5	Access Control Management — Require MFA and JIT for privileged operations; prohibit standing admin roles; record sessions per §6.1.
CIS Controls v8	8.2	Audit Log Management — Preserve immutable audit trails (WORM/equivalent) with time sync and schema standards for correlation per §6.5.
CSA CCM	DSI-03	Data Security & Information Lifecycle — Encrypt data at rest (AES-256) and in transit (TLS 1.3+); manage keys centrally via KMS per §6.3.
CSA CCM	IAM-05	Identity & Access Management — Enforce least privilege and MFA for administrative and privileged platform/workload operations per §6.1.
CSA CCM	IAM-09	Identity & Access Management — Strong authentication and short-lived tokens for platform/workload access; remove stale identities per §6.1.

Framework	Control ID	Control Name / Description
CSA CCM	IVS-06	Virtualization & Network Security — Implement micro-segmentation and isolation for hypervisors, clusters, and namespaces per §6.2.
OWASP ASVS	V2.1	Authentication Architectural Requirements — Strong authentication for platform/cluster/admin APIs and workload endpoints; token scope/TTL per §6.4.
OWASP Top 10	A02:2021	Cryptographic Failures — Enforce approved ciphers/protocols, KMS rotation, and secret handling policies (no secrets in images/plain env) per §6.3/§6.4.
OWASP Top 10	A04:2021	Insecure Design — Encode guardrails as IaC/PaC; verify before start/admission to prevent non-conformant workloads per §6.4.

NOTE: Use of External Control Frameworks.

ISAUnited maps to external control frameworks to provide alignment and traceability, but does not speak on behalf of those organizations. Practitioners shall consult and follow the official practices, recommendations, and implementation guidance of the Center for Internet Security (CIS), the Cloud Security Alliance (CSA), and the Open Worldwide Application Security Project (OWASP) when applying controls. Always verify control identifiers, scope, and version currency against the publishers' latest materials. Where wording differs, use the framework's official documentation while maintaining consistency with ISAUnited security invariants and this standard's requirements.

Additional References

As platform and workload security evolves, Sub-Standard Authors may add controls from other frameworks (e.g., CIS Benchmarks, NIST SP 800 series) to maintain relevance.

Sub-Standard Expectations

Sub-standards developed under ISAU-DS-CPW-1000 must:

- Select and enforce explicit technical controls relevant to their focus area (e.g., IAM, runtime protection, segmentation).

- Provide detailed mappings of these controls to verification, implementation, and operational requirements.
- Justify and document deviations from referenced control families to maintain transparency and defensibility.

Section 10. Engineering Discipline

This section defines the architectural thinking, rigorous engineering processes, and disciplined operational behaviors required to implement the Compute, Platform & Workload Security Architecture (ISAU-DS-CPW-1000). ISAUnited's Defensible Standards are not compliance checklists; they are engineered systems, grounded in systems thinking, critical reasoning, and Verification & Validation (V&V), that produce measurable, auditable, defensible outcomes across platforms, control planes, and workloads.

10.1 Purpose & Function

Purpose. Establish a repeatable, auditable way of working that integrates systems thinking, lifecycle controls, adversary-aware design, and measurable outcomes for compute, platform, and workload security.

Function in D10S. Parent Standards set expectations and invariants. Sub-Standards convert them into controls as code, test specifications, and evidence artifacts embedded in delivery and operations.

10.2 Systems Thinking

Goal: Make the CPW system legible end-to-end; components, interfaces, dependencies, and failure modes—so controls sit where risk actually manifests.

10.2.1 System Definition & Boundaries

- Declare system purpose, scope, stakeholders, and in-/out-of-scope assets (control planes, hosts/OS, VMs, containers, serverless, registries, KMS, vaults, SIEM/XDR).
- Model trust zones, segmentation, and interconnects (clusters/nodes/namespaces; VPC/VNet/subnets; service endpoints/private links; administrative paths/bastions).

10.2.2 Interfaces & Contracts

- Maintain Interface Control Documents (ICDs) for every interconnection (platform/administrative APIs, admission controllers, registries, service mesh, queues, data stores, identity providers).
- For each interface, specify: authentication/authorization model, identity type (human/service/workload), data classification, rate/flow

limits, error handling, telemetry, and security invariants (for example, “verify-before-start on artifact pull”).

10.2.3 Dependencies & Emergent Behavior

- Map shared services (KMS, DNS, IAM patterns, logging, time sync) and blast radius per dependency.
- Identify emergent risks from composition (for example, benign configuration A + default B → unsigned image admitted; mesh policy + permissive egress → lateral movement).

10.2.4 Failure Modes & Safeguards

- For critical paths, document failure modes (misconfiguration, drift, overload, credential abuse, artifact-trust failure, control-plane compromise) and safeguards (deny by default, least privilege, egress allowlists, circuit breakers, canary deploys, immutable infrastructure, verify-on-pull/verify-before-start).
- Treat security invariants as non-negotiable requirements (for example, “no public ingress to management plane,” “unapproved registries denied at admission,” “secrets not embedded in images”).

Required Artifacts (minimum): Context diagram with trust boundaries; interface map with ICDs; dependency and blast-radius matrix; invariants register.

10.3 Critical Thinking

Goal: Replace assumptions with explicit reasoning that survives review, attack, and audit.

10.3.1 Decision Discipline

- Use Architecture Decision Records (ADRs): problem → options → constraints/assumptions → trade-offs → decision → invariants → test/evidence plan (who, when, how measured).

10.3.2 Engineering Prompts

- **Boundaries:** What is the CPW system? Where are the trust boundaries and why?
- **Interfaces:** What must always be true at each interface (invariants)? How do we test it?
- **Adversary:** Which attack techniques are credible here (for example, control-plane abuse, container escape, supply-chain path)? What is the shortest attack path?
- **Evidence:** What objective signals prove this control works today and after the change?
- **Failure:** When this fails, does it fail safe? What is the operator’s next action?

Required Artifacts (minimum): ADRs; assumptions and constraints log; evidence plan per decision.

10.4 Domain-Wide Engineering Expectations

Secure System Design

- Define CPW boundaries (clusters/namespaces, control planes/administrative paths, registries, KMS/vaults, telemetry sinks).
- Validate boundaries and trust relationships through structured reviews using §10.2 artifacts.

Implementation Philosophy — "Built in, not bolted on."

- Integrate controls at design time and pipeline/admission time; avoid post-hoc patching.
- Express controls as policies/configuration as code bound to invariants in §10.2.4 (for example, verify-before-start, default-deny east–west, non-root execution).

Lifecycle Integration

- Embed CPW controls into DevSecOps (IaC/PaC), change management, and immutable deployments.
- Enforce version-controlled reviews with required ADRs and evidence updates for every change.

Verification Rigor (V&V)

- Combine automated checks (policy validation, IaC scanning, admission/segmentation tests, runtime guardrails) with manual tests (penetration testing, adversary emulation).
- Require continuous validation in pipelines and runtime monitoring tied to invariants (for example, deny unsigned images; block unapproved egress).

Operational Discipline

- Monitor for drift and unauthorized change; auto-remediate where safe with time-bounded exceptions.
- Maintain pre-approved playbooks for control-plane incidents, key rotation, artifact quarantine/rollback, and namespace isolation.

10.5 Engineering Implementation Expectations

- Policy/Configuration as Code. Manage network/segmentation, admission, identity, cryptography, and logging policies as code under version control with peer review and provenance.
- Structured Enforcement Pipelines. CI/CD gates for unit/policy tests → security-integration tests → verify-on-pull/verify-before-start → canary/blue-green → rollback.
- Explicit Security Boundaries. Maintain diagrams and ICDs; continuously validate posture (e.g., default-deny, approved registries) through targeted audits and smoke tests.
- Automated Security Testing. Integrate IaC scanning, configuration validation, secrets detection, dependency/image checks, and adversary emulation before production.

- Traceable Architecture Decisions. Link ADRs to controls, tests, and evidence; update ADRs and evidence on every change request.

Required Artifacts (minimum): Controls-as-code repository; pipeline policy gates; boundary/ICD set; automated test results; evidence ledger (see §10.7 and §12).

10.6 Sub-Standard Alignment (inheritance rules)
Sub-Standards must operationalize this discipline with domain-specific detail:

- Platform & Workload Identity (for example, ISAU-DS-CPW-1030). Identity as code; least-privilege baselines; short-lived tokens; automated policy validation; pipeline enforcement; peer review and automated tests before deploy.
- Zero Trust Segmentation (for example, ISAU-DS-CPW-1040). Identity-centric network policy; default-deny east–west; private endpoints; admission-time checks; breach-and-attack simulation for lateral-movement paths.
- Software Supply-Chain Integrity & Provenance (for example, ISAU-DS-CPW-1080). End-to-end signing/attestation; verify-on-pull/verify-before-start; exception handling with sunset; automated verification in CI and at deploy.

10.7 Evidence & V&V (what proves it works)
Establish an Evidence Pack per system containing:

- **Design Evidence:** diagrams with trust boundaries, ICDs, invariants register, ADRs.
- **Build Evidence:** IaC/PaC repositories; signed artifacts/attestations; pipeline logs; test results.
- **Operate Evidence:** runtime policy decisions; drift reports; control telemetry (for example, admission denials, image-verify events, capability/syscall violations); incident and rollback records.
- **Challenge Evidence:** red-team/penetration-test reports; adversary-emulation outcomes; remediation closure with re-test.
- Each control requires objective pass/fail criteria, a specified test frequency, a designated responsible owner, and a defined retention policy. Map Evidence Pack IDs into §12 traceability.

10.8 Example: Sub-Standard Discipline Alignment (Zero Trust Segmentation)

Scope: ISAU-DS-CPW-1040 Zero Trust Segmentation for CPW
Design: Define identity and trust zones (clusters/namespaces, administrative paths, service identities); enumerate invariants (for example, “default-deny namespace east–west,” “platform management endpoints unreachable from workloads”).
Implement: Express segmentation and admission policies as code; enforce

identity-based authorization and mTLS; deny unapproved egress and unapproved registries.
V&V: Automated policy tests; namespace smoke tests for lateral movement; admission denials for unapproved artifacts; periodic adversary emulation focused on east–west bypass.
Operate: The Evidence Pack includes policy repository history, gate results, admission/deny logs, segmentation maps, incident records, and closed-loop remediation.

Section 11. Associate Sub-Standards Mapping

Purpose of Sub-Standards

ISAUnited Defensible Sub-Standards are detailed, domain-specific extensions of the Compute, Platform & Workload Security Architecture (ISAU-DS-CPW-1000). Each Sub-Standard delivers:

- Granular technical guidance tailored to specialized CPW security domains.
- Actionable implementation strategies translating architectural intent into practical operational controls.
- Precise validation methodologies ensuring outputs are measurable and auditable.
- Alignment with foundational architectural principles and technical specifications of the Parent Standard.

Sub-Standards bridge the gap between broad architectural direction and the detailed technical requirements necessary for robust engineering, validation, and auditing across platform, control plane, and workload security.

Scope and Focus of CPW Sub-Standards

Sub-Standards under ISAU-DS-CPW-1000 will address specialized topics, including:

Platform & Workload Hardening & Secure Configuration
Example: ISAU-DS-CPW-1010: Hardened Configuration for Platform & Containerized Workloads
• Prescribes CIS, cloud-provider, and orchestrator benchmarks.
• Requires automated scanning in CI/CD pipelines to detect vulnerabilities and enforce baselines.
• Mandates removal of unnecessary packages and disabling privileged/root access.
• Enforces immutable infrastructure and drift-prevention controls.

Runtime Threat Detection & Response
Example: ISAU-DS-CPW-1020: Runtime Threat Detection for Platforms & Workloads
- Specifies runtime security agent deployment for control planes, VMs, containers, and serverless.
- Defines behavioral anomaly detection for process, network, and file activity.
- Outlines automated response actions (isolation, restart) upon suspicious activity.
- Integrates with SIEM/SOAR for centralized detection and response.

Identity & Access Management (IAM)
Example: ISAU-DS-CPW-1030: Platform & Workload Identity Lifecycle Management
- Details least-privilege IAM role/policy engineering for platform services and workloads.
- Requires automated credential provisioning, rotation, and revocation.
- Mandates integration with platform-native identity providers and service meshes.
- Establishes periodic access reviews and orphan-privilege detection.

Zero Trust Segmentation
Example: ISAU-DS-CPW-1040: Zero Trust Segmentation for CPW Environments
- Provides segmentation methodologies based on identity, risk, and context.
- Requires continuous verification of all platform-workload communications.
- Enforces policy-driven micro-segmentation (e.g., SDPs, mesh/mTLS).
- Integrates posture-based dynamic access controls.

Data Protection & Encryption
Example: ISAU-DS-CPW-1050: CPW Data Encryption & Key Management
- Mandates encryption of all data at rest and in transit (AES-256, TLS 1.3).
- Requires centralized key management with automated rotation and strict access controls.
- Includes compliance checks for residency/privacy laws.
- Defines secure key backup/recovery and audit logging.

Infrastructure as Code (IaC) Governance
Example: ISAU-DS-CPW-1060: Secure Infrastructure as Code for CPW
- Requires all infrastructure/platform definitions as code with version control and peer review.
- Mandates automated IaC scans pre-deployment.
- Enforces policy-as-code checks in CI/CD.
- Documents change management and rollback processes.

API & Secrets Management
Example: ISAU-DS-CPW-1070: Secure API & Secrets Management for CPW
- Requires API gateways with integrated authentication, authorization, and rate limiting.
- Mandates secret/credential storage in secure vaults with rotation.

- Specifies runtime controls for APIs (input validation, logging).
- Outlines API abuse monitoring and automated alerting.

Software Supply-Chain Integrity & Provenance
Example: ISAU-DS-CPW-1080: Artifact Integrity, Signing & Attestation for CPW

- Requires signing/attestation for images/functions; verify-on-pull/verify-before-start at admission.
- Maintains approved registry/namespace lists; prohibits floating tags in production.
- Stores SBOMs with artifacts; blocks deployment on missing/invalid provenance.
- Defines exception handling with sunset dates and automated verification in CI and at deploy.

Table C-5. Example Future Sub-Standards

Sub-Standard ID	Sub-Standard Name	Focus Area
ISAU-DS-CPW-1010	Hardened Configuration for Platform & Containerized Workloads	Platform & Workload Hardening
ISAU-DS-CPW-1020	Runtime Threat Detection for Platforms & Workloads	Threat Detection & Response
ISAU-DS-CPW-1030	Platform & Workload Identity Lifecycle Management	IAM
ISAU-DS-CPW-1040	Zero Trust Segmentation for CPW Environments	Zero Trust Segmentation
ISAU-DS-CPW-1050	CPW Data Encryption & Key Management	Data Protection & Encryption
ISAU-DS-CPW-1060	Secure Infrastructure as Code for CPW	IaC Governance

Sub-Standard ID	Sub-Standard Name	Focus Area
ISAU-DS-CPW-1070	Secure API & Secrets Management for CPW	API & Runtime Security
ISAU-DS-CPW-1080	Software Supply-Chain Integrity & Provenance for CPW	Supply-Chain Integrity

Each Sub-Standard Will Specify:

- Inputs (Requirements): Preconditions for implementation.
- Outputs (Technical Specifications): Measurable engineering deliverables.
- Validation Methodologies: Testing and verification methods.
- Implementation Guidelines: Practical, scalable, and secure deployment practices.

Development and Approval Process:

- Open Season Submission: Contributors submit proposed sub-standards aligned with CPW-1000 objectives.
- Technical Peer Review: Evaluation by the Technical Fellow Society for validity, accuracy, and applicability.
- Approval and Publication: Approved sub-standards receive formal versioning and publication as authoritative extensions of CPW-1000.

Future Development (Q4 2025 Onward)

- Practitioners can expect detailed technical guidance aligned with CPW-1000.
- Each sub-standard will map to core principles and technical outputs defined in CPW-1000.
- Contributions invited via the Open Season process.

Practitioner Guidance:

As the suite expands, every sub-standard will inherit the engineering discipline, validation rigor, and architectural alignment needed for consistent, defensible, and auditable CPW implementations. Ensure each proposed sub-standard: (1) ties every output to a §12 test and Evidence Pack ID (EP-03.x), and (2) references the §6 controls it operationalizes (for example, admission/verify-before-start, segmentation, secrets handling, telemetry).

Section 12. Verification and Validation

The effectiveness and defensibility of a compute, platform, and workload security architecture must be continuously verified and validated using structured, engineering-grade assessments. While detailed test requirements for specific technologies will live in CPW sub-standards, the Parent establishes the gold-standard expectations below.

> **Verification** confirms the environment has been implemented according to this standard's Requirements (Inputs, §5) and Technical Specifications (Outputs, §6).
>
> **Validation** proves the environment performs under real operating conditions and withstands adversarial testing.

Core Verification Activities

- Confirm §6 controls exist and are enforced in target environments: bastioned administrative paths with MFA/JIT; identity- and label-based micro-segmentation; verify-on-pull/verify-before-start; approved registries/namespaces; secrets delivered via identity; TLS 1.3/mTLS; immutable logging.
- Review host/OS, orchestrator, image, and policy baselines against adopted benchmarks and enterprise baselines; verify admission policies deny non-conformant workloads.
- Verify integration points and contracts: registry ↔ admission controller, mesh ↔ workload identity, KMS/vault ↔ workloads, SIEM/XDR ↔ control-plane audit—confirm that controls do not disrupt business-critical services.
- Peer review architecture diagrams, trust/segmentation maps, admission/network policies, registry policies, and control mappings for completeness and accuracy.

Core Validation Activities

- Perform adversarial testing—penetration testing, red teaming, and BAS/emulation—focused on control-plane abuse, container escape, supply-chain and admission bypass, lateral movement, and egress governance.
- Validate runtime resilience using automated and manual methods aligned to credible attack paths (for example, deny unsigned images at admission; block unapproved egress; quarantine/rollback on high-severity runtime findings).
- Test operational resilience: rollback to a last-known-good signed image, cluster failover of critical services, incident response for registry/key compromise, and privilege-escalation drills on administrative paths.
- Measure performance against targets such as MTTD, MTTC, MTTR, failed-gate rate (admission/CI), drift MTTR, and signature/SBOM coverage.

Required Deliverables

1. **Test Plans and Procedures** — Scope, tooling, and methods for verification and validation phases.
2. **Validation Reports** — Pass/fail results, residual risk ranking, and prioritized remediation.
3. **Evidence Artifacts** — Policy exports; admission/deny logs; image signatures/attestations; SBOM reports; capability/syscall violations; segmentation maps; control-plane audit logs; SIEM/XDR detections—each labeled with an Evidence Pack ID (EP-03.x) and referenced in Table C-5.
4. **Corrective Action Plans** — Time-bounded remediation for findings that must be closed prior to acceptance.

Common Pitfalls to Avoid

- Treating penetration testing as a check-the-box exercise rather than adversary-aware validation of CPW invariants (for example, admission denials, default-deny east–west, non-root execution).
- Missing evidence: tests run, but artifacts are not versioned, immutable, or linked to Table C-5/EP-03.x.
- Skipping continuous validation in dynamic or high-risk areas (new clusters, new image families, policy changes).

Table C-6. Traceability Matrix: Requirements (§5) to Verification/Validation (§12) and Technical Specifications (§6):

Requirement ID	Requirement (summary)	Verification (build-correct)	Validation (works-right)	Related Technical Specs
5.1	Zero Trust Platform & Workload Security	• MFA/JIT enforced; RBAC/ABAC applied • session capture configured	Phishing/token-replay requires step-up; implicit-trust lateral movement is blocked	§6.1 Identity & Access; §6.2 Segmentation
5.2	Platform & shared responsibility alignment.	• Responsibility matrix approved • provider vs. organization controls monitored	Spot checks confirm provider defaults (for example, storage encryption) and organization controls (for example, key rotation) effective	§6.3 Data & Crypto; §6.5 Monitoring/IR
5.3	Automated security enforcement	• WSPM/IaC/PaC gates active • admission/verify-before-start enabled	Unsigned/unattested image denied in staging/production within the target window	§6.4 API & Runtime; §6.5 Monitoring/IR

Requirement ID	Requirement (summary)	Verification (build-correct)	Validation (works-right)	Related Technical Specs
5.4	Segmentation & trust boundary enforcement	• Network and namespace policies deployed • private endpoints set	• BAS shows east–west block rate meets target • unapproved service-to-service calls blocked	§6.2 Segmentation
5.5	Data encryption & compliance	• Encryption on by default • centralized KMS with rotation • data classified/tagged	• Encrypted restore drill passes • transport scans meet policy • key hygiene checks pass	§6.3 Data & Crypto
5.6	Supply-chain integrity & artifact trust	• Sign/attest all artifacts • SBOM present • registry allowlists in place	• Pipeline rejects unknown provenance • production SBOM coverage = 100 % • admission blocks unapproved tags/registries	§6.4 API & Runtime; §6.5 Monitoring/IR
5.7	Administrative access & privileged operations	• Bastion + MFA/JIT • session capture • no standing cluster-admin	• Privilege-escalation simulations require JIT approval • 0 unrecorded privileged sessions in the sample	§6.1 Identity; §6.2 Segmentation
5.8	Baseline & hardening standards	• Host/OS, orchestrator • image baselines active • admission blocks non-conformant	• Spot checks show non-root, read-only filesystem, minimal caps, syscall/cap profiles • exceptions time-bounded	§6.4 Runtime
5.9	Telemetry, logging & evidence readiness	• Required fields/schema present • immutable storage • control-plane audit on	• Correlation (admission denial ↔ image provenance ↔ workload identity) succeeds • integrity checks pass	§6.5 Monitoring/IR

Evidence guidance

Attach plans and procedures; approved diagrams; policy-as-code repositories; admission/verify and registry policies; CI/CD logs; scan/signature/attestation/SBOM reports; segmentation maps; control-plane audit logs; KMS rotation logs; SIEM queries and detections. Label each set with an EP-03.x identifier and reference the corresponding row in Table C-5.

How to use the matrix

- Plan: ensure each §5 item has at least one verification and one validation activity.
- Execute: record the EP-03.x ID for each row when the activity completes.
- Review: when a requirement or §6 specification changes, update linked activities and §6 references to maintain end-to-end traceability.

Practitioner Guidance:

Treat §12 as a continuous engineering function.

- Map every §5 requirement to one verification and one validation in Table C-5, each with a unique EP-03.x.
- Exercise adversary-informed tests that match CPW: control-plane abuse, container escape, registry poisoning, privilege escalation, and lateral movement.
- Track MTTD/MTTC/MTTR, failed-gate rates, drift MTTR, and signature/SBOM coverage; adjust controls and policies accordingly.
- Re-validate management-plane isolation, admission denials, and default-deny east–west after every major change window.

Quick Win Playbook:

Title: Namespace Default-Deny and Egress Allowlists (Production)

Objective: Reduce lateral-movement and exfiltration risk by enforcing default-deny east–west policy and tightly scoped egress allowlists for a single production namespace.

Target: Enforce default-deny east–west and allowlisted egress for one production namespace (§6.2).

	Component/System: Kubernetes namespace (cluster networking and policy engine). **Protects:** Workloads in the namespace from lateral movement and unsanctioned outbound calls. **Stops/Detects:** Internal reconnaissance/scanning, unauthorized service-to-service calls, data exfiltration, command-and-control beacons over open egress. **Action:** Apply namespace policies (default-deny ingress/egress); permit only DNS/KMS/approved services; run a lateral-movement/egress smoke test (blocked vs allowed paths). **Proof:** Policy manifests, deny-event logs, and smoke-test outputs → Evidence Pack EP-03.71 (and reference Table C-5, row 5.4). **Metric:** Unauthorized lateral/egress attempts are denied and logged within target MTTD; allowlisted traffic passes. **Rollback:** Restore the previous policy commit; remove temporary allow entries; record any exception with owner and expiry.

Section 13. Implementation Guidelines

This section does not prescribe vendor-specific tactics. Parent Standards are stable, long-lived architectural foundations. Here, we define how sub-standards and delivery teams must translate the Parent's intent into operational behaviors that are testable, automatable, and auditable for Compute, Platform & Workload (CPW) environments.

Purpose of This Section in Sub-Standards

Sub-standards must use Implementation Guidelines to:

- Translate architectural expectations from the Parent Standard into enforceable run-time and pipeline behaviors (for example, admission/verify-before-start, default-deny east–west).
- Provide platform-agnostic practices that improve adoption, reduce failure, and align with ISAUnited's defensible design philosophy.
- Highlight common failure modes and how to prevent them with measurable gates and checks.
- Offer repeatable patterns (as code) that enforce controls, trust models, and engineering discipline across control planes, hosts/OS, containers/VMs/serverless, and telemetry.

Open Season Guidance for Contributors

Contributors developing sub-standards Must:

- Align all guidance with the strategic posture in this Parent Standard.
- Avoid vendor/product terms; express controls as requirements, tests, and evidence.
- Include lessons learned (what fails, why, and how the test proves it).
- Focus on repeatable engineering patterns, not one-offs.
- Provide a minimal Standards Mapping (Spec/Control → NIST/ISO clause from §8 → Evidence Pack ID EP-03.x).

Technical Guidance

A. Organizing Principles (normative)

1. Everything as code — Policies for segmentation, admission/verify-before-start, identity/cryptography, logging/telemetry, pipelines, and runbooks Must be version-controlled, peer-reviewed, and promoted on protected branches.
2. Gated change — Every merge and deployment Must pass non-bypassable security gates (for example, block unsigned images, deny unapproved registries, require non-root pods) tied to acceptance criteria from §6 and §12.
3. Immutable, reproducible releases — No manual policy or node changes post-build; releases Must be reproducible and verified at deploy/admission.
4. Least privilege & JIT — Pipeline identities, automation runners, and administrators Must use scoped permissions with time-bounded elevation; break-glass is exceptional and fully audited.
5. Environment parity — Staging Must mirror production controls (authentication/authorization, egress, TLS/mTLS, logging schema, admission policies) so test results are predictive; drift Must be monitored and reconciled.

B. Guardrails by Pipeline Stage (normative)

1. **Pre-commit / local**
   - Secrets scanning and signed commits required.
   - Pre-commit hooks Should run linters and policy checks for IaC/PaC and network/admission definitions.

2. **Pull request (PR) / code review**
   - CODEOWNERS approval required; record a Threat-Model Delta for significant changes.
   - IaC/PaC gate (or equivalent) for segmentation, identity, cryptography, logging, egress, and admission baselines; Critical findings = 0.
   - Include evidence pointers in the PR (planned §12 tests and EP-03.x ID stubs).

3. **Build & package**

- Deterministic artifacts (pinned versions; no ad-hoc fetch at deploy); artifacts signed/attested.
- Integrity verified before promotion; review transitive dependencies for pipeline components.

4. **Pre-deploy / release**
   - Configuration-drift detection against approved baselines; change approvals “as code.”
   - Progressive rollout (staged/canary) for network and admission policies with health SLOs and automatic rollback.
   - Negative/positive traffic-contract tests for inter-service flows; egress allowlist tests per namespace/zone.

5. **Deploy & runtime**
   - TLS 1.3 at edges; mTLS for service-to-service/administrative paths where required; certificates managed via PKI/KMS with rotation.
   - Verify-on-pull / verify-before-start for images/functions; allow only approved registries/namespaces; prohibit floating tags in production.
   - Namespace default-deny; explicit egress allowlists; runners/automation isolated with restricted outbound.
   - Unified logging schema (timestamp, actor, action, resource, result, trace_id, control_id, env); logs to immutable storage with authenticated time sync.
   - Management-plane isolation with bastion + MFA/JIT + session recording.

6. **Post-deploy validation & operations**
   - Continuous validation (BAS/adversary-emulation scenarios) scheduled; failover/DR drills; rollback to last-known-good signed image on high-severity findings.
   - Track Security SLOs: target MTTD/MTTC/MTTR, segmentation block-rate, admission failed-gate rate, drift MTTR, signature/SBOM coverage.
   - Auto-generate an Evidence Pack per release (policy diffs, validation results, admission/deny logs, image-verify events, segmentation maps, drift reports, ADR links).

C. Identity, Secrets, and Keys (normative alignment to §6)

- Use KMS for key storage; define issuance/rotation/revocation; maintain service/workload identity inventories.
- Use short-lived credentials for pipelines and bastions; scope secrets to job/environment; redact in logs.
- No secrets in repositories or images; inject at runtime; full auditability of access.

D. Supply-Chain Integrity (normative)

- Deploy only signed, verified configurations and images from trusted sources; restrict registries/repositories and namespaces.

- Quarantine and verify third-party artifacts (scripts, modules); enforce license and integrity checks.
- Separate build and deploy identities; forbid production writes from build jobs; enforce admission deny for unknown provenance.

E. Measurement & Acceptance (aligned to §6 and §12)

- mTLS coverage for designated paths meets target; certificate inventory is current with no expirations inside the policy window.
- Zone/namespace egress: default-deny enforced; allowlisted destinations only; exceptions time-bounded with approvals.
- Admission baselines: non-root UID, read-only root filesystem, minimal capabilities; privileged/host mounts disallowed unless approved with expiry.
- Logging & evidence: authenticated time sync; required fields present; retention immutable; each change linked to an Evidence Pack ID (EP-03.x) tying §5 → §6 → §12.

Common Pitfalls (and the engineered countermeasure)

1. Pipelines as suggestions → Enforce non-bypassable gates; block merges/releases on fails; store failing artifacts as proof.
2. One-time scanning → Treat checks as gates with thresholds; require coverage for changed items and admission/verify events.
3. Manual hot-fixes/drift → Detect and reconcile drift; forbid out-of-band edits; require ADRs and rollback plans.
4. Open egress / shared runners → Isolate runners; restrict outbound; allowlist per zone/namespace.
5. Management-plane exposure → Bastion-only with MFA/JIT; block direct access from production subnets.
6. Weak cryptography / stale certificates → Enforce TLS 1.3/mTLS where required; rotate and monitor via PKI/KMS.
7. Incomplete logging/time → Enforce unified schema, authenticated time sync, immutable retention.
8. No evidence → Every release Must have an Evidence Pack ID with linked tests and results.

ISAUnited encourages organizations to utilize these guidelines as foundational references for continuous improvement. Although detailed technical instructions and controls will be elaborated in subsequent sub-standards, consistent application of these guidelines will significantly enhance the cloud security posture and ensure operational resilience.

Practitioner Guidance:

- Treat these guidelines as operational defaults; exceptions require written justification and time-bounded compensating controls.
- Map each practice to a §5 readiness input, a §6 output, and a §12 test; assign an EP-03.x for traceability.
- Maintain a single source of truth (diagrams, policies, repositories); review quarterly or after major architectural change.
- Enforce fail-closed CI/CD and admission gates on missing MFA/JIT, segmentation/admission policies, encryption settings, or PaC checks.
- Record owners and approvers for every change; require two-person review for privileged changes.
- Capture before/after diffs and attach them to the Evidence Pack to support verification and audits.

Quick Win Playbook:

Title: Management-Plane Isolation with Bastion and JIT

Objective: Eliminate direct access to platform control planes, require bastion-mediated MFA and time-bounded JIT elevation, and record every privileged session for audit and forensics.

Target: Enforce management-plane isolation with bastion + MFA/JIT + session capture for one platform/control plane (§6.1, §6.2).

Component/System: Control-plane administrative paths (cluster API, hypervisor/management consoles).

Protects: Management interfaces from direct exposure and credential-only compromise.

Stops/Detects: Direct API hits from production/workload subnets, stolen-credential reuse without step-up, unrecorded privileged activity.

Action: Deny direct administrative access from restricted networks; force bastion path; disable standing admin; smoke test: direct attempt = deny, bastion + JIT = allow and record.

Proof: Access-policy diff, deny log for direct attempt, bastion/JIT configuration snapshot, and session-recording excerpt → Evidence Pack EP-03.93 (reference Table C-5, row 5.7).

Metric: 100 % of privileged sessions traverse the bastion with MFA/JIT; 0 direct management-plane connections from restricted networks; 100 % of privileged

	sessions recorded. **Rollback:** Restore the previous access policy from version control under a time-bounded exception; retain artifacts in EP-03.93 as superseded.

Appendices

Appendix A. Engineering Traceability Matrix:

Req ID	Requirement (Inputs) (§5)	Technical Specifications (Outputs) (§6)	Core Principles (§7)	Control Mappings (§9)	Verification – Build Correct (§12)	Validation – Works Right (§12)	Evidence Pack ID
5.1	Zero Trust Platform & Workload Security	§6.1 Identity & Access Security; §6.2 Segmentation	RP-02 Zero Trust; RP-01 Least Privilege	CIS 6.5; CSA CCM IAM-05 / IAM-09	MFA/JIT enforced; RBAC/ABAC applied; session capture configured	Phishing/token-replay requires step-up; implicit-trust lateral movement blocked	EP-03.3
5.2	Platform & Shared Responsibility Alignment	§6.3 Data Protection & Crypto; §6.5 Monitoring & IR	RP-05 Secure by Design; RP-15 Evidence	CIS 4.3; CSA CCM DSI-03	Responsibility matrix approved; provider vs org controls monitored	Spot checks confirm provider defaults (e.g., encryption) and org controls (e.g., key rotation) effective	EP-03
5.3	Automated Security Enforcement	§6.4 API & Runtime Security; §6.5 Monitoring & IR	RP-10 Secure Defaults; RP-12 Security as Code	CIS 4.3; CIS 13.1; CSA CCM IVS-09	WSPM/IaC/PaC gates active; admission/verify-before-start enabled	Unsigned/unattested image denied in stage/prod; auto-remediation meets target window	EP-03.6
5.4	Segmentation & Trust Boundary Enforcement	§6.2 Segmentation	RP-04 Defense in Depth; RP-06 Minimize Attack Surface	CIS 13.1; CSA CCM IVS-06	Network & namespace policies deployed; private endpoints set	BAS shows east-west block rate meets target; unapproved service-to-service calls blocked	EP-03.4

Req ID	Requirement (Inputs) (§5)	Technical Specifications (Outputs) (§6)	Core Principles (§7)	Control Mappings (§9)	Verification – Build Correct (§12)	Validation – Works Right (§12)	Evidence Pack ID
5.5	Data Encryption & Compliance	§6.3 Data Protection & Encryption	RP-18 Protect Confidentiality; RP-19 Integrity	CSA CCM DSI-03	Encryption on by default; KMS rotation + key hygiene checks	Encrypted restore drill passes; transport scans meet policy; certificate/key hygiene validated	EP-03.5
5.6	Supply-Chain Integrity & Artifact Trust	§6.4 API & Runtime Security; §6.5 Monitoring & IR	RP-05 Secure by Design; RP-10 Secure Defaults	CIS 4.3; OWASP ASVS V2.1; CSA CCM IVS-09	Signing/attestation validated; SBOM present; registry allowlists verified	Pipeline rejects unknown provenance; admission blocks unapproved registries/tags; SBOM coverage = 100%	EP-03.8
5.7	Administrative Access & Privileged Operations	§6.1 Identity & Access; §6.2 Segmentation	RP-01 Least Privilege; RP-02 Zero Trust; RP-10 Secure Defaults	CIS 6.5; CSA CCM IAM-09	Bastion + MFA/JIT enforced; session capture validated	Privilege-escalation simulations require JIT; 0 unrecorded privileged sessions	EP-03.3
5.8	Baseline & Hardening Standards	§6.4 Runtime Security	RP-06 Minimize Attack Surface; RP-10 Secure Defaults	CIS 4.3; OWASP A04:2021	Host/OS/orchestrator/image baselines active; admission blocks non-conformant workloads	Runtime checks confirm non-root, read-only FS, minimal caps, syscall/cap profiles; exceptions expire	EP-03.1

Req ID	Requirement (Inputs) (§5)	Technical Specifications (Outputs) (§6)	Core Principles (§7)	Control Mappings (§9)	Verification – Build Correct (§12)	Validation – Works Right (§12)	Evidence Pack ID
5.9	Telemetry, Logging & Evidence Readiness	§6.5 Monitoring & IR	RP-15 Evidence Production; RP-20 Availability	CIS 14.4; CSA CCM DSI-03	Required fields/schema present; immutable storage validated	Correlation succeeds (admission denial ↔ image provenance ↔ workload identity); integrity checks pass	EP-03.2

Appendix B. EP-03 Summary Matrix – Evidence Pack Overview:

Layer	EP Identifier	Purpose	Evidence Categories Included
Parent EP	EP-03	Serves as the master Evidence Pack for the D03 Parent Standard. Stores platform-wide compute, host, orchestrator, and workload architecture evidence; golden images; invariants; and global V&V artifacts supporting §§5, 6, 10, and 12.	• Compute/platform architecture diagrams • Cluster/topology maps (control-plane, nodes, namespaces) • Trust boundaries & segmentation maps • Invariants register • Interface Control Documents (ICDs) • Golden image catalog & SBOM coverage • Admission/verify-before-start policies • Control-plane audit logs and configuration exports • Parent-level V&V evidence (Table C-6)
Sub-EP	EP-03.1	Supports future Sub-Standard CPW-1010: Hardened Configuration for Platform & Containerized Workloads.	• Host/OS baselines • Hardened image baselines • CIS/orchestrator benchmark outputs • Drift detection logs • Admission denials for baseline violations
Sub-EP	EP-03.2	Supports future Sub-Standard CPW-1020: Runtime Threat Detection & Response.	• Runtime agent telemetry • Behavioral anomaly detections • Quarantine/rollback events • Syscall/capability violation logs • Policy enforcement evidence
Sub-EP	EP-03.3	Supports future Sub-Standard CPW-1030: Platform & Workload Identity Lifecycle Management.	• Workload/service identity assignments • Key/credential issuance logs • JIT/MFA privileged access evidence • Identity drift detection reports • Access review outcomes
Sub-EP	EP-03.4	Supports future Sub-Standard CPW-1040: Zero Trust Segmentation for CPW Environments.	• Namespace/app-level network policies • East–west block test results • Micro-segmentation maps

Layer	EP Identifier	Purpose	Evidence Categories Included
			• Private endpoint enforcement evidence • BAS/ATT&CK segmentation tests
Sub-EP	EP-03.5	Supports future Sub-Standard CPW-1050: CPW Data Encryption & Key Management.	• TLS/mTLS enforcement scans • KMS rotation logs • Key usage audit trails • Encryption configuration evidence • Data classification/tagging proof
Sub-EP	EP-03.6	Supports future Sub-Standard CPW-1060: Secure Infrastructure-as-Code (IaC) Governance.	• IaC repos & policy-as-code validation • IaC scan reports • Drift detection and remediation logs • Pipeline gates (fail-closed) evidence
Sub-EP	EP-03.7	Supports future Sub-Standard CPW-1070: Secure API & Secrets Management for CPW.	• API gateway configs • Authentication/authorization logs • Secret vault access logs • Serverless secret-delivery evidence • Rotated secret/credential audit data
Sub-EP	EP-03.8	Supports future Sub-Standard CPW-1080: Software Supply-Chain Integrity & Provenance.	• Image signatures/attestations • SBOM coverage reports • Registry allowlist evidence • Verify-before-start logs • Denials of unsigned artifacts
Sub-EP (Expansion)	EP-03.9+	Additional child Evidence Packs reserved for future sub-standards (for example, workload anomaly detection, elasticity-safe controls, platform DR).	• Will inherit the same EP structure and add new domain-specific evidence categories as CPW sub-standards mature

Adoption References

NOTE: ISAUnited Charter Adoption of External Organizations.

ISAUnited formally adopts the work of the International Organization for Standardization / International Electrotechnical Commission (ISO/IEC) and the National Institute of Standards and Technology (NIST) as foundational standards bodies, and the Center for Internet Security (CIS), the Cloud Security Alliance (CSA), and the Open Worldwide Application Security Project (OWASP) as security control–framework organizations. This adoption aligns with each organization's public mission and encourages use by practitioners and institutions. ISAUnited incorporates these organizations into its charter so that every Parent Standard and Sub-Standard is grounded in a common, defensible foundation.

a) **Foundational Standards (Parent level).**
ISAUnited adopts *ISO/IEC* and *NIST* as foundational standards organizations. Parent Standards align with these bodies for architectural grounding and auditability, and extend that foundation through ISAUnited's normative, testable specifications. This alignment does not supersede *ISO/IEC* or *NIST*.
b) **Security Control Frameworks (Control level).**
ISAUnited adopts *CIS, CSA*, and *OWASP* as control framework organizations. Control mappings translate architectural intent into enforceable technical controls within Parent Standards and Sub-Standards. These frameworks provide alignment at the implementation level rather than at the foundational level.
c) **Precedence and scope.**
Foundational alignment (*ISO/IEC, NIST*) establishes the architectural baseline. Control frameworks (*CIS, CSA, OWASP*) provide enforceable mappings. ISAUnited's security invariants and normative requirements govern implementation details while remaining consistent with the adopted organizations.
d) **Mapping.**
Each cited control mapping is tied to a defined output, an associated verification and validation activity, and an Evidence Pack ID to maintain end-to-end traceability from requirement to control, test, and evidence.
e) **Attribution.**
ISAUnited cites organizations by name, respects attribution requirements, and conducts periodic alignment reviews. Updates are recorded in the Change Log with corresponding evidence.
f) **Flow-downs.**

(Parent → Sub-Standard). Parent alignment to the International *ISO/IEC* and *NIST* flows down as architectural invariants and minimum requirements that Sub-Standards must uphold or tighten. Parent-level mappings to *CIS, CSA*, and *OWASP* flow down as implementation control intents that Sub-Standards must operationalize as controls-as-code, tests, and evidence. Each flow-down shall reference the Parent clause, the adopted organization name, the Sub-Standard clause that implements it, the associated verification/validation test, and an Evidence Pack ID for traceability. Any variance requires a written rationale, compensating controls, and a time-bounded expiry recorded with an Evidence Pack ID.

Change Log and Revision History

Review Date	Changes	Committee	Action	Status
December 2025	Standards Revision	Standards Committee	Publication	Pending
November 2025	Standards Submitted	Technical Fellow Society	Peer review	Pending
October 2025	Standards Revision	Task Group ISAU-TG39-2024	Draft submitted	Complete
December 2024	Standards Development (Parent D01)	Task Group ISAU-TG39-2024	Draft complete	Complete