OWASP AISVS

ARTIFICIAL INTELLIGENCE SECURITY VERIFICATION STANDARD

Artificial Intelligence Security Verification Standard

VERSION 1.0

June 2026

OWASP®

Open Worldwide Application Security Project (OWASP) Foundation

Table of Contents

Frontispiece

About the Standard

The **Artificial Intelligence Security Verification Standard (AISVS)** is a community-driven catalogue of testable security requirements for AI-enabled systems. It gives data scientists, MLOps engineers, software architects, developers, testers, security professionals, tool vendors, regulators, and consumers a common language for specifying security controls across the AI lifecycle. That lifecycle spans data collection and model development through deployment, monitoring, and retirement. With a shared vocabulary, organizations can measure and improve the resilience, privacy, and safety of their AI solutions.

Every requirement in AISVS has been developed from the ground up to reflect the AI threat landscape. While AISVS draws inspiration from broader security best practices, it is purpose-built for artificial intelligence systems and complements (rather than duplicates) governance frameworks such as NIST AI RMF and ISO/IEC 42001.

Copyright and License

Acknowledgments

AISVS v1.0 is the result of a collaborative effort by its project leads, working group members, and community contributors. We thank everyone who has contributed requirements, reviews, and feedback to make this standard possible.

Project Leads

- Jim Manico (jmanico)
- Otto Sulin (ottosulin)
- Rico Komenda (RicoKomenda)
- Russ Memisyazici (vtknightmare)
- Raza Sharif (razashariff)

Contributors and Reviewers

The list below reflects authored and edited content. It does not fully capture contributors whose impact came mainly through reviews and issue discussion, and we thank them as well.

- b1oo (b1oo)
- Jim Schwoebel (jim-schwoebel)
- Vineeth Sai Narajala (vineethsai)
- RL Thornton (thornshadow99)
- Almog Langleben (almogbhl)
- Khalid Al-Amri (khalidwalidalamri)
- Barno Kaharova (BarnoKa)
- Joshua Beck (Josh-Beck)
- Vishal Jindal (vishaljindal1990)
- Stefan Aeschbacher (imix)
- DotDotSlash (DotDotSlash)
- Tetsuo Seto (tetsuoseto)
- Tametomo (Tametomo)
- Martin Bjerke (kattn)
- Deepak Pandey (deepakrpandey12)
- RogueValley (RogueValley)
- emmanuelgjr (emmanuelgjr)
- Cyril Mathew (cy10101)
- mattijs moens (mattijsmoens)
- Jerry Hoff (jerryhoff)
- Ronald Robertson (Treyrob3)
- Vatsal Gupta (vatsalgupta)
- Zoe Braiterman (zbraiterman)
- Ralph Andalis (csfreak92)
- Uncle Joe (sydseter)
- Stuart Small (stusmall)
- Boone Carlson (KeystoneSmartQuotes)
- Joe-B-Security (Joe-B-Security)
- hackwither (hackwither)
- Mayur Agnihotri (Mayur021)
- Mohamad Khalil Yossif (MohamadKhalilYossif)
- William Jawad (wiljav)
- Hari Mukundhan (harimukundhan)
- Sandhya (sandhya13r)
- Starr Brown (mamicidal)

Preface

Welcome to the **Artificial Intelligence Security Verification Standard (AISVS) version 1.0**.

By adopting AISVS, organizations can systematically evaluate and strengthen the security posture of their AI systems, building a foundation of secure AI engineering practices that evolves alongside the technology itself.

Why AISVS Exists

AI systems introduce security risks that traditional application security standards were not designed to address. Prompt injection allows attackers to override model instructions through crafted inputs, turning a language model into a tool for data exfiltration, unauthorized actions, or bypassing safety controls. Training data can be poisoned to install backdoors or degrade model behavior. Models can be extracted, inverted, or manipulated through adversarial inputs. Autonomous agents can take actions with real-world consequences, acting on prompt-injected instructions they cannot tell apart from legitimate ones. Retrieval pipelines can be exploited to leak sensitive information or to inject malicious content into model context. The supply chain for models, datasets, and frameworks presents novel integrity challenges that existing software composition analysis alone cannot solve.

AISVS was created to give organizations a structured, testable set of security controls purpose-built for these risks. It does not replace existing standards; it fills the gap that none of them cover.

Design Principles

AISVS is organized into 12 control families. Each control family is divided into focused sections that support its control objective. Each section contains verification requirements. AISVS defines three verification levels, defined under Using the AISVS; sections need not include requirements at every level.

Each requirement must address a single concern that can ordinarily be implemented and verified as one technical mechanism. Requirements must not duplicate controls defined elsewhere in AISVS. Higher assurance levels may introduce stricter criteria, but those criteria must be stated as separate requirements. Requirements should use clear, technology-neutral language, referencing specific technologies only as examples where they improve clarity.

Every AISVS requirement follows four design principles derived from the standard's name:

- **Artificial Intelligence.** Requirements must address AI/ML-specific assets, workflows, or runtime behavior, including datasets, models, training and evaluation pipelines, retrieval systems, agents, tools, memory, and inference-time operation. AISVS does not duplicate general application security controls from standards such as ASVS unless the control has AI-specific implementation or verification concerns.
- **Security.** Requirements must mitigate an identifiable security, privacy, or safety risk. Controls that serve only operational, governance, compliance, or business objectives are out of scope.
- **Verification.** Requirements must be objectively verifiable through testing, inspection, or audit. Sufficient implementation guidance or tooling must exist to support both implementation and verification. Purely theoretical, subjective, or aspirational guidance is excluded.
- **Standard.** Requirements must use consistent structure, terminology, and assurance-level semantics so AISVS remains coherent, navigable, and suitable for repeatable assessment.

Using the AISVS

The Artificial Intelligence Security Verification Standard (AISVS) defines security requirements for modern AI applications and services, focusing on aspects within the control of application developers.

The AISVS is intended for anyone developing or evaluating the security of AI applications, including developers, architects, security engineers, and auditors. This chapter introduces the structure and use of the AISVS, including its verification levels, intended use cases, and how it is positioned alongside other security standards.

How to Read This Standard

Chapter Structure

Each of the 12 requirement chapters follows the same format:

- **Control Objective.** A brief statement of the security goal for the chapter.
- **Sections.** Requirements are grouped into related sections, each with a short description of the defense goal.
- **Requirement Tables.** Individual requirements are presented in tables with the following columns:

Column	Meaning
#	Unique requirement identifier (e.g., 1.1.1, 9.3.2).
Description	The requirement text, always beginning with "Verify that" to emphasize testability.
Level	The verification level (1, 2, or 3) indicating the depth of assurance required; see the verification levels below.

Appendices

Three appendices support the core requirements:

- **Appendix A (Glossary)** defines key terms and acronyms used throughout the standard.
- **Appendix B (AI Security Controls Inventory)** is a cross-reference of every defense technique in AISVS, organized by security control category (authentication, authorization, encryption, input validation, and so on) with mappings back to specific requirement identifiers.
- **Appendix C (AI-Assisted Secure Coding)** provides controls for the safe use of AI coding tools during software development.

Artificial Intelligence Security Verification Levels

The AISVS defines three ascending levels of security verification. Each level adds depth and complexity, enabling organizations to tailor their security posture to the risk level of their AI systems.

Organizations may begin at Level 1 and progressively adopt higher levels as security maturity and threat exposure increase. AISVS levels are aligned with ASVS levels and are intended to be applied at the matching ASVS level (see Alignment with ASVS Levels below).

Definition of the Levels

Each requirement in AISVS v1.0 is assigned to one of the following levels:

Level 1 requirements

Level 1 includes the most critical and foundational security requirements. These focus on preventing common attacks that do not rely on other preconditions or vulnerabilities. Most Level 1 controls are either straightforward to implement or essential enough to justify the effort.

Level 2 requirements

Level 2 addresses more advanced or less common attacks, as well as layered defenses against widespread threats. These requirements may involve more complex logic or target specific attack prerequisites.

Level 3 requirements

Level 3 includes controls that are typically harder to implement or situational in applicability. These often represent defense-in-depth mechanisms or mitigations against niche, targeted, or high-complexity attacks.

Alignment with ASVS Levels

AISVS levels are aligned with ASVS levels. Verifying an AI application against AISVS Level *N* assumes the application has also been, or is being, verified against ASVS Level *N*. The two standards are designed to be applied together at matching levels:

AISVS Level	Corresponding ASVS Level	Typical use
1	1	**Baseline security for any AI application that handles untrusted input or operates on data of any sensitivity.**
2	2	**AI applications handling sensitive business data, regulated data, or operating in adversarial contexts.**
3	3	**High-assurance AI applications such as those handling life-safety decisions, critical infrastructure, or highly sensitive personal data.**

If an AISVS requirement appears to overlap with an ASVS requirement, the AISVS version is restated only because it has AI-specific implementation details, attack surface, or evidence that an auditor needs to evaluate differently.

Scope of the AISVS

AISVS is intentionally narrow. It only defines security requirements that are specific to AI and ML systems, or where general security controls have AI-specific nuances that warrant restating. It is not a self-contained security program for an AI application. AISVS assumes that the underlying application, infrastructure, and organizational practices are already verified against established general-purpose standards, and adds the AI-specific layer.

The following are intentionally out of scope and are not duplicated in AISVS chapters:

- **General application security.** Authentication, session management, authorization, transport security, input and output handling for non-AI surfaces, secrets management, file upload handling, error handling, and similar controls are defined by the OWASP Application Security Verification Standard (ASVS).
- **General software supply chain security.** Dependency scanning, version pinning, lockfile enforcement, build provenance, reproducible builds, generic SBOM generation, and CI/CD pipeline integrity are defined by the OWASP Software Component Verification Standard (SCVS), SLSA, and the CIS Controls.
- **General infrastructure and platform hardening.** Container, host, network, cloud, and Kubernetes baseline hardening are defined by the CIS Benchmarks, NIST SP 800-53, NIST SP 800-190, and the NIST Cybersecurity Framework (CSF).
- **General data protection and privacy operations.** Data classification, encryption at rest and in transit, retention scheduling, secure deletion of conventional storage, audit log immutability, and consent management platform operation are defined by ASVS, ISO/IEC 27001, and applicable privacy regulations such as the GDPR.
- **General logging and monitoring.** Log storage access control, retention, backup, encryption, redaction, tamper protection, SIEM integration, and operational telemetry are defined by ASVS and standard observability practice.
- **AI governance and risk management.** Organizational AI governance, AI impact assessments, fairness and ethics documentation, model cards, public transparency reports, and risk-management process design are defined by ISO/IEC 42001, ISO/IEC 23894, and the NIST AI RMF.
- **Vendor-specific guidance.** AISVS is vendor-neutral. It specifies what to verify, not which product to use.

When verifying an AI application against AISVS, the equivalent level of those underlying standards should be verified in parallel.

Cross-References Inside AISVS

AISVS chapters are organized by control family rather than by attack or component. As a result, defending against a given AI threat usually requires applying requirements from several chapters together. For example, defending against prompt injection in an agentic application combines requirements from C2 (input validation), C7 (model behavior), C9 (orchestration and agentic security), C10 (MCP-specific controls), C11 (adversarial robustness), and C12 (detection and logging).

When applying AISVS, treat the standard as a whole and consult Appendix B (AI Security Controls Inventory) for a cross-cutting view of where each defense technique appears.

AISVS Requirements and Scope in Assessments

Requirements can often be assessed using a combination of technical testing and vendor documentation, such as model cards for AI models. Another option is to mark requirements outside the organization's control as out of scope.

C1 Training Data Integrity & Traceability

Control Objective

This chapter addresses protecting the integrity and traceability of training data as it is sourced, handled, and maintained.

C1.1 Training Data Origin & Data Security

Training data origin and security are critical to the trustworthiness of any AI system. Datasets must be sourced from verifiable origins, tracked across their full lifecycle, and protected against tampering, corruption, and poisoning so that unauthorized modification can be detected.

#	Description	Level
1.1.1	**Verify that** training data includes only features, attributes, and fields required for the model's stated purpose.	1
1.1.2	**Verify that** an up-to-date inventory is kept of every training-data source, including its origin, responsible party, license, collection method, intended use constraints, and processing history.	2
1.1.3	**Verify that** data integrity is provided when training data is stored and transferred.	2
1.1.4	**Verify that** integrity monitoring is applied to guard against unauthorized modifications or corruption of training data.	2
1.1.5	**Verify that** datasets are watermarked so their use can be attributed and any unauthorized use detected.	3

C1.2 Data Labeling and Annotation Security

Labeling and annotation processes must be protected against unauthorized modification, data leakage, and integrity compromise. Annotation platforms should enforce access control, preserve auditability, and protect labeling artifacts and sensitive label content throughout the training pipeline.

#	Description	Level
1.2.1	**Verify that** labeling platforms enforce access controls that restrict who can create, modify, or approve annotations.	1
1.2.2	**Verify that** cryptographic integrity is applied to labeling artifacts.	2
1.2.3	**Verify that** sensitive information in labels is redacted, anonymized, or encrypted before being used in any labeling artifact.	2

C1.3 Training Data Quality and Security Assurance

Quality and security assurance controls help detect corruption, poisoning, labeling errors, and exploitable dataset patterns before they affect model behavior. Pipelines should combine automated validation, poisoning detection, label quality checks, and bias analysis.

#	Description	Level
1.3.1	**Verify that** training and fine-tuning pipelines implement poisoning detection techniques to identify potential data poisoning or unintentional corruption in training data.	2
1.3.2	**Verify that** automatically generated labels are subject to confidence thresholds and consistency checks to detect misleading or low-confidence labels.	2
1.3.3	**Verify that** models used in security-relevant decisions are evaluated for bias patterns.	2
1.3.4	**Verify that** disallowed content is detected and removed before training.	2
1.3.5	**Verify that** defenses against clean-label poisoning attacks are implemented.	3

References

- NIST AI Risk Management Framework
- EU AI Act: Article 10: Data & Data Governance
- CISA Advisory: Securing Data for AI Systems
- MITRE ATLAS: Poison Training Data (AML.T0020)
- ISO/IEC 42001:2023 Artificial Intelligence Management System

C2 Input Validation

Control Objective

This chapter addresses validation of all inputs as a first-line defense against prompt injection, one of the most damaging attacks on AI systems.

C2.1 Prompt Injection Defenses

Prompt injection is one of the top risks for AI systems, and defending against it requires a combination of pattern filters, data classifiers, and instruction hierarchy enforcement.

#	Description	Level
2.1.1	**Verify that** input normalization is applied before tokenization or embedding.	1
2.1.2	**Verify that** encoding and representation smuggling in inputs is detected and mitigated. Approved mitigations include canonicalization, strict schema validation, policy-based rejection, or explicit marking.	1
2.1.3	**Verify that** all inputs that could steer model behavior are treated as untrusted and screened by a prompt injection detection ruleset or classifier, with flagged inputs blocked.	1
2.1.4	**Verify that** input length controls prevent content from exceeding the context window. The controls must reject inputs that exceed token limits rather than truncating them.	1
2.1.5	**Verify that** the system implements a character set restriction for all inputs. The restriction must use an allow-list approach that permits only characters that are explicitly required.	1
2.1.6	**Verify that** the system enforces an instruction hierarchy in which system and developer messages override user instructions and other untrusted inputs, even after user instructions have been processed.	2
2.1.7	**Verify that** reserved special tokens are encoded as literal characters and cannot be injected into the model context.	2
2.1.8	**Verify that** the system can detect many-shot jailbreaking patterns.	3

C2.2 Content & Policy Screening

Syntactically valid prompts may still request disallowed content such as policy-violating instructions, harmful material, or restricted information. Input-side content screening prevents such prompts from reaching the model.

#	Description	Level
2.2.1	**Verify that** every prompt is scored by a content classifier for violence, self-harm, hate, and sexual content against configurable thresholds. Prompts that exceed those thresholds are rejected or sanitized before reaching the model context.	1

#	Description	Level
2.2.2	**Verify that** prompt content classification is evaluated for unsupported languages.	1
2.2.3	**Verify that** non-text inputs (image/video/audio) are checked for adversarial perturbations, steganographic payloads, hidden or embedded content, or known attack patterns.	2
2.2.4	**Verify that** coordinated attacks spanning multiple input types (e.g., steganographic payloads in images combined with prompt injection in text) are detected and blocked.	3

References

- OWASP LLM01:2025 Prompt Injection
- LLM Prompt Injection Prevention Cheat Sheet
- MITRE ATLAS: Adversarial Input Detection
- MITRE ATLAS: LLM Prompt Injection (AML.T0051)

C3 Model Lifecycle Management & Change Control

Control Objective

This chapter addresses control of model changes so that unauthorized or unsafe modifications cannot reach production.

C3.1 Model Authorization & Integrity

Only authorized models with verified integrity should reach production environments.

#	Description	Level
3.1.1	**Verify that** a model registry maintains an inventory of all deployed model artifacts and their origin.	1
3.1.2	**Verify that** all model artifacts (weights, configurations, tokenizers, base models, fine-tunes, adapters, and safety/policy models) are cryptographically signed by authorized entities.	2
3.1.3	**Verify that** model cryptographic signatures are verified at deployment admission and on load.	2

C3.2 Model Validation & Testing

Models must pass defined security and safety validations before deployment.

#	Description	Level
3.2.1	**Verify that** models undergo automated input validation testing, safety evaluation testing, and output sanitization testing before deployment.	1
3.2.2	**Verify that** models subjected to post-training quantization are re-evaluated against the same safety and alignment test suite on the compressed artifact before deployment.	2
3.2.3	**Verify that** provider model, version, or routing changes trigger security re-evaluation before continued use.	3

C3.3 Controlled Deployment & Rollback

Model deployments must be controlled, monitored, and reversible to support lifecycle management.

#	Description	Level
3.3.1	**Verify that** production deployments implement rollout mechanisms with automated rollback triggers.	2
3.3.2	**Verify that** rollback capabilities restore the complete model state.	2
3.3.3	**Verify that** model versions running in parallel use isolated runtime state so that AI-specific shared resources are not shared across deployments.	2

C3.4 Secure Development Practices

Model development environments must be separated from production environments.

#	Description	Level
3.4.1	**Verify that** AI-specific runtime components are not shared across environment boundaries (e.g., development, staging, production).	1
3.4.2	**Verify that** model training and fine-tuning environments are isolated from production environments.	2

C3.5 Pipeline Fine-Tuning

Fine-tuning pipelines are high-privilege operations that can alter deployed model behavior at scale. Multi-stage pipelines compound this risk because a compromise at any intermediate stage produces a subtly altered artifact that subsequent stages accept.

#	Description	Level
3.5.1	**Verify that** models used in RLHF fine-tuning are versioned and integrity-verified before use in a training run.	2
3.5.2	**Verify that** RLHF training stages include automated detection of reward hacking or reward model over-optimization.	3
3.5.3	**Verify that** in multi-stage fine-tuning pipelines, each stage's output is integrity-verified before it is consumed by the next stage.	3
3.5.4	**Verify that** fine-tuning checkpoints are registered as distinct artifacts.	3

References

- MITRE ATLAS
- OWASP AI Testing Guide
- NIST SP 800-218A: Secure Software Development Practices for Generative AI
- ISO/IEC 42001:2023 Artificial Intelligence Management System
- NIST AI Risk Management Framework

C4 Infrastructure, Configuration & Deployment Security

Control Objective

This chapter addresses hardening AI-specific infrastructure components against model theft, data leakage, and cross-tenant contamination.

C4.1 AI Workload Sandboxing & Validation

Untrusted AI models must be isolated in secure sandboxes, and sensitive AI workloads protected using trusted execution environments (TEEs) and confidential computing technologies.

#	Description	Level
4.1.1	**Verify that** AI models execute in isolated sandboxes.	1
4.1.2	**Verify that** model artifact loading enforces an explicit allow-list of serialization formats that do not permit arbitrary code execution during deserialization.	1
4.1.3	**Verify that** workload attestation is performed before model loading to provide proof that the execution environment has not been tampered with.	3
4.1.4	**Verify that** confidential inference services protect model weights during runtime through isolated execution environments.	3

C4.2 AI Hardware Security

AI-specific hardware components, including GPUs, TPUs, and specialized AI accelerators, must be secured.

#	Description	Level
4.2.1	**Verify that** AI accelerator (GPU) firmware is version-pinned, signed, and attested at boot.	2
4.2.2	**Verify that** execution within a trusted execution environment (TEE) provides hardware-enforced isolation, memory encryption, and integrity protection.	3
4.2.3	**Verify that** AI accelerator (GPU) integrity is validated using hardware-based attestation mechanisms before each workload executes.	3
4.2.4	**Verify that** accelerator (GPU) memory is isolated between workloads through partitioning mechanisms with memory sanitization between jobs.	3
4.2.5	**Verify that** accelerator interconnects are restricted to approved topologies and authenticated endpoints.	3

C4.3 Edge & Distributed AI Security

Distributed AI deployments, including edge computing, federated learning, and multi-site architectures, must be secured.

#	Description	Level
4.3.1	**Verify that** edge AI devices authenticate to central infrastructure using strong authentication mechanisms.	1
4.3.2	**Verify that** models deployed to edge or mobile devices are cryptographically signed during packaging, and that the on-device runtime validates these signatures or checksums before loading or inference.	2
4.3.3	**Verify that** inference runtimes enforce process, memory, and file access isolation.	3
4.3.4	**Verify that** model weights and sensitive parameters stored locally are encrypted using hardware-backed key stores or secure enclaves.	3
4.3.5	**Verify that** models packaged within mobile, IoT, or embedded applications are encrypted at rest, and decrypted only inside a trusted runtime or secure enclave, preventing direct extraction from the app package or filesystem.	3

References

- NIST AI Risk Management Framework
- NIST SP 800-190: Application Container Security Guide
- NSA/CISA Kubernetes Hardening Guidance
- Confidential Computing Consortium

C5 Access Control & Identity for AI Components & Users

Control Objective

This chapter addresses access control challenges that AI systems introduce beyond traditional application security.

C5.1 Authentication

AI agents and human users accessing resources must be properly authenticated and authorized for their level of access.

#	Description	Level
5.1.1	**Verify that** high-risk AI operations (model deployment, weight export, training data access, production configuration changes) require step-up authentication.	3
5.1.2	**Verify that** AI agents in federated or multi-system deployments authenticate using short-lived, minimal-scoped, cryptographically signed tokens.	3

C5.2 AI Resource Authorization & Classification

The caller's authorization context must be enforced through AI-specific query pipelines (RAG retrieval, embedding lookups, inference chains) so the system does not return data the caller is not entitled to access.

#	Description	Level
5.2.1	**Verify that** every AI resource (datasets, endpoints, vector collections, embedding indices, compute instances) enforces access controls with explicit allow-lists and default-deny policies.	2
5.2.2	**Verify that** retrieval pipelines (e.g., RAG queries, embedding lookups) enforce the end-user's authorization context at each retrieval and assembly stage, rather than relying solely on the service account's permissions.	2
5.2.3	**Verify that** sensitive data is retrieved via retrieval pipelines (e.g., RAG queries, embedding lookups) to prevent permanent storage in models.	2
5.2.4	**Verify that** post-inference filtering mechanisms prevent responses from including data that the requester is not authorized to receive.	2
5.2.5	**Verify that** the policy decision point for agent authorization is isolated from the agent's execution environment.	2
5.2.6	**Verify that** privileged access to model weights, training pipelines, and production AI configuration is granted just in time, with a defined maximum session duration and automatic expiry. Zero Standing Privilege (ZSP) to these resources is encouraged.	3

#	Description	Level
5.2.7	**Verify that** data classification labels propagate to downstream resources (embeddings, prompt caches, model outputs).	3

C5.3 Multi-Tenant Isolation

Cross-tenant information leakage through AI-specific shared infrastructure, such as inference caches and shared model state, must be prevented.

#	Description	Level
5.3.1	**Verify that** shared model serving infrastructure prevents one tenant's fine-tuning, inference, or embedding operations from influencing or observing another tenant's operations.	2
5.3.2	**Verify that** one tenant cannot influence or observe another tenant's operations through shared compute resources. Satisfying this requirement typically requires hardware partitioning, confidential computing, or dedicated per-tenant compute allocation.	3

References

- NIST SP 800-207: Zero Trust Architecture
- NIST SP 800-63-3: Digital Identity Guidelines
- OAuth 2.1 (IETF Draft)
- OpenID Connect Core 1.0
- I Know What You Asked: Prompt Leakage via KV-Cache Sharing in Multi-Tenant LLM Serving (NDSS 2025)

C6 Supply Chain Security for Models

Control Objective

This chapter addresses defending against AI supply chain attacks that exploit third-party models, frameworks, or datasets to embed backdoors, bias, or exploitable code.

C6.1 Model Artifact Integrity

Third-party model origins must be authenticated and checked for hidden behavior before fine-tuning or deployment, and AI artifacts should be downloaded only from approved sources.

#	Description	Level
6.1.1	**Verify that** models are scanned for malicious code before import.	1
6.1.2	**Verify that** model weights, datasets, and fine-tuning adapters are downloaded only from approved sources.	1
6.1.3	**Verify that** every third-party model artifact can be integrity-verified.	2
6.1.4	**Verify that** models pass a behavioral acceptance test suite before being promoted to any non-development environment.	2

C6.2 AI BOM & Supply Chain Monitoring

Detailed AI-specific bills of materials must be generated and signed, with readiness to respond to supply chain compromise events.

#	Description	Level
6.2.1	**Verify that** every model artifact publishes a version-controlled, machine-readable AI BOM listing datasets, weights, licenses, and data-origin statements.	1
6.2.2	**Verify that** AI BOMs are cryptographically signed before deployment.	2
6.2.3	**Verify that** AI BOM completeness checks fail the build if any component metadata is missing.	2

References

- OWASP LLM03:2025 Supply Chain
- MITRE ATLAS: Supply Chain Compromise
- SBOM Overview: CISA
- CycloneDX: Machine Learning Bill of Materials
- OWASP AIBOM

C7 Model Behavior, Output Control & Safety Assurance

Control Objective

This chapter addresses constraining, validating, and monitoring model outputs so that unsafe, malformed, or high-risk responses cannot reach users or downstream systems.

C7.1 Output Format Enforcement

Model outputs must be structured and validated to reduce downstream injection risk.

#	Description	Level
7.1.1	**Verify that** the application validates all model outputs against a defined schema and rejects any output that does not match.	1
7.1.2	**Verify that** model-generated output is bounded by length limits and termination controls.	1

C7.2 Hallucination Detection & Mitigation

Potentially inaccurate or fabricated content must be detected so unreliable outputs do not reach users or downstream systems.

#	Description	Level
7.2.1	**Verify that** the system assesses the reliability of generated answers using a confidence estimation method.	2
7.2.2	**Verify that** the application automatically blocks answers or switches to a fallback message if the confidence score drops below a defined threshold.	2
7.2.3	**Verify that** for responses classified as high-risk by policy, the system performs an additional verification step.	3

C7.3 Output Safety

Technical controls must detect and remove unsafe content before it is shown to the user.

#	Description	Level
7.3.1	**Verify that** automated classifiers scan every response and block content that matches defined harmful content categories.	1
7.3.2	**Verify that** output filters detect and block responses that disclose system prompt content or backend data.	2

#	Description	Level
7.3.3	**Verify that** model-generated output is prevented from triggering outbound requests.	2
7.3.4	**Verify that** model outputs are checked for hidden, encoded, or misleading content created through homoglyphs, formatting, metadata, or structured fields.	3

C7.4 Source Attribution & Citation Integrity

RAG-grounded outputs must be traceable to their source documents, with cited claims verifiably supported by retrieved content.

#	Description	Level
7.4.1	**Verify that** responses generated using retrieval-augmented generation (RAG) include attribution to the source documents.	1
7.4.2	**Verify that** RAG attributions are derived from retrieval metadata and are not generated by the model, so provenance cannot be fabricated.	1
7.4.3	**Verify that** claims in a RAG response can be traced to the retrieved chunk.	2
7.4.4	**Verify that** generated media is watermarked to prove it was AI-generated.	3

References

- OWASP LLM05:2025 Improper Output Handling
- OWASP LLM06:2025 Excessive Agency
- OWASP LLM09:2025 Misinformation
- NIST AI 600-1: Generative AI Profile (AI RMF Companion)
- MITRE ATLAS

C8 Memory, Embeddings & Vector Database Security

Control Objective

This chapter addresses securing the embeddings and vector stores that act as semi-persistent and persistent "memory" for AI systems through Retrieval-Augmented Generation (RAG).

C8.1 Access Controls on Memory & RAG Indices

Fine-grained access controls and query-time scope enforcement must be applied to every vector collection.

#	Description	Level
8.1.1	**Verify that** vector identifiers and namespaces enforce uniqueness per tenant and prevent cross-tenant collisions.	1
8.1.2	**Verify that** document metadata tags are immutable after the initial write.	2
8.1.3	**Verify that** retrieval operations enforce scope constraints.	2

C8.2 Embedding Sanitization & Validation

Content must be pre-screened before vectorization, and memory writes treated as untrusted input, to prevent ingestion of unsafe payloads.

#	Description	Level
8.2.1	**Verify that** sensitive fields are detected before embedding and are masked, tokenized, or dropped.	1
8.2.2	**Verify that** vectors that fall outside normal clustering patterns are flagged and quarantined before entering production indices.	2
8.2.3	**Verify that** agent outputs and tool outputs are not automatically written to trusted agent memory without explicit source validation.	2
8.2.4	**Verify that** content crafted to manipulate retrieval results is detected and rejected or quarantined before vectorization.	3
8.2.5	**Verify that** new content written to memory is checked for contradictions with what is already stored and that conflicts trigger alerts.	3

C8.3 Memory Expiry & Revocation

Retention and revocation must be explicit and enforceable for memory and RAG indices.

#	Description	Level
8.3.1	**Verify that** expired vectors are excluded from retrieval results.	2
8.3.2	**Verify that** memory can be reset.	2
8.3.3	**Verify that** quarantined content is retained but excluded from all retrieval results.	3

References

- OWASP LLM08:2025 Vector and Embedding Weaknesses
- OWASP LLM04:2025 Data and Model Poisoning
- OWASP LLM02:2025 Sensitive Information Disclosure
- MITRE ATLAS: RAG Poisoning
- MITRE ATLAS: Infer Training Data Membership

C9 Orchestration & Agentic Security

Control Objective

This chapter addresses ensuring autonomous and multi-agent systems execute only authorized, intended, and bounded actions.

C9.1 Execution Budgets, Loop Control, and Circuit Breakers

Runtime expansion (recursion, concurrency, cost) must be bounded, with safe halting on runaway behavior.

#	Description	Level
9.1.1	**Verify that** per-tool quotas and timeouts (e.g., CPU, memory, disk, egress, and execution time) are enforced.	1
9.1.2	**Verify that** per-execution budgets (e.g., max recursion depth, token use, and monetary spend) are configured and enforced by the runtime.	1
9.1.3	**Verify that** a swarm-level kill-switch exists that can halt all active agent instances.	2

C9.2 High-Impact Action Approval and Irreversibility Controls

Privileged, high-impact, or hard-to-reverse agent actions must require trusted approval checkpoints.

#	Description	Level
9.2.1	**Verify that** the agent runtime blocks execution of privileged, high-impact, or irreversible actions until explicit human approval is received and verified.	1
9.2.2	**Verify that** approval requests display canonicalized and complete action parameters, such as diffs, commands, recipients, amounts, resources, and scopes, without truncation or unsafe transformation.	2
9.2.3	**Verify that** each high-impact action has a trusted reversibility classification, such as read-only, reversible, externally reversible, or irreversible.	2
9.2.4	**Verify that** the agent runtime enforces reversibility classifications by blocking, requiring approval, or restricting actions based on their impact and ability to be reversed.	2
9.2.5	**Verify that** any self-modification capability (e.g., prompt rewriting, tool-list changes, parameter updates) is restricted by enforceable boundaries.	2
9.2.6	**Verify that** agentic systems include an AI-augmented review of planned high-risk actions before execution that adds to, and does not replace, the deterministic policy gate.	2
9.2.7	**Verify that** the AI-augmented review mechanism is protected against manipulation by adversarial inputs, and cannot be overridden or bypassed through prompt injection.	2

#	Description	Level
9.2.8	**Verify that** approvals are cryptographically bound to action parameters, requester identity, execution context, and a unique single-use nonce.	3
9.2.9	**Verify that** cryptographic key material or credentials used to issue approvals are isolated from the agent runtime.	3
9.2.10	**Verify that** approval gates for multi-step or multi-agent action chains enforce the highest-impact reversibility classification present anywhere in the chain.	3

C9.3 Component Isolation and Tool Authorization

Tool and plugin execution, loading, and outputs must be constrained to prevent unauthorized system access and unsafe side effects.

#	Description	Level
9.3.1	**Verify that** each tool/plugin executes in a least-privilege sandbox or is otherwise isolated from model operations.	1
9.3.2	**Verify that** tool outputs are validated against schemas.	1
9.3.3	**Verify that** tool manifests declare required privileges, resource limits, and output validation requirements.	2
9.3.4	**Verify that** the runtime enforces the privileges, resource limits, and output-validation requirements declared in tool manifests.	2
9.3.5	**Verify that** components processing untrusted data are isolated from tool-calling capabilities, ensuring that compromised data processing cannot trigger unauthorized tool invocations.	2
9.3.6	**Verify that** there is architectural separation between processing of untrusted tool outputs and agent operations.	2
9.3.7	**Verify that** external resources named in model output are verified against an approved allow-list or registry before the agent installs or invokes them.	2
9.3.8	**Verify that** policy violations trigger automated tool containment.	3

C9.4 Agent and Orchestrator Identity

Every action must be attributable and every mutation detectable.

#	Description	Level
9.4.1	**Verify that** each agent instance has a unique cryptographic identity and authenticates as a first-class principal to downstream systems.	2
9.4.2	**Verify that** agent-initiated actions are cryptographically bound to each step of the execution chain for non-repudiation.	2
9.4.3	**Verify that** agent identity credentials rotate on a defined schedule.	3

#	Description	Level
9.4.4	**Verify that** agent state persisted between invocations is integrity-protected.	3

C9.5 Agent Authorization, Delegation, and Continuous Enforcement

Every action must be authorized at execution time and constrained by scope.

#	Description	Level
9.5.1	**Verify that** agent actions are authorized against fine-grained policies enforced by the runtime that restrict which tools an agent may invoke, and which parameter values it may supply.	2
9.5.2	**Verify that** when an agent acts on a user's behalf, the runtime propagates an integrity-protected, scope-limited token that carries the user's authorization context and is enforced at every downstream call.	2
9.5.3	**Verify that** all access control decisions are enforced by application logic or a policy engine, never by the AI model itself.	2
9.5.4	**Verify that** secrets and credentials required by an agent at runtime are not exposed within the model's observable context, including the context window, system prompts, or tool call parameters.	2
9.5.5	**Verify that** inter-agent task delegation is restricted by an explicit authorization policy.	2
9.5.6	**Verify that** long-running agent sessions re-evaluate current backend authorization policy on every privileged action.	3

C9.6 Shutdown and Graceful Degradation

Shutdown and graceful degradation paths must remain under human control, with mechanisms that stay reliable and are exercised over time.

#	Description	Level
9.6.1	**Verify that** a manual kill-switch mechanism exists to immediately halt AI model inference and outputs.	1
9.6.2	**Verify that** when a human-approval gate is not satisfied within the defined approval time, the system blocks the pending action.	2
9.6.3	**Verify that** kill-switch commands are implemented through an out-of-band channel that is isolated from the agent runtime.	3

References

- OWASP Agentic AI Threats and Mitigations
- OWASP Top 10 for Agentic Applications 2026
- OWASP LLM06:2025 Excessive Agency
- NIST AI 100-1: AI Risk Management Framework (AI RMF 1.0)
- Regulation (EU) 2024/1689 (EU AI Act), Article 14: Human Oversight

C10 Model Context Protocol (MCP) Security

Control Objective

This chapter addresses secure discovery, authentication, authorization, transport, and use of MCP-based tool and resource integrations.

C10.1 Component Integrity

Only trusted MCP components must be used, and locally launched servers must be secured.

#	Description	Level
10.1.1	**Verify that** MCP components are obtained only from trusted sources and cryptographically verified.	1
10.1.2	**Verify that** only allow-listed MCP servers are permitted.	2
10.1.3	**Verify that** locally launched MCP servers run in a least-privilege sandbox with restricted file system, network, and system access.	2

C10.2 Authentication & Authorization

Callers must be authenticated and access to MCP servers authorized, following protocol best practices.

#	Description	Level
10.2.1	**Verify that** MCP servers validate access tokens for each request and do not rely on transport security alone.	1
10.2.2	**Verify that** MCP servers validate the presented access token's issuer, audience, expiration, and scope claims in accordance with OAuth 2.1.	1
10.2.3	**Verify that** MCP servers acting as OAuth 2.1 resource servers do not store or persist access tokens or user credentials.	1
10.2.4	**Verify that** MCP tools/list returns only tools permitted by resource owners' authorized scopes.	2
10.2.5	**Verify that** MCP servers enforce access control on every tool invocation, validating that the user's access token authorizes both the requested tool and the specific argument values supplied.	2
10.2.6	**Verify that** MCP servers ensure all session artifacts are removed when a session terminates.	2
10.2.7	**Verify that** MCP servers do not pass through access tokens received from clients to downstream APIs.	2

C10.3 Secure Transport

MCP communications must be secured following protocol best practices.

#	Description	Level
10.3.1	**Verify that** authenticated, encrypted streamable HTTP is used for MCP transport for remote services.	1
10.3.2	**Verify that** stdio transport is permitted only in controlled local environments.	1
10.3.3	**Verify that** MCP servers validate both the Origin header and the Host header independently on all HTTP-based transports to prevent DNS rebinding attacks.	2
10.3.4	**Verify that** MCP clients enforce a minimum acceptable protocol version and reject initialize responses that propose a version below that minimum.	2
10.3.5	**Verify that** access tokens between the MCP client and server are sender-constrained using mTLS or DPoP.	3

C10.4 Schema, Message, and Input Validation

Schema, message, and input validation must be enforced in both MCP servers and clients.

#	Description	Level
10.4.1	**Verify that** MCP tools/list and tools/call responses are validated against their declared schemas before being injected into the model context.	1
10.4.2	**Verify that** MCP tools/list and tools/call responses are screened for indirect prompt injection before being injected into the model context.	1
10.4.3	**Verify that** MCP servers reject unrecognized or oversized parameters in function calls.	1
10.4.4	**Verify that** all MCP servers enforce strict schema validation.	2
10.4.5	**Verify that** all MCP transports enforce maximum payload size limits.	2
10.4.6	**Verify that** MCP servers sign tool responses with a unique nonce and timestamp so MCP clients can detect replay attempts.	2
10.4.7	**Verify that** MCP clients present users with explicit consent dialogue and cancellation options upon installation of a local MCP server.	2
10.4.8	**Verify that** MCP clients maintain a snapshot of tool definitions and that any change to a tool definition triggers re-approval before the modified tool can be invoked.	3

References

- Model Context Protocol (MCP) Specification
- OWASP MCP Security Cheat Sheet
- NIST SP 800-207: Zero Trust Architecture
- OAuth 2.1 (IETF Draft)
- OWASP Top 10 for Agentic Applications 2026

C11 Adversarial Robustness

Control Objective

This chapter addresses keeping AI systems reliable and abuse-resistant when facing evasion, inference, extraction, or poisoning attacks.

C11.1 Model Alignment, Safety, and Robustness Testing and Training

Model resilience to manipulated inputs designed to cause misclassification or policy bypass must be increased, primarily through adversarial testing and robustness benchmarking.

#	Description	Level
11.1.1	**Verify that** the model has undergone alignment and safety training or fine-tuning to prevent the model from generating disallowed content categories.	1
11.1.2	**Verify that** a version-controlled alignment test suite is run on every model update or release.	1
11.1.3	**Verify that** models are evaluated against known adversarial attack techniques relevant to their modality.	1
11.1.4	**Verify that** models are hardened against adversarial inputs.	2
11.1.5	**Verify that** an automated evaluator measures harmful-content rate and flags regressions beyond a defined threshold.	3

C11.2 Membership-Inference and Model-Inversion Mitigation

The ability to determine whether a specific record was in the training data must be limited, and reconstruction of private training data or sensitive attributes from model outputs prevented.

#	Description	Level
11.2.1	**Verify that** model-inferred sensitive attributes are not directly returned in outputs.	1
11.2.2	**Verify that** inference endpoints enforce per-principal and global rate limits sized to the extraction threat model, and not solely as a generic API throttle.	1
11.2.3	**Verify that** model outputs are calibrated to reduce overconfident predictions.	2
11.2.4	**Verify that** training on sensitive datasets employs differentially-private optimization.	2
11.2.5	**Verify that** membership-inference attack simulations demonstrate that attack accuracy does not exceed random guessing on evaluated data.	3

C11.3 Model-Extraction Defense

Unauthorized model cloning through API abuse must be detected and deterred using rate limiting, query-pattern analysis, and watermarking.

#	Description	Level
11.3.1	**Verify that** query-pattern analysis feeds an extraction-attempt detector.	1
11.3.2	**Verify that** raw model outputs are not directly exposed beyond the application backend, and that externally visible responses are calibrated to the extraction risk level.	2
11.3.3	**Verify that** model watermarking or fingerprinting techniques are applied so that unauthorized copies can be identified.	3
11.3.4	**Verify that** detection of suspected extraction triggers response measures.	3

C11.4 Model Runtime Anomaly Detection

Manipulated, backdoored, or adversarial data entering the model context at inference time via external sources must be identified and neutralized.

#	Description	Level
11.4.1	**Verify that** inputs from external or untrusted sources pass through anomaly detection before model inference.	2
11.4.2	**Verify that** inputs flagged as anomalous trigger gating actions.	2
11.4.3	**Verify that** the safety violation feedback pipeline includes poisoning detection and human review gates to prevent adversarial manipulation of the improvement mechanism.	3

References

- NIST AI 100-2e2023 Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations
- OWASP LLM04:2025 Data and Model Poisoning
- MITRE ATLAS: Evade ML Model (AML.T0015)
- MITRE ATLAS: Backdoor ML Model
- MITRE ATLAS: Extract ML Model

C12 Monitoring, Logging & Anomaly Detection

Control Objective

This chapter addresses real-time and forensic visibility into what the model and other AI components see, do, and return, so that AI-specific threats can be detected and triaged.

C12.1 Request & Response Logging

AI requests and responses must be logged to create an audit trail and support incident response.

#	Description	Level
12.1.1	**Verify that** AI interactions are logged with session context and AI-specific telemetry.	1
12.1.2	**Verify that** safety filtering and policy decisions are logged with sufficient detail to support audit, debugging, and forensic analysis of content moderation systems.	2
12.1.3	**Verify that** log entries for AI inference events follow a structured, interoperable schema that includes at least the model identifier, token usage (input and output), provider name, and operation type.	2
12.1.4	**Verify that** RAG pipeline retrieval events are logged, including the query, documents retrieved, and knowledge source.	2

C12.2 Detection and Alerting

AI-specific attack patterns (jailbreak, prompt injection, model extraction, multi-turn trajectory attacks, covert channels over LLM endpoints) must be detected, and security events enriched with AI-specific context so downstream detection and response systems can act on them.

#	Description	Level
12.2.1	**Verify that** the system detects and alerts on known jailbreak patterns, prompt injection attempts, and adversarial inputs.	1
12.2.2	**Verify that** behavioral anomaly detection identifies unusual conversation patterns, excessive retry attempts, or probing behaviors.	2
12.2.3	**Verify that** custom rules detect AI-specific threat patterns for coordinated jailbreak attempts, prompt injection, and system prompt extraction attempts.	2
12.2.4	**Verify that** extraction-alert events include offending query metadata to support investigation.	2
12.2.5	**Verify that** token usage is tracked at granular attribution levels including per user, per session, per feature endpoint, and per team or workspace.	2
12.2.6	**Verify that** LLM API traffic is monitored for covert-channel indicators and communication signatures to identify malware and command-and-control (C2) activity.	3

C12.3 Model, Data, and Performance Drift Detection

Drift and degradation across model outputs, input distributions, and data schemas must be monitored to identify quality regressions and security-relevant behavioral shifts.

#	Description	Level
12.3.1	**Verify that** data drift detection monitors input distribution changes that may impact model performance, using statistically validated methods matched to the input data type (e.g., KS test or PSI for tabular numeric features, embedding-distance metrics for text or image).	1
12.3.2	**Verify that** hallucination detection monitors identify and flag model outputs that contain factually incorrect, inconsistent, or fabricated information.	2
12.3.3	**Verify that** hallucination rates are tracked as continuous time-series metrics to enable trend analysis and detection of sustained model degradation.	2
12.3.4	**Verify that** unexplained behavioral shifts are distinguished from gradual, expected operational drift.	3

C12.4 Proactive Security Behavior Monitoring

Security threats arising from proactive (agent-initiated) behavior must be detected and prevented, including pre-execution validation, behavior pattern analysis, and audit trails for approval of security-critical actions.

#	Description	Level
12.4.1	**Verify that** autonomous action triggers include proactive behavior-pattern analysis, security evaluation, and threat-landscape assessment.	2
12.4.2	**Verify that** audit logs capture security-critical proactive actions, including approver identity, timestamp, action parameters, and decision outcomes.	2
12.4.3	**Verify that** kill-switch activations and override commands are logged.	2

C12.5 Training Data & Model Lifecycle Audit

The provenance and change history of training data, model artifacts, and knowledge sources must be auditable throughout the AI development lifecycle.

#	Description	Level
12.5.1	**Verify that** dataset lineage records each dataset and its components, including all transformations, augmentations, and merges.	1
12.5.2	**Verify that** all labeling activities are recorded in logs.	1
12.5.3	**Verify that** all model changes generate immutable audit records.	2
12.5.4	**Verify that** every ingested document is tagged at write time with source, writer identity, and timestamp.	2

References

- OWASP Top 10 for LLM Applications 2025
- MITRE ATLAS - Adversarial Threat Landscape for AI Systems
- NIST AI Risk Management Framework (AI RMF 1.0)
- OWASP Agentic AI Threats and Mitigations
- Microsoft Agent Governance Toolkit
- NIST SP 800-207 Zero Trust Architecture

Appendix A: Glossary

This glossary defines key AI, ML, and security terms used throughout the AISVS to ensure clarity and common understanding.

- **Adapter** – A lightweight module (e.g., LoRA, QLoRA) added to a pre-trained model to specialize its behavior on a specific task without modifying the original weights.
- **Adversarial Example** – An input deliberately crafted to cause an AI model to make a mistake, often by adding subtle perturbations imperceptible to humans.
- **Adversarial Robustness** – A model's ability to maintain its performance and resist being fooled or manipulated by intentionally crafted, malicious inputs designed to cause errors.
- **Adversarial Training** – A training technique that augments training data with adversarial examples to improve model robustness against perturbation attacks.
- **Agent** – An AI software system that uses reasoning, planning, and memory to pursue goals and complete tasks on behalf of users, with a degree of autonomy to make decisions, learn, and adapt. Also referred to as Agentic AI.
- **AI BOM (AI Bill of Materials)** – A structured record of all components in an AI system, including models, datasets, weights, hyperparameters, frameworks, and licenses. May follow SPDX or CycloneDX formats. Distinct from a traditional SBOM in that it covers model-specific artifacts beyond software dependencies. Also referred to as AIBOM or MBOM (Model Bill of Materials).
- **Alignment** – The degree to which a model's behavior and outputs match human intentions, values, and safety requirements. Alignment is shaped during training through techniques such as RLHF and Constitutional AI, and is the property that adversarial attacks such as jailbreaks attempt to subvert.
- **AppArmor** – A Linux kernel security module that restricts program capabilities through per-program security profiles, used to sandbox AI workloads.
- **Attention Map** – A visualization of which parts of an input a transformer model attends to when producing an output, used as an interpretability tool.
- **Attribute-Based Access Control (ABAC)** – An access control paradigm where authorization decisions are based on attributes of the user, resource, action, and environment, evaluated at query time.
- **Backdoor Attack** – A type of data poisoning attack where the model is trained to respond in a specific way to certain triggers while behaving normally otherwise.
- **Bias** – Systematic errors in AI model outputs that can lead to unfair or discriminatory outcomes for certain groups or in specific contexts.
- **Bias Exploitation** – An attack technique that takes advantage of known biases in AI models to manipulate outputs or outcomes.
- **Blue-Green Deployment** – A deployment strategy that runs two identical production environments (blue and green), allowing instant rollback by switching traffic between them.

- **Byzantine Fault Tolerance** – The ability of a distributed system to reach consensus and continue operating correctly even when some nodes fail or act maliciously.
- **Canary Deployment** – A deployment strategy that gradually routes a small percentage of traffic to a new model version to detect issues before full rollout.
- **Cedar** – An open-source policy language and evaluation engine for fine-grained permissions, originally created by Amazon. Used in implementing ABAC for AI systems.
- **Certified Robustness** – A formal mathematical guarantee that a model's prediction will not change within a specified perturbation bound around an input, verified through techniques such as interval-bound propagation.
- **Chain of Thought** – A technique for improving reasoning in language models by generating intermediate reasoning steps before producing a final answer.
- **CI/CD (Continuous Integration / Continuous Deployment)** – A software engineering practice that automates building, testing, and deploying code changes, used in AI systems for model and pipeline deployment.
- **Circuit Breaker** – A mechanism that automatically halts AI system operations when specific risk thresholds are exceeded, such as runaway agent loops or budget exhaustion.
- **CMP (Consent Management Platform)** – A system that tracks user consent preferences including opt-in status, purpose, and retention period, and enforces consent decisions across data processing pipelines.
- **Concept Drift** – A change in the statistical relationship between model inputs and outputs over time, causing model predictions to become less accurate even if input distributions remain stable.
- **Confidential Computing** – A security paradigm that protects data in use by performing computation within hardware-enforced trusted execution environments, ensuring code and data remain encrypted and isolated from the host.
- **Confidential Inference** – An inference service that runs AI models inside a trusted execution environment (TEE), ensuring model weights and inference data remain encrypted, sealed, and protected from unauthorized access or tampering.
- **Constitutional AI** – A training approach in which a model is guided by a set of written principles (a "constitution") and trained to critique and revise its own outputs for policy compliance, using a self-critique process as an alternative or supplement to human feedback. See also: RLHF.
- **Context Window** – The maximum amount of text (measured in tokens) that a language model can process in a single inference call, encompassing the system prompt, conversation history, retrieved documents, and tool outputs. The context window defines what information is available to the model at inference time and is a finite resource that can be exhausted or manipulated by adversarial inputs.
- **Counterfactual Explanation** – An interpretability technique that explains a model decision by describing the minimal changes to input features that would change the prediction outcome.
- **Covert Channel** – An unintended communication path that can be exploited to transfer information in violation of security policy, such as through timing or resource usage patterns in shared AI infrastructure.

- **CycloneDX** – An open standard for software and AI bill of materials, supporting component inventory, vulnerability tracking, and license compliance.
- **DAG (Directed Acyclic Graph)** – A graph structure with directed edges and no cycles, used in AI systems to represent agent decision paths, reasoning traces, and workflow dependencies.
- **Data Augmentation** – A technique that creates modified copies of training data (e.g., through rotation, noise addition, or paraphrasing) to increase dataset diversity and improve model robustness.
- **Data Drift** – A change in the statistical distribution of model input data over time compared to the data the model was trained on, potentially degrading prediction quality.
- **Data Leakage** – Unintended exposure of sensitive information through AI model outputs or behavior.
- **Data Lineage** – The documented chain of origin, transformation, and movement of data through an AI system's lifecycle, from collection through preprocessing, training, fine-tuning, embedding, and inference. Lineage records capture source identity, transformation operations, timestamps, and responsible parties, enabling auditability and the removal of data whose provenance cannot be verified.
- **Data Minimization** – The principle of collecting, processing, and retaining only the minimum data necessary for a defined and documented purpose. In AI systems this extends to training data selection, feature engineering, context window construction, retrieval chunk inclusion, and memory and embedding retention policies.
- **Data Poisoning** – The deliberate corruption of training data to compromise model integrity, often to install backdoors or degrade performance.
- **Defense-in-Depth** – A security strategy that layers multiple independent defensive controls so that if one layer fails, others continue to provide protection.
- **Defensive Distillation** – A training technique where a model is trained on the soft probability outputs of another model to smooth decision boundaries and reduce susceptibility to adversarial perturbation.
- **Differential Privacy** – A mathematically rigorous framework for releasing statistical information about datasets while protecting the privacy of individual data subjects, quantified by an epsilon (ε) privacy budget.
- **DoS (Denial of Service)** – An attack that attempts to make a system unavailable by overwhelming it with requests or exhausting its resources.
- **Downgrade (response)** – Returning a model response that is less specific, less personalized, or otherwise reduced in scope when full processing would exceed an authorization or consent boundary. Examples include filtering out retrieval chunks sourced from non-consenting data subjects, suppressing personalized fields, or returning a generic answer instead of one that materially relies on restricted data. Refusal is always a valid downgrade. Acceptable downgrade behaviors should be documented per inference path.
- **DPIA (Data Protection Impact Assessment)** – A formal assessment required under regulations such as GDPR to evaluate and mitigate risks to personal data before processing begins.

- **DPoP (Demonstrating Proof-of-Possession)** – An OAuth 2.0 mechanism (RFC 9449) that binds an access token to a cryptographic key held by the client, so a stolen token cannot be replayed by another party. Used alongside or instead of mTLS to sender-constrain tokens between MCP clients and servers. See also: Sender-Constrained Token, mTLS.
- **DP-SGD (Differentially Private Stochastic Gradient Descent)** – A training algorithm that adds calibrated noise to gradient updates during model training to provide formal differential privacy guarantees.
- **DRTM (Dynamic Root of Trust for Measurement)** – A hardware mechanism that establishes a trusted execution starting point at runtime, enabling integrity verification of AI accelerator workloads.
- **Embedding Inversion** – An attack technique that reconstructs approximate plaintext content from vector embeddings, potentially exposing sensitive information that was assumed to be protected by the embedding transformation. Related: MITRE ATLAS AML.T0024.001. See also: Model Inversion.
- **Embeddings** – Dense vector representations of data (text, images, etc.) that capture semantic meaning in a high-dimensional space.
- **Excessive Agency** – A vulnerability class in which an AI agent is granted more capability, permission, or autonomy than its task requires, allowing benign or manipulated behavior to cause disproportionate harm. Mitigated by least privilege, scoped tools, and human-in-the-loop approval for high-impact actions.
- **Exfiltration** – The unauthorized transfer of data outside a system or security boundary. In AI systems, exfiltration paths include model outputs, covert channels in generated content, tool side effects, and memory or embedding leakage.
- **Explainability** – The ability of an AI system to provide human-understandable reasons for its decisions and predictions, through techniques such as SHAP, LIME, attention maps, and counterfactual explanations. Also referred to as Explainable AI (XAI).
- **Fail-Closed / Fail-Open** – Fail-closed describes a system that defaults to a secure, blocked state when it encounters an error or component failure, preventing uncontrolled operation. Fail-open describes the inverse: operation continues unrestricted on failure. AISVS requires AI components with safety or authorization responsibilities to fail closed.
- **Feature Attribution** – An interpretability method that assigns importance scores to individual input features indicating their contribution to a specific model prediction.
- **Federated Learning** – A machine learning approach where models are trained across multiple decentralized devices holding local data samples, without exchanging the data itself.
- **Fine-tuning** – The process of continuing to train a pre-trained model on a smaller, task-specific dataset to adapt it for a particular use case.
- **FIPS 140-3** – A U.S. government standard that defines security requirements for cryptographic modules, with Level 3 requiring physical tamper-resistance and identity-based authentication.
- **Guardrails** – Constraints implemented to prevent AI systems from producing harmful, biased, or otherwise undesirable outputs.

- **Hallucination** – A phenomenon where an AI model generates incorrect or misleading information that is not grounded in its training data, retrieved context, or factual reality.
- **Homoglyph** – A character that visually resembles another character from a different script or encoding (e.g., Cyrillic "a" vs. Latin "a"), exploited in attacks to bypass text-based input validation.
- **HSM (Hardware Security Module)** – A dedicated physical device that manages, processes, and stores cryptographic keys in a tamper-resistant environment.
- **Human-in-the-Loop (HITL)** – Systems designed to require human oversight, verification, or intervention at crucial decision points.
- **Indirect Prompt Injection** – A prompt injection attack where the malicious instructions are not supplied directly by the user but are embedded in external content the model later consumes, such as a retrieved document, web page, email, or tool output. Because the agent treats that content as trusted context, the injected instructions can hijack its behavior. See also: Prompt Injection.
- **Inference** – The process of running a trained model on new input to produce an output, as distinct from training. Inference time is when the system prompt, user input, retrieved context, and tool outputs are combined and processed, and is the point at which many runtime controls apply.
- **Infrastructure as Code (IaC)** – Managing and provisioning infrastructure through code instead of manual processes, enabling security scanning and consistent deployments.
- **Interval-Bound Propagation** – A formal verification technique that propagates bounds through neural network layers to certify that model predictions are robust within specified input perturbation ranges.
- **Jailbreak** – Techniques used to circumvent safety guardrails in AI systems, particularly in large language models, to produce prohibited content.
- **JIT (Just-in-Time) Privileged Access** – A security practice where elevated permissions are granted only for a short, defined window when needed for a specific task and automatically revoked afterward, minimizing standing privilege exposure.
- **JWT (JSON Web Token)** – A compact, self-contained token format for securely transmitting identity and authorization claims between parties, signed to ensure integrity.
- **k-anonymity** – A privacy property where each record in a dataset is indistinguishable from at least k-1 other records with respect to certain identifying attributes.
- **Kill-Switch** – A mechanism to immediately halt AI model inference, agent execution, or system outputs on command or in response to a safety trigger. Kill-switches for autonomous agents must be delivered through a channel the agent runtime cannot access or suppress, so that a compromised agent cannot block its own shutdown.
- **KMS (Key Management Service)** – A managed service for creating, storing, rotating, and controlling access to cryptographic keys used to protect data and artifacts.

- **Labeling** – The process of assigning classification tags, annotations, or ground-truth values to training data records or fields, performed by human annotators, automated systems, or a combination of both. Labeling encompasses the full annotation pipeline including annotator identity tracking, label integrity verification, and preference data collection for RLHF.
- **l-diversity** – A privacy property extending k-anonymity that requires each equivalence class to contain at least l distinct values for sensitive attributes, preventing attribute disclosure.
- **Least Privilege** – The security principle of granting only the minimum necessary access rights for users and processes.
- **LIME (Local Interpretable Model-agnostic Explanations)** – A technique to explain the predictions of any machine learning classifier by approximating it locally with an interpretable model.
- **Linkage Attack** – An attack that combines quasi-identifiers across multiple datasets to re-identify individuals whose data was supposedly anonymized.
- **LLM (Large Language Model)** – A neural network, typically transformer-based, trained on large text corpora to predict and generate language; the core model type behind most generative AI applications, assistants, and agents.
- **Machine Unlearning** – Techniques to remove the influence of specific training data from a trained model, supporting data subject deletion requests and regulatory compliance.
- **Many-Shot Jailbreaking** – An attack technique that embeds a large number of fabricated user-model exchange pairs in the context window to shift the model's apparent behavioral pattern and override its safety guardrails through accumulated in-context examples.
- **MCP (Model Context Protocol)** – A protocol that enables AI models and agents to access external tools, data sources, and resources by exchanging structured, typed requests and responses over a defined transport.
- **Membership Inference Attack** – An attack that aims to determine whether a specific data point was used to train a machine learning model.
- **MIG (Multi-Instance GPU)** – An NVIDIA technology that partitions a single GPU into multiple isolated instances, each with dedicated memory and compute resources for secure multi-tenant workloads.
- **MITRE ATLAS** – Adversarial Threat Landscape for Artificial-Intelligence Systems; a knowledge base of adversarial tactics and techniques against AI systems.
- **Model Card** – A document that provides standardized information about an AI model's performance, limitations, intended uses, and ethical considerations to promote transparency and responsible AI development.
- **Model Extraction** – An attack where an adversary repeatedly queries a target model to create a functionally similar copy without authorization. Also referred to as model stealing or model theft.
- **Model Inversion** – An attack that attempts to reconstruct training data by analyzing model outputs.

- **Model Lifecycle Management** – The process of overseeing all stages of an AI model's existence, including design, development, deployment, monitoring, maintenance, and eventual retirement.
- **Model Poisoning** – Introducing vulnerabilities or backdoors directly into a model during the training process.
- **mTLS (Mutual TLS)** – A TLS configuration where both client and server authenticate each other using certificates, ensuring bidirectional identity verification for service-to-service communication.
- **Multi-agent System** – A system composed of multiple interacting AI agents, each with potentially different capabilities and goals.
- **NFC (Normal Form Composed)** – A Unicode normalization form that decomposes characters and then recomposes them into a canonical representation, used to prevent encoding-based bypass attacks.
- **Non-repudiation** – A security property ensuring that a party cannot credibly deny having performed an action. In AI systems, achieved through cryptographic signing of agent actions and audit log entries, enabling attribution of decisions to specific principals.
- **NVLink** – A high-bandwidth interconnect technology for GPU-to-GPU communication, requiring authentication and encryption in multi-tenant AI environments.
- **OAuth 2.1** – An authorization framework that consolidates OAuth 2.0 best practices into a single specification, used in AISVS as the required authentication mechanism for MCP clients and servers.
- **OIDC (OpenID Connect)** – An identity layer built on OAuth 2.0 that enables clients to verify user identity based on authentication performed by an authorization server.
- **OPA (Open Policy Agent)** – An open-source, general-purpose policy engine that evaluates authorization and admission control policies written in Rego, enabling unified policy enforcement across applications, APIs, and infrastructure.
- **PDP (Policy Decision Point)** – A component in a policy enforcement architecture that evaluates authorization requests against defined policies and returns an allow or deny decision. In agentic AI systems, the PDP is isolated from the agent's execution environment to prevent a compromised agent from influencing its own authorization decisions.
- **PII (Personally Identifiable Information)** – Any information that can be used to identify, contact, or locate a specific individual, either alone or combined with other data.
- **Policy-as-Code** – The practice of defining security and compliance policies in machine-readable code that can be version-controlled, tested, and automatically enforced in CI/CD pipelines.
- **Privacy-Preserving Machine Learning (PPML)** – Techniques and methods to train and deploy ML models while protecting the privacy of the training data.
- **Prompt Injection** – An attack where malicious instructions are embedded in inputs to override a model's intended behavior.

- **Prompt Template** – A structured text pattern used to construct prompts submitted to an AI model, containing fixed instructions, variable placeholders for user inputs, and formatting directives. Prompt templates are AI-specific configuration artifacts that require version control, integrity protection, and access controls equivalent to source code.
- **Quantization** – A post-training compression technique that reduces model weight precision (e.g., from 32-bit to 8-bit or 4-bit integers) to decrease memory footprint and inference latency. Quantization can alter model behavior, requiring safety and robustness properties to be re-evaluated after application.
- **RAG (Retrieval-Augmented Generation)** – A technique that enhances large language models by retrieving relevant information from external knowledge sources before generating a response.
- **RBAC (Role-Based Access Control)** – An access control model where permissions are assigned to roles rather than individual users, and users are granted access by being assigned to appropriate roles.
- **Red-Teaming** – The practice of actively testing AI systems by simulating adversarial attacks to identify vulnerabilities.
- **Re-identification Risk** – The probability that an individual can be identified from a supposedly anonymized dataset, measured against defined thresholds.
- **Remote Attestation** – A mechanism by which a trusted execution environment provides cryptographic proof to a remote party that specific code is running in a genuine, unmodified TEE.
- **Reward Model** – A machine learning model trained to predict human preference scores for AI outputs, used as a proxy reward signal in RLHF training pipelines. Because reward models are ML artifacts, they are subject to data poisoning attacks that can subvert alignment training outcomes.
- **RLHF (Reinforcement Learning from Human Feedback)** – A training technique where a model is fine-tuned using human preference judgments as a reward signal to improve alignment with human values and safety requirements.
- **SAML (Security Assertion Markup Language)** – An XML-based standard for exchanging authentication and authorization data between identity providers and service providers.
- **Sandboxing** – An isolation technique that confines a process or component to a controlled environment with restricted filesystem access, network egress, and system call permissions. In AI systems, sandboxing is used to contain tool and plugin execution, AI workloads, and third-party model inference to prevent unauthorized host access or cross-tenant contamination.
- **SBOM (Software Bill of Materials)** – A formal record containing the details and supply chain relationships of software components used in building an application. See also AI BOM for model-specific artifacts.
- **Scanned** – Subjected to automated security analysis by a tool, integrated into a pipeline or controlled process (as opposed to an ad-hoc or manual check).
- **SCVS (Software Component Verification Standard)** – An OWASP framework for verifying the security properties of software components, referenced by AISVS for supply chain integrity controls applicable to AI frameworks, libraries, and model dependencies.

- **seccomp (Secure Computing Mode)** – A Linux kernel feature that restricts the system calls a process can make, used to sandbox AI workloads and reduce attack surface.
- **Secure Boot** – A firmware security feature that verifies the cryptographic signature of each component in the boot chain before execution, preventing unauthorized or tampered software from loading.
- **Secure Multi-Party Computation (SMPC)** – A cryptographic technique that enables multiple parties to jointly compute a function over their private inputs without revealing those inputs to each other.
- **SELinux (Security-Enhanced Linux)** – A Linux kernel security module that provides mandatory access controls using security policies, used to enforce fine-grained process isolation for AI workloads.
- **Sender-Constrained Token** – An access token cryptographically bound to the legitimate client so it cannot be used by another party if stolen, through mechanisms such as mTLS or DPoP. AISVS requires sender-constrained tokens between MCP clients and servers. See also: DPoP, mTLS.
- **Sensitive Fields** – Individual data attributes, columns, or record elements within a dataset that contain personal, regulated, or otherwise protected information (e.g., names, identifiers, health data, financial data, or biometric data). Sensitive fields require access controls, minimization, redaction, or encryption. In AI systems, sensitive field detection is required before data is used for training, embedding, or inference to prevent unintentional leakage or memorization.
- **Shadow Deployment** – A deployment pattern in which a new model version receives a copy of live production traffic alongside the current version without serving responses to end users, enabling behavioral comparison and safety validation before promotion.
- **Shadow Model** – A model trained by an attacker to mimic a target model's behavior, used in membership inference attacks and as a baseline for evaluating machine unlearning effectiveness.
- **SHAP (SHapley Additive exPlanations)** – A game theoretic approach to explain the output of any machine learning model by computing the contribution of each feature to the prediction.
- **Side-Channel Attack** – An attack that extracts information from a system through indirect observation of physical characteristics such as timing, power consumption, electromagnetic emissions, or cache behavior, rather than exploiting software vulnerabilities.
- **SIEM (Security Information and Event Management)** – A platform that aggregates, correlates, and analyzes security event data from multiple sources to detect threats, support incident response, and satisfy compliance requirements.
- **SLSA (Supply-chain Levels for Software Artifacts)** – A security framework defining incremental levels of supply chain integrity guarantees, from basic build-process documentation to fully reproducible, hermetically sealed builds with authenticated artifact provenance. Referenced by AISVS for AI model and artifact supply chain controls.
- **SOC (Security Operations Center)** – A team or facility responsible for monitoring, detecting, analyzing, and responding to security incidents. In AISVS, SOC teams consume AI security event logs for correlation, triage, and incident response.

- **SPDX (Software Package Data Exchange)** – An open standard for communicating software and AI component bill of materials information, including component origin, licensing, and security references.
- **SSE (Server-Sent Events)** – A web technology that enables a server to push real-time updates to a client over an HTTP connection, used as a transport mechanism in MCP.
- **stdio (Standard Input/Output)** – A process communication mechanism using standard input, output, and error streams, used in MCP as a local-only transport restricted to single-process, same-machine communication.
- **Steganography** – The practice of hiding data within other media (images, audio, video) in a way that is not apparent to observers, used as an attack vector to smuggle payloads past content filters.
- **Strong Authentication** – Authentication that resists credential theft and replay by requiring at least two factors (knowledge, possession, inherence) and phishing-resistant mechanisms such as FIDO2/WebAuthn, certificate-based service auth, or short-lived tokens.
- **Supply Chain Attack** – Compromising a system by targeting less-secure elements in its supply chain, such as third-party libraries, datasets, or pre-trained models.
- **Synthetic Data** – Artificially generated data that preserves the statistical properties of real data while containing no actual individual records, used to protect privacy during model training and testing.
- **System Prompt** – Instructions supplied to a model by the application or developer that establish its role, constraints, and policies, separate from user input. System prompt content is sensitive: disclosure can reveal guardrails and aid evasion, so AISVS requires output filters to block its leakage. See also: Prompt Template, Context Window.
- **TEE (Trusted Execution Environment)** – A hardware-isolated processing environment that provides confidentiality and integrity guarantees for code and data, protecting them from the host operating system and other tenants.
- **Temperature Scaling** – A post-hoc calibration technique that adjusts model output confidence scores to better reflect true prediction probabilities.
- **TLS (Transport Layer Security)** – A cryptographic protocol that provides end-to-end encryption, authentication, and integrity for data transmitted over a network. AISVS requires TLS 1.3 or later.
- **Tokenizer** – A component that converts raw text into a sequence of tokens (subwords, words, or characters) that a language model can process as input.
- **TPM (Trusted Platform Module)** – A dedicated hardware chip that provides cryptographic functions including secure key generation, storage, and platform integrity measurement.
- **Transfer Learning** – A technique where a model developed for one task is reused as the starting point for a model on a second task.
- **Trust Boundary** – A point where data or control passes between zones that hold different levels of trust, such as from untrusted external input to a more privileged internal component. Flows crossing a trust boundary should be validated, authorized, and monitored, and content entering a higher-trust zone should be treated as untrusted until checked.

- **Vector Database** – A specialized database designed to store high-dimensional vectors (embeddings) and perform efficient similarity searches.
- **VRAM (Video Random Access Memory)** – Memory on a GPU used to store model weights, activations, and intermediate computations during AI inference and training, requiring zeroing between tenant workloads.
- **Vulnerability Scanning** – Automated tools that identify known security vulnerabilities in software components, including AI frameworks and dependencies.
- **WASM (WebAssembly)** – A portable binary instruction format that enables sandboxed execution of code, used as an isolation mechanism for AI tools and plugins.
- **Watermarking** – Techniques to embed imperceptible markers in AI-generated content or model weights to track origin, detect unauthorized copies, or identify AI-generated media.
- **WORM (Write-Once-Read-Many)** – A storage technology that prevents modification or deletion of data after it is written, used for tamper-evident audit logs and backup protection.
- **Zero-Day Vulnerability** – A previously unknown vulnerability that attackers can exploit before developers create and deploy a patch.
- **Zero Standing Privilege (ZSP)** – A security principle requiring that no user, service account, or agent holds persistent elevated permissions. All privileged access is granted just in time for a specific task, scoped to the minimum necessary rights, and automatically revoked after a defined maximum session duration or upon task completion.
- **Zero-Trust** – A security model that assumes no implicit trust for any user, device, or network, requiring continuous verification of identity and authorization for every access request.

Appendix B: AI Security Controls Inventory

Objective

This appendix is a consolidated, developer-facing inventory of the security controls mandated across the AISVS requirements. Controls are grouped by control family so an implementer can find all related defenses in one place, regardless of which chapter defines them, and each control links back to the AISVS requirement IDs that mandate it.

This inventory is non-normative. It reorganizes existing requirements for ease of implementation and does not add, remove, or change any requirement. The requirement chapters (C1 through C12) remain the source of truth. Requirement IDs are written in canonical `C{chapter}.{section}.{requirement}` form (for example, `C5.1.1`). Every numbered requirement in the standard appears in exactly one control family below, so the inventory can be checked for completeness against the chapters.

AD.1 Authentication & Identity

Verify the identity of users, agents, services, edge devices, and MCP clients/servers before granting access.

Control / Technique	Requirement IDs
Step-up authentication for high-risk AI operations (model deployment, weight export, training-data access, production configuration changes)	C5.1.1
Short-lived, minimal-scoped, cryptographically signed tokens for federated or multi-system agent authentication	C5.1.2
Strong authentication of edge AI devices to central infrastructure	C4.3.1
Unique cryptographic identity per agent instance, authenticating as a first-class principal to downstream systems	C9.4.1
Scheduled rotation of agent identity credentials	C9.4.3
MCP per-request access-token validation (not transport security alone)	C10.2.1
MCP access-token claim validation (issuer, audience, expiration, scope) per OAuth 2.1	C10.2.2
MCP resource servers do not store or persist access tokens or user credentials	C10.2.3
Removal of all MCP session artifacts on session termination	C10.2.6
No pass-through of client access tokens to downstream APIs	C10.2.7
Sender-constrained MCP access tokens (mTLS or DPoP)	C10.3.5

Common pitfalls: reusing end-user credentials for agent-to-agent calls; not rotating agent credentials on suspected compromise; treating transport security as a substitute for per-request token validation.

AD.2 Authorization & Access Control

Enforce access decisions across users, agents, tools, and resources using policy that the model cannot override.

Control / Technique	Requirement IDs
Access controls on every AI resource (datasets, endpoints, vector collections, embedding indices, compute) with explicit allow-lists and default-deny	C5.2.1
End-user authorization context enforced at each retrieval and assembly stage, not the service account alone	C5.2.2
Post-inference filtering so responses exclude data the requester is not entitled to receive	C5.2.4
Policy decision point isolated from the agent execution environment	C5.2.5
Just-in-time privileged access to model weights, training pipelines, and production configuration with automatic expiry	C5.2.6
Fine-grained, runtime-enforced authorization of agent actions (which tools, which parameter values)	C9.5.1
Integrity-protected, scope-limited delegation token propagated to every downstream call	C9.5.2
Access-control decisions enforced by application logic or a policy engine, never by the model	C9.5.3
Inter-agent task delegation restricted by an explicit authorization policy	C9.5.5
Re-evaluation of backend authorization on every privileged action in long-running sessions	C9.5.6
Scope-filtered MCP tool discovery (tools/list returns only authorized tools)	C10.2.4
Per-invocation MCP access control validating both the tool and the supplied argument values	C10.2.5

Common pitfalls: relying on the service account's permissions instead of the caller's; letting model-generated output drive authorization; not re-checking authorization when context changes mid-session.

AD.3 Data Classification & Tenant Isolation

Keep data within its authorization and tenancy boundaries as it flows through AI-specific transformations and shared infrastructure.

Control / Technique	Requirement IDs
Sensitive data served through retrieval pipelines rather than persisted into model weights	C5.2.3
Classification labels propagated to downstream resources (embeddings, prompt caches, model outputs)	C5.2.7
Cross-tenant isolation in shared model serving (fine-tuning, inference, embedding operations)	C5.3.1

Control / Technique	Requirement IDs
Cross-tenant isolation across shared compute (hardware partitioning, confidential computing, or dedicated allocation)	C5.3.2

Common pitfalls: dropping classification labels when data is embedded or cached; assuming logical multi-tenancy is sufficient against side channels in shared inference caches.

AD.4 Encryption & Data Protection

Protect data and secrets at rest, in transit, and in the model's observable context.

Control / Technique	Requirement IDs
Integrity protection of training data while stored and transferred	C1.1.3
Redaction, anonymization, or encryption of sensitive information in labels before use in any labeling artifact	C1.2.3
Encryption of locally stored model weights and sensitive parameters using hardware-backed key stores or secure enclaves	C4.3.4
Encryption at rest of models packaged in mobile, IoT, or embedded apps, decrypted only inside a trusted runtime or secure enclave	C4.3.5
Secrets and credentials kept out of the model's observable context (context window, system prompts, tool-call parameters)	C9.5.4

Common pitfalls: encrypting the database but not model checkpoints or embeddings; leaving model weights extractable from an app package; exposing API keys inside tool-call parameters.

AD.5 Integrity, Signing & Provenance

Verify authenticity and detect tampering of models, artifacts, messages, tool definitions, and generated media.

Control / Technique	Requirement IDs
Integrity monitoring of training data against unauthorized modification or corruption	C1.1.4
Cryptographic integrity for labeling artifacts	C1.2.2
Cryptographic signing of all model artifacts (weights, configs, tokenizers, base models, fine-tunes, adapters, safety/policy models)	C3.1.2
Signature verification at deployment admission and on load	C3.1.3
Signed edge/mobile model packages with on-device signature or checksum validation before load	C4.3.2

Control / Technique	Requirement IDs
Cryptographic binding of agent-initiated actions to each step of the execution chain for non-repudiation	C9.4.2
Integrity protection of agent state persisted between invocations	C9.4.4
Signed MCP tool responses with a unique nonce and timestamp for replay defense	C10.4.6
Tool-definition snapshotting with re-approval required on any change before invocation	C10.4.8
Watermarking of AI-generated media to prove it was AI-generated	C7.4.4

Common pitfalls: using mutable tags instead of immutable digests; not re-verifying tool definitions between MCP invocations; missing replay protection on tool responses.

AD.6 Input Validation & Sanitization

Validate, normalize, and constrain all inputs (including tool, MCP, and retrieved content) before they reach the model or downstream systems.

Control / Technique	Requirement IDs
Input normalization applied before tokenization or embedding	C2.1.1
Encoding and representation-smuggling detection and mitigation (canonicalization, strict schema validation, policy-based rejection, or explicit marking)	C2.1.2
Untrusted-input screening by a prompt-injection detection ruleset or classifier, with blocking	C2.1.3
Input length controls that reject (not truncate) content exceeding the context window	C2.1.4
Allow-list character-set restriction on all inputs	C2.1.5
Instruction hierarchy enforcement (system and developer messages override user and untrusted input)	C2.1.6
Reserved special tokens encoded as literal characters and not injectable into context	C2.1.7
Many-shot jailbreaking pattern detection	C2.1.8
Adversarial-perturbation, steganography, and hidden-content checks on non-text inputs (image, video, audio)	C2.2.3
Cross-modal coordinated attack detection	C2.2.4
Schema validation of tool outputs	C9.3.2
Verification of external resources named in model output against an approved allow-list or registry before install or invocation	C9.3.7
MCP response schema validation before injection into model context	C10.4.1
Indirect-prompt-injection screening of MCP responses before injection into model context	C10.4.2

Control / Technique	Requirement IDs
Rejection of unrecognized or oversized MCP function-call parameters	C10.4.3
Strict MCP schema validation	C10.4.4
Maximum MCP payload size limits	C10.4.5
Anomaly detection on external or untrusted inputs before inference	C11.4.1
Gating actions on inputs flagged as anomalous	C11.4.2

Common pitfalls: validating only the text modality while ignoring image/audio channels; relying on regex alone without semantic detection; not validating tool and MCP outputs before they re-enter agent context.

AD.7 Inbound Content & Policy Screening

Screen prompts and training content against policy before they reach the model or the training pipeline.

Control / Technique	Requirement IDs
Inbound content classification (violence, self-harm, hate, sexual) against configurable thresholds, with rejection or sanitization before model context	C2.2.1
Evaluation of content classification for unsupported languages	C2.2.2
Detection and removal of disallowed content before training	C1.3.4

Common pitfalls: deploying classifiers tuned only for one language; screening prompts but not the training corpus.

AD.8 Output Handling & Safety

Constrain, filter, and validate model outputs before they reach users or downstream systems.

Control / Technique	Requirement IDs
Schema validation of model outputs with rejection on mismatch	C7.1.1
Length limits and termination controls on generated output	C7.1.2
Confidence or uncertainty estimation for generated answers	C7.2.1
Automatic blocking or fallback when confidence drops below a defined threshold	C7.2.2
Additional verification step for responses classified as high-risk by policy	C7.2.3
Automated classifiers that scan responses and block defined harmful-content categories	C7.3.1
Detection and blocking of responses that disclose system prompt content or backend data	C7.3.2

Control / Technique	Requirement IDs
Prevention of model-generated output triggering outbound requests	**C7.3.3**
Detection of hidden, encoded, or misleading output (homoglyphs, formatting, metadata, structured fields)	**C7.3.4**

Common pitfalls: enforcing stop sequences in batch mode but not on streaming output; leaking the system prompt through paraphrase; treating a confidence score as available when the provider does not expose one.

AD.9 Rate Limiting, Budgets & Resource Control

Bound consumption to prevent abuse, runaway execution, denial of service, and model extraction.

Control / Technique	Requirement IDs
Per-tool quotas and timeouts (CPU, memory, disk, egress, execution time)	**C9.1.1**
Per-execution budgets (maximum recursion depth, token use, monetary spend) enforced by the runtime	**C9.1.2**
Per-principal and global inference rate limits sized to the extraction threat model, not a generic API throttle	**C11.2.2**

Common pitfalls: rate-limiting per endpoint but not per agent session; ignoring tool fan-out when sizing budgets; treating extraction defense as ordinary throttling.

AD.10 Sandboxing & Workload Isolation

Isolate models, tools, agents, and hardware workloads to contain failures and prevent lateral movement.

Control / Technique	Requirement IDs
Execution of AI models in isolated sandboxes	**C4.1.1**
Allow-list of serialization formats that do not permit code execution during deserialization	**C4.1.2**
Workload attestation before model loading	**C4.1.3**
Confidential inference protecting model weights at runtime through isolated execution	**C4.1.4**
Trusted execution environment with hardware-enforced isolation, memory encryption, and integrity protection	**C4.2.2**
GPU integrity validation via hardware attestation before each workload	**C4.2.3**
GPU memory partitioning with sanitization between jobs	**C4.2.4**
Version-pinned, signed, boot-attested accelerator firmware	**C4.2.1**

Control / Technique	Requirement IDs
Process, memory, and file-access isolation in edge inference runtimes	C4.3.3
Least-privilege sandbox or isolation for each tool or plugin	C9.3.1
Tool manifests declaring required privileges, resource limits, and output-validation requirements	C9.3.3
Runtime enforcement of declared tool-manifest privileges and limits	C9.3.4
Isolation of untrusted-data processing from tool-calling capability	C9.3.5
Architectural separation of untrusted tool-output processing from agent operations	C9.3.6
Least-privilege sandbox for locally launched MCP servers (restricted file system, network, system access)	C10.1.3
AI-specific runtime components not shared across environment boundaries (development, staging, production)	C3.4.1
Training and fine-tuning environments isolated from production	C3.4.2

Common pitfalls: sharing infrastructure between dev and prod; granting tool sandboxes more capability than needed; allowing untrusted data processing to reach tool-calling paths.

AD.11 Network & Egress Control

Control network boundaries, transport security, and traffic flow for AI workloads and MCP integrations.

Control / Technique	Requirement IDs
Authenticated, encrypted streamable HTTP for remote MCP transport	C10.3.1
stdio MCP transport restricted to controlled local environments	C10.3.2
Independent Origin and Host header validation on HTTP-based transports (DNS rebinding defense)	C10.3.3
MCP client minimum protocol-version enforcement (downgrade defense)	C10.3.4
Accelerator interconnects restricted to approved topologies and authenticated endpoints	C4.2.5

Common pitfalls: exposing stdio or SSE transports beyond the local host; skipping Origin/Host validation and enabling DNS rebinding; accepting downgraded protocol versions.

AD.12 Supply Chain & Artifact Integrity

Verify origin and authenticity of models, datasets, frameworks, and MCP components, and maintain an AI bill of materials.

Control / Technique	Requirement IDs
Model registry inventory of all deployed model artifacts and their origin	C3.1.1
Malicious-code scanning of models before import	C6.1.1
Approved-source-only download of model weights, datasets, and fine-tuning adapters	C6.1.2
Integrity verification of every third-party model artifact	C6.1.3
Behavioral acceptance test suite passed before promotion beyond development	C6.1.4
Version-controlled, machine-readable AI BOM per model artifact (datasets, weights, licenses, data-origin statements)	C6.2.1
Cryptographic signing of AI BOMs before deployment	C6.2.2
Build-failing AI BOM completeness checks when component metadata is missing	C6.2.3
MCP components obtained only from trusted sources and cryptographically verified	C10.1.1
Allow-listed MCP servers only	C10.1.2

Common pitfalls: treating AI BOMs as static documents rather than signed, version-controlled artifacts; not scanning pretrained weights for backdoors; pulling models from unapproved registries.

AD.13 Model Lifecycle, Deployment & Rollback

Manage model validation, deployment, rollback, and fine-tuning pipeline integrity.

Control / Technique	Requirement IDs
Pre-deployment automated input-validation, safety-evaluation, and output-sanitization testing	C3.2.1
Re-evaluation of models subjected to post-training quantization against the same safety and alignment test suite before deployment	C3.2.2
Security re-evaluation triggered by provider model, version, or routing changes	C3.2.3
Rollout mechanisms with automated rollback triggers	C3.3.1
Complete model-state restoration on rollback	C3.3.2
Isolated runtime state for model versions running in parallel	C3.3.3
Versioned, integrity-verified RLHF reward models before a training run	C3.5.1
Detection of reward hacking or reward-model over-optimization in RLHF stages	C3.5.2
Stage-by-stage integrity verification in multi-stage fine-tuning pipelines	C3.5.3
Fine-tuning checkpoints registered as distinct artifacts	C3.5.4

Common pitfalls: not testing rollback before it is needed; leaving retired model artifacts in serving caches; treating reward models as static infrastructure rather than versioned, validated artifacts.

AD.14 Training Data Integrity & Governance

Source, vet, and document training data so tampering, poisoning, and corruption can be detected and traced.

Control / Technique	Requirement IDs
Data minimization to only the features, attributes, and fields required for the stated purpose	C1.1.1
Up-to-date inventory of every training-data source (origin, responsible party, license, collection method, use constraints, processing history)	C1.1.2
Dataset watermarking for usage attribution and detection of unauthorized use	C1.1.5
Labeling-platform access controls restricting who can create, modify, or approve annotations	C1.2.1
Poisoning detection in training and fine-tuning pipelines	C1.3.1
Confidence thresholds and consistency checks on automatically generated labels	C1.3.2
Bias evaluation for models used in security-relevant decisions	C1.3.3
Defenses against clean-label poisoning attacks	C1.3.5
Dataset lineage recording (transformations, augmentations, merges)	C12.5.1
Logging of all labeling activities	C12.5.2
Write-time tagging of every ingested document (source, writer identity, timestamp)	C12.5.4

Common pitfalls: not scanning fine-tuning datasets for poisoning; collecting more attributes than the purpose requires; losing dataset lineage across transformations and merges.

AD.15 Memory, Embeddings & RAG Security

Harden vector stores, memory pipelines, and retrieval-augmented generation against leakage, poisoning, and fabricated provenance.

Control / Technique	Requirement IDs
Per-tenant uniqueness of vector identifiers and namespaces, preventing cross-tenant collisions	C8.1.1
Immutability of document metadata tags after the initial write	C8.1.2
Scope constraints enforced on retrieval operations	C8.1.3
Detection and masking, tokenization, or dropping of sensitive fields before embedding	C8.2.1
Detection, rejection, or quarantine of retrieval-manipulation content before vectorization	C8.2.4
Flagging and quarantine of outlier vectors before they enter production indices	C8.2.2
Source validation before agent or tool outputs are written to trusted memory	C8.2.3

Control / Technique	Requirement IDs
Contradiction checks on new memory writes, with conflicts triggering alerts	C8.2.5
Exclusion of expired vectors from retrieval results	C8.3.1
Memory reset capability	C8.3.2
Retention of quarantined content while excluding it from all retrieval results	C8.3.3
Attribution of RAG responses to their source documents	C7.4.1
RAG attributions derived from retrieval metadata, not generated by the model	C7.4.2
Traceability of RAG claims to the retrieved chunk	C7.4.3

Common pitfalls: auto-writing tool output into trusted memory without validation; serving expired or quarantined vectors; letting the model fabricate citations instead of deriving them from retrieval metadata.

AD.16 Adversarial Robustness & Privacy Defense

Test for and defend against evasion, membership inference, model inversion, extraction, and poisoning of the improvement loop.

Control / Technique	Requirement IDs
Alignment and safety training or fine-tuning to suppress disallowed content categories	C11.1.1
Version-controlled alignment test suite run on every model update or release	C11.1.2
Evaluation against known adversarial attack techniques relevant to the modality	C11.1.3
Hardening of models against adversarial inputs	C11.1.4
Automated evaluator that measures harmful-content rate and flags regressions beyond a threshold	C11.1.5
Suppression of directly returned model-inferred sensitive attributes	C11.2.1
Output calibration to reduce overconfident predictions exploitable by inference attacks	C11.2.3
Differentially-private optimization for training on sensitive datasets	C11.2.4
Membership-inference attack simulation demonstrating accuracy no better than random guessing	C11.2.5
Raw model outputs not exposed beyond the backend, with externally visible responses calibrated to extraction risk	C11.3.2
Model watermarking or fingerprinting so unauthorized copies can be identified	C11.3.3
Poisoning detection and human review gates protecting the safety-violation feedback pipeline	C11.4.3

Common pitfalls: testing only known jailbreak patterns without adaptive attacks; not re-running the alignment suite after model updates; exposing raw confidence vectors that accelerate extraction.

AD.17 Logging & Audit

Capture security-relevant events with sufficient context and integrity for forensic reconstruction and accountability.

Control / Technique	Requirement IDs
AI interaction logging with session context and AI-specific telemetry	C12.1.1
Logging of safety filtering and policy decisions in enough detail to audit content moderation	C12.1.2
Structured, interoperable log schema for inference events (model identifier, token usage, provider, operation type)	C12.1.3
Logging of RAG pipeline retrieval events (query, documents retrieved, knowledge source)	C12.1.4
Audit logs capturing the approval chain for security-critical proactive actions (approver identity, timestamp, parameters, outcome)	C12.4.2
Logging of kill-switch activations and override commands	C12.4.3
Immutable audit records for all model changes	C12.5.3

Common pitfalls: logging prompts without redaction; using mutable log storage without integrity protection; logging agent actions and approvals but not human-initiated overrides such as kill-switch activations.

AD.18 Monitoring, Detection & Incident Response

Detect AI-specific abuse, drift, and anomalies, and respond to incidents.

Control / Technique	Requirement IDs
Automated tool containment triggered by policy violations	C9.3.8
Extraction-attempt detector fed by query-pattern analysis	C11.3.1
Response measures triggered on detection of suspected model extraction	C11.3.4
Signature-based detection and alerting on jailbreak patterns, prompt injection, and adversarial inputs	C12.2.1
Behavioral anomaly detection (unusual conversation patterns, excessive retries, systematic probing)	C12.2.2
Custom detection rules for AI-specific threat patterns (coordinated jailbreak attempts, prompt injection, system prompt extraction)	C12.2.3
Extraction-alert events including offending query metadata	C12.2.4

Control / Technique	Requirement IDs
Granular token-usage attribution (per user, session, feature endpoint, team or workspace)	C12.2.5
Monitoring of LLM API traffic for covert-channel and command-and-control indicators	C12.2.6
Data drift detection using methods matched to the input type (KS test or PSI for tabular, embedding-distance for text/image)	C12.3.1
Hallucination detection monitoring of model outputs	C12.3.2
Hallucination rates tracked as continuous time-series metrics	C12.3.3
Distinction of unexplained behavioral shifts from gradual operational drift	C12.3.4
Security evaluation and threat-landscape assessment for autonomous action triggers	C12.4.1

Common pitfalls: not correlating AI-specific events with broader SIEM alerts; treating drift as a scheduled check rather than continuous monitoring; lacking AI-specific forensic tooling during an incident.

AD.19 Human Oversight & Shutdown Control

Require human approval for high-impact actions and provide reliable, exercised shutdown and graceful-degradation paths under human control.

Control / Technique	Requirement IDs
Swarm-level kill-switch that halts all active agent instances	C9.1.3
Runtime blocking of privileged, high-impact, or irreversible actions until explicit human approval is received and verified	C9.2.1
Approval requests displaying canonicalized, complete action parameters (diffs, commands, recipients, amounts, resources, scopes) without truncation	C9.2.2
Trusted reversibility classification for each high-impact action (read-only, reversible, externally reversible, irreversible)	C9.2.3
Runtime enforcement of reversibility classifications (block, require approval, or restrict)	C9.2.4
Restriction and bounding of any self-modification capability (prompt rewriting, tool-list changes, parameter updates)	C9.2.5
AI-augmented review of planned high-risk actions, adding to (not replacing) the deterministic policy gate	C9.2.6
Protection of the AI-augmented review mechanism against prompt-injection bypass	C9.2.7
Approvals cryptographically bound to parameters, requester identity, execution context, and a single-use nonce	C9.2.8
Isolation of approval-issuing key material or credentials from the agent runtime	C9.2.9
Multi-step or multi-agent chains enforcing the highest-impact reversibility classification in the chain	C9.2.10

Control / Technique	Requirement IDs
Manual kill-switch to immediately halt model inference and outputs	**C9.6.1**
Fail-closed blocking of a pending action when a human-approval gate is not satisfied within the defined time	**C9.6.2**
Kill-switch commands delivered through an out-of-band channel isolated from the agent runtime	**C9.6.3**
Explicit consent dialogue and cancellation option on installation of a local MCP server	**C10.4.7**

Common pitfalls: documenting a high-risk action policy never wired to a runtime gate; binding approval to parameters without binding to identity or context; defaulting to fail-open when the approver does not respond; assuming an in-band kill-switch will work against a compromised agent; implementing a kill-switch that is never exercised.

References

- NIST AI Risk Management Framework 1.0
- ISO/IEC 42001:2023: AI Management Systems Requirements
- OWASP Top 10 for Large Language Model Applications
- OWASP Application Security Verification Standard (ASVS)
- NIST SP 800-218A: Secure Software Development Practices for Generative AI

Appendix C: AI-Assisted Secure Coding

Objective

This appendix lists organizational controls for using AI coding tools safely. The range is baseline to advanced. Scope is coding, code review, and the rest of the SSDLC.

AC.1 AI-Assisted Secure-Coding Workflow

AI tooling has to slot into the existing SSDLC without weakening any of the security gates already in place. Equally important: write down the adversarial-AI threat scenarios that justify each guardrail. Doing this up front is much easier than reconstructing it after the fact.

#	Description	Level
AC.1.1	**Verify that** a written workflow says when AI tools may generate, refactor, or review code. The workflow names the approved tools, the prohibited use cases, and the data classifications that are allowed as input.	1
AC.1.2	**Verify that** the workflow covers every SSDLC phase from design and implementation through code review, testing, deployment, and post-deployment monitoring, and names the security gates that stay mandatory whether AI was involved or not.	2
AC.1.3	**Verify that** the workflow names the adversarial-AI threat scenarios it is built to mitigate. The list should cover prompt injection delivered through PR content, AI-generated supply-chain payloads, autonomous agents approving their own work, fork-PR secret exfiltration, and compromise of the model supply chain.	2
AC.1.4	**Verify that** metrics are collected on AI-produced and AI-mediated code, and that the results are compared against a human-only baseline. Vulnerability density, mean-time-to-detect, AI-attributable defect rate, prompt-injection detection rate, and fork-PR rejection rate are all useful.	3

Mappings & References:

- **AC.1.1:** NIST SSDF PO.1 (Define Security Requirements for Software Development); ISO/IEC 42001 Clauses 6, 8; OWASP SAMM Strategy & Metrics (SM), Policy & Compliance (PC).
- **AC.1.2:** NIST SSDF PW.1, PW.7; OWASP SAMM Education & Guidance (EG); ISO/IEC 5338 Clause 6.
- **AC.1.3:** MITRE ATLAS (Reconnaissance & Initial Access tactics); NIST AI 600-1 GOVERN; OWASP LLM Top 10 (2025) LLM03; OWASP Agentic Top 10 (2026) ASI04.
- **AC.1.4:** NIST AI RMF MEASURE; ISO/IEC 42001 Clause 9; OWASP SAMM Strategy & Metrics (SM).

AC.2 AI Tool Qualification & Threat Modeling

Do not adopt an AI coding tool until it has been evaluated. Three areas in particular: its security capabilities, its resistance to adversarial input, and the risk inherited from its supply chain.

#	Description	Level
AC.2.1	**Verify that** every AI tool, whether it is an assistant, a reviewer, an agent, or an MCP server, has a threat model. The threat model covers misuse, model inversion, training-data leakage, prompt injection from untrusted input, insecure output handling, excessive agency, and risk inherited from its dependency chain.	1
AC.2.2	**Verify that** the evaluation of each tool covers the local components (static and dynamic analysis), the SaaS endpoints (TLS, AuthN/AuthZ, logging, data residency), and the vendor's model supply chain (training-data provenance, fine-tune history, RAG sources). Each of these is reviewed and the review is written down.	2
AC.2.3	**Verify that** each tool goes through adversarial robustness testing before onboarding. The testing is repeated after any material change to the model or to the system prompts. Coverage includes automated prompt-injection probes, jailbreak suites, and indirect-injection corpora delivered through realistic PR and issue surfaces.	2
AC.2.4	**Verify that** evaluations follow a recognized framework such as NIST AI RMF, NIST AI 600-1 Generative AI Profile, or ISO/IEC 42001. Evaluations are repeated after a major version change, a vendor incident, or new threat intelligence relevant to the tool class.	3

Mappings & References:

- **AC.2.1:** OWASP LLM Top 10 (2025) LLM01, LLM06; OWASP Agentic Top 10 (2026) ASI01, ASI02, ASI03; AISVS C9; MITRE ATLAS (Threat modeling).
- **AC.2.2:** OWASP LLM Top 10 (2025) LLM03; OWASP Agentic Top 10 (2026) ASI04; NIST SSDF PO.1, PO.5; ISO/IEC 42001 Clause 8.
- **AC.2.3:** MITRE ATLAS (Adversarial ML testing); AISVS C2.1, C11.1; NIST AI 600-1 MEASURE.
- **AC.2.4:** ISO/IEC 42001 Clause 9.2; NIST AI RMF GOVERN.

AC.3 Secure Prompt & Context Management

Two goals in this family. First: stop secrets, proprietary code, and personal data from leaking into prompts. Second: treat any content sourced from the repository, a PR, or a third party as untrusted input. Any of it can carry a prompt-injection payload, and most of it usually does not, which is part of what makes the rare hostile case easy to miss.

> **Relationship to AISVS C2.1:** AC.3.3, AC.3.4, and AC.3.5 apply AISVS C2.1 (Prompt Injection Defenses) to the secure-coding case. If a finding here is something that C2.1 verification did not already close, count it as an additional gap (specific to coding-tool prompt construction). If C2.1 already closed it, do not count it twice.

#	Description	Level
AC.3.1	**Verify that** written guidance forbids putting secrets, credentials, PII, or classified data in any prompt sent to an AI tool. The guidance is enforced in pre-commit hooks, IDE integrations, and CI.	1
AC.3.2	**Verify that** technical controls automatically strip sensitive material from any context window sent to an AI tool. Client-side redaction, approved context filters, and secret scanners with pre-prompt hooks all qualify.	1

#	Description	Level
AC.3.3	**Verify that** any externally sourced context being fed to an AI tool is treated as untrusted and screened for prompt injection before it reaches the prompt. Sources to cover: PR descriptions and comments, fork-supplied diffs, issue bodies, commit messages, third-party documentation, web search results, and MCP tool outputs.	1
AC.3.4	**Verify that** the AI tool enforces an instruction hierarchy, with system and developer messages taking precedence over untrusted repository content. This hierarchy has to hold across multi-turn conversations and tool-augmented workflows.	1
AC.3.5	**Verify that** input length controls stop untrusted PR or repository content from crowding system instructions or safety directives out of the effective context window. Oversized inputs are rejected outright. Silent truncation is not acceptable.	2
AC.3.6	**Verify that** prompts and AI responses are encrypted in transit and at rest, and retained per the data-classification policy. Tenants and projects are cryptographically separated from each other.	3

Mappings & References:

- **AC.3.1:** OWASP LLM Top 10 (2025) LLM02 (Sensitive Information Disclosure); OWASP ASVS v5 V14 (Data Protection); ISO/IEC 27001:2022 A.8.12 (Data Leakage Prevention).
- **AC.3.2:** AISVS C2.2; OWASP LLM Top 10 (2025) LLM02; NIST SSDF PW.3.
- **AC.3.3:** AISVS C2.1; OWASP LLM Top 10 (2025) LLM01; OWASP Agentic Top 10 (2026) ASI06; MITRE ATLAS (Indirect prompt injection).
- **AC.3.4:** AISVS C2.1.2; OWASP LLM Top 10 (2025) LLM01; CISA Secure by Design.
- **AC.3.5:** OWASP LLM Top 10 (2025) LLM10; AISVS C2.1.4.
- **AC.3.6:** OWASP ASVS v5 V6 (Cryptography), V14 (Data Protection); ISO/IEC 27001:2022 A.8.24 (Use of Cryptography).

AC.4 Validation of AI-Generated Code

Catch the vulnerabilities AI output introduces. Fix them before the code reaches a merge or a deployment.

#	Description	Level
AC.4.1	**Verify that** AI-generated code always goes through code review by a qualified human engineer. The reviewer must not be the same identity that asked for the AI generation in the first place (separation of duties). And the AI agent itself does not count as the human reviewer.	1
AC.4.2	**Verify that** automated security testing runs on every pull request containing AI-generated code: SAST, IAST, DAST, secret scanning, IaC scanning, and SCA. Where the scanner supports them, AI-attribution-aware rules are turned on.	2
AC.4.3	**Verify that** pull requests containing AI-generated code are blocked from merging when an automated scan surfaces a critical security finding, defined as CVSS >= 9.0 or the equivalent threshold in the organization's vulnerability severity policy. Bypassing the block requires a written exception approved by an authorized human.	2

#	Description	Level
AC.4.4	**Verify that** security-critical files require an elevated review threshold when AI generated or modified them: two-person review, security-team sign-off, or stricter. Security-critical files here include authentication, authorization, and cryptography code; IAM policy; CI/CD workflow definitions; deployment manifests; and sandbox or network policy artifacts.	2
AC.4.5	**Verify that** differential fuzz testing or property-based tests cover the security-critical behaviors of AI-generated code: input validation, authorization logic, and deserialization safety.	3

Mappings & References:

- **AC.4.1:** NIST SSDF PW.7; OWASP ASVS v5 V10 (Coding Quality); ISO/IEC 27001:2022 A.5.3 (Segregation of Duties).
- **AC.4.2:** NIST SP 800-204D (Pipeline scanning controls); SLSA v1.2 Build Track L2; OWASP SAMM Security Testing (ST).
- **AC.4.3:** OWASP CI/CD Top 10 CICD-SEC-04 (Poisoned Pipeline Execution); NIST SSDF PW.7, PW.8.
- **AC.4.4:** NIST SSDF PW.4, PW.7; OWASP CI/CD Top 10 CICD-SEC-01 (Insufficient Flow Control); ISO/IEC 27001:2022 A.8.32 (Change Management).
- **AC.4.5:** NIST SSDF PW.8; OWASP ASVS v5 V11 (Business Logic).

AC.5 Explainability & Traceability of Code Suggestions

Auditors, defenders, and the developers themselves need to be able to see why a given AI suggestion was made, and how it ended up in a shipped artifact.

#	Description	Level
AC.5.1	**Verify that** prompt-and-response pairs are logged with stable correlation identifiers, so that an investigator can later replay the whole chain: prompt → response → commit → build → deployment.	1
AC.5.2	**Verify that** developers can pull up the citations (training snippets, retrieved documents, MCP tool outputs) that support a suggestion, and that the citation chain travels with the artifact.	3
AC.5.3	**Verify that** explainability reports, AI-event logs, and citation records are kept in tamper-evident storage (append-only, WORM, or an immutable log store) and are referenced during security reviews.	3

Mappings & References:

- **AC.5.1:** ISO/IEC 42001 Clause 7.5 (Documented Information); OWASP ASVS v5 V8 (Logging); NIST SP 800-218A (Generative AI logging guidance).
- **AC.5.2:** NIST AI RMF MEASURE; OWASP LLM Top 10 (2025) LLM03.
- **AC.5.3:** ISO/IEC 27001:2022 A.8.15 (Logging); NIST AI 600-1 MEASURE; ISO/IEC 42001 (traceability).

AC.6 Continuous Feedback, Adversarial Testing & Model Fine-Tuning

Improve model security over time. Watch for negative drift. Keep red-teaming the AI tooling. The red-team scope in this family is the AI tooling itself; the underlying systems and services the tooling depends on are handled by separate programs.

#	Description	Level
AC.6.1	**Verify that** developers and reviewers can flag insecure or non-compliant suggestions, and that each flag is tracked to closure with links back to the originating prompt and response and forward to any downstream artifacts.	1
AC.6.2	**Verify that** aggregated feedback feeds into periodic system-prompt updates or retrieval-augmented generation against vetted secure-coding corpora (OWASP Cheat Sheets, internal coding standards). Where the organization controls model training infrastructure, fine-tuning on the same feedback corpus is also required.	2
AC.6.3	**Verify that** scheduled red-team exercises target the AI tooling itself. The exercises include direct and indirect prompt-injection probes delivered through realistic PR, issue, and comment surfaces, jailbreak corpora, and supply-chain payload generation. Findings are remediated under tracked severity SLAs.	2
AC.6.4	**Verify that** a closed-loop evaluation harness runs regression tests after every fine-tune, system-prompt change, or model upgrade. Security metrics must meet or exceed the prior baseline before deployment.	3

Mappings & References:

- **AC.6.1:** NIST AI RMF MANAGE; ISO/IEC 42001 Clause 10; OWASP SAMM Defect Management (DM).
- **AC.6.2:** OWASP LLM Top 10 (2025) LLM03; NIST SSDF PO.3.
- **AC.6.3:** MITRE ATLAS (Adversarial ML lifecycle); NIST AI 600-1 MEASURE 2.7; OWASP SAMM Security Testing (ST).
- **AC.6.4:** ISO/IEC 42001 Clause 9.1; NIST AI RMF MEASURE.

AC.7 AI-Generated Infrastructure & Pipeline Artifacts

Infrastructure code, CI/CD workflow files, deployment manifests, and security policy artifacts each have outsized impact when they are wrong. When AI has generated them, the validation needs to be correspondingly stricter than for ordinary application code.

#	Description	Level
AC.7.1	**Verify that** AI-generated or AI-modified artifacts are clearly labeled and tracked as such. Artifact classes in scope include infrastructure-as-code (Terraform, CloudFormation, Pulumi, Bicep), CI/CD workflow files (GitHub Actions, GitLab CI, Jenkinsfile, Argo Workflows, Tekton), container and orchestration manifests (Dockerfile, Kubernetes, Helm), and security policy artifacts (IAM, OPA/Rego, NetworkPolicy, admission controllers).	1
AC.7.2	**Verify that** AI-generated infrastructure and pipeline configurations require human review and approval before they run in any environment beyond a hermetic sandbox.	2

#	Description	Level
AC.7.3	**Verify that** AI-generated infrastructure and workflow changes pass policy-as-code enforcement (OPA, Conftest, Checkov, tfsec, KICS, kube-linter) at the same level as, or stricter than, human-authored changes. Policy violations block promotion.	2
AC.7.4	**Verify that** changes to high-impact pipeline trigger configurations require both dual control and a security-team review, no matter who or what produced the change. The configurations in scope include GitHub Actions `pull_request_target` and `workflow_run`, self-hosted runner labels, workflow `permissions:` blocks, OIDC trust policies, and secret-environment mappings.	2
AC.7.5	**Verify that** drift detection compares deployed infrastructure and live workflow configurations against signed, AI-attributed baselines, and alerts on any unauthorized modification.	3

Mappings & References:

- **AC.7.1:** OWASP CI/CD Top 10 CICD-SEC-05 (Insufficient PBAC); SLSA v1.2 Build Track provenance; NIST SSDF PW.1.
- **AC.7.2:** NIST SP 800-204D (Approval gating); OWASP CI/CD Top 10 CICD-SEC-01; ISO/IEC 27001:2022 A.8.32 (Change Management).
- **AC.7.3:** OWASP ASVS v5 V10 (CI/CD Deployment Security); OWASP CI/CD Top 10 CICD-SEC-07 (Insecure System Configuration); NIST SSDF PW.4.
- **AC.7.4:** OWASP CI/CD Top 10 CICD-SEC-01, CICD-SEC-02; GitHub Security Lab "Preventing pwn requests" series; NIST SP 800-204D (Pipeline governance).
- **AC.7.5:** NIST SP 800-204D (Continuous monitoring); ISO/IEC 27001:2022 A.8.19.

AC.8 Autonomous Agent Change Control Constraints

Autonomous AI agents that generate code or configuration get the same separation-of-duties treatment that humans do. They cannot approve, merge, or promote their own work. This applies at the policy layer and at the technical layer.

#	Description	Level
AC.8.1	**Verify that** autonomous agents cannot approve, merge, sign, or deploy artifacts that they themselves generated, and that this constraint is enforced by the source-control system, the CI system, and the artifact registry. Policy alone does not satisfy this control.	1
AC.8.2	**Verify that** AI systems run with scoped, non-human identities (service accounts, workload identities, OIDC-issued ephemeral tokens), and that those identities cannot be used to promote their own generated artifacts across environments.	2
AC.8.3	**Verify that** autonomous agents cannot bypass branch protection, required reviews, required status checks, signed-commit requirements, or merge queues. Any attempt by an agent to change these settings raises a security alert.	2
AC.8.4	**Verify that** separation of duties holds across the stages of an AI-generated change. Each stage (generation, review, approval, deployment) is performed by a distinct principal, whether human or system.	3

Mappings & References:

- **AC.8.1:** OWASP Agentic Top 10 (2026) ASI03 (Identity and Privilege Abuse), ASI10 (Rogue Agents); OWASP ASVS v5 V10; NIST SP 800-53r5 AC-5 (Separation of Duties).
- **AC.8.2:** NIST SP 800-207 (Zero Trust Architecture); OWASP CI/CD Top 10 CICD-SEC-02; ISO/IEC 27001:2022 A.5.15 (Access Control).
- **AC.8.3:** OWASP CI/CD Top 10 CICD-SEC-01; GitHub Docs (Branch protection rules and rulesets); OWASP Agentic Top 10 (2026) ASI03.
- **AC.8.4:** NIST SSDF PO.2; ISO/IEC 27001:2022 A.5.3; NIST SP 800-53r5 AC-5.

AC.9 AI Artifact Origin Validation for Deployment

Deployment and promotion pipelines need to validate the cryptographic origin and the generation history of AI-generated artifacts. They do this before letting the artifact through.

#	Description	Level
AC.9.1	**Verify that** AI-generated artifacts carry signed origin and generation metadata (in-toto or SLSA provenance attestations, AI BOM entries) identifying the AI system that produced them, the generation context, the humans involved, and the associated audit records.	2
AC.9.2	**Verify that** deployment pipelines check the presence, signature, and integrity of origin and generation metadata on AI-generated artifacts before promotion, using a trusted verifier (Sigstore/cosign, in-toto verification).	3
AC.9.3	**Verify that** artifacts are rejected at deployment and quarantined for review when they are missing required origin and generation information, signed by untrusted keys, or produced by an unapproved AI system or environment.	3

Mappings & References:

- **AC.9.1:** SLSA v1.2 (Provenance attestations); CycloneDX ML-BOM; in-toto Attestation Framework.
- **AC.9.2:** SLSA v1.2 (Verification Summary Attestations); Sigstore/cosign (Signature verification); OWASP SCVS.
- **AC.9.3:** SLSA v1.2 (Verifier requirements); NIST SP 800-204D (Promotion gating).

AC.10 Generation Audit Trail Completeness and Validation

AI-generated artifacts need complete and consistent origin and generation records, validated before integration or deployment. The reason matters. Policy-based enforcement of origin tracking only works if the recorded information is itself complete and consistent. When records are missing fields, or when the fields they do have contradict each other, detections get missed and enforcement opens gaps. So origin tracking is treated as a first-class requirement here, and validated before an artifact is accepted.

#	Description	Level
AC.10.1	**Verify that** AI-generated artifacts carry the required origin and generation fields: model identity and version, tool or agent identity, generation context, prompt hash, human involvement, session identifiers, and correlation IDs.	1
AC.10.2	**Verify that** origin and generation metadata is checked for completeness and consistency: no missing or ambiguous fields, values normalized to a single representation, and a signature chain that validates back to a trusted root.	2
AC.10.3	**Verify that** artifacts with incomplete, inconsistent, or unverifiable origin and generation metadata are rejected before merge or deployment, and that the rejection event is logged so trends can be tracked. Rejection happens on the verifier side, against the attestation or proof model defined in SLSA and the verification criteria in ISO/IEC 42001.	3

Mappings & References:

- **AC.10.1:** CycloneDX ML-BOM schema; NIST SP 800-218A (Generative AI provenance); ISO/IEC 42001 Clause 7.5.
- **AC.10.2:** OWASP SCVS (Provenance and Pedigree); SLSA v1.2 VSA verification.
- **AC.10.3:** SLSA v1.2 (Verifier-side enforcement); ISO/IEC 42001 Clause 9.

AC.11 AI Code-Review & Assistant Bot Hardening

AI code-review bots, PR-comment bots, MCP-driven assistants (Model Context Protocol), and IDE copilots are all reachable through untrusted repository content. The reachable surfaces include PR diffs, descriptions, comments, issues, and any workflow files supplied from a fork. This family covers the case where an attacker uses one of those surfaces to push a defender's own AI agent into approving, ignoring, or actively assisting a supply-chain attack.

> **Relationship to AISVS C2.1, C9.3, and C9.5:** AC.11.1 through AC.11.5 are applications of three AISVS chapter controls to the specific case of AI code-review and assistant bots operating over untrusted PR content. The three chapter controls are C2.1 (Prompt Injection Defenses), C9.3 (Component Isolation and Tool Authorization), and C9.5 (Agent Authorization, Delegation, and Continuous Enforcement). The appendix restates each one with bot-specific guidance. Counting rule is the same as elsewhere: a finding here is either an additional gap that the upstream chapter did not close, or it is already counted under the chapter. Not both.

#	Description	Level
AC.11.1	**Verify that** AI review and assistant bots treat every piece of PR-supplied content (diff, title, description, comments, file contents, commit messages, linked external URLs) as untrusted input, and apply the AISVS C2.1 prompt-injection defenses: instruction-hierarchy enforcement, content sanitization, and indirect-injection detection.	1
AC.11.2	**Verify that** AI review and assistant bot system prompts and policy configurations are integrity-checked at load time (signed, hash-pinned), and that nothing in the repository, in branch contents, in PR-sourced environment variables, or in any other user-controllable input can modify them.	1

#	Description	Level
AC.11.3	**Verify that** AI review and assistant bots emit only structured, schema-validated output (JSON with an allow-list of fields and actions). Any free-form output is treated as untrusted and never executed as a command, a query, a shell snippet, or a workflow step.	1
AC.11.4	**Verify that** AI review and assistant bots run in network-isolated, least-privilege sandboxes: a dedicated namespace, default-deny egress with an allow-list to approved APIs only, no mounted repository secrets, and ephemeral credentials only.	2
AC.11.5	**Verify that** any privileged action a bot can take (approving a PR, merging, labeling, dismissing reviews, posting comments outside its sandbox, invoking external tools) goes through a separate, audited authorization path. That path is adjudicated by a policy engine, not by the LLM.	2
AC.11.6	**Verify that** AI review and assistant bots log all prompts (including externally sourced context), tool calls, and outputs to tamper-evident storage. Egress patterns (URLs, IPs, DNS, payload sizes) are continuously monitored for exfiltration indicators, with alerting tuned for webhook, paste-site, and bin-service destinations.	2
AC.11.7	**Verify that** AI review bots run in a zero-privilege, read-only shadow mode for untrusted fork PRs. In shadow mode, inline code-generation commentary is restricted and privileged workflow interaction is forbidden, until a repository maintainer has cleared an initial first-time-contributor verification gate.	2
AC.11.8	**Verify that** AI review and assistant bots are subject to continuous adversarial testing: indirect-prompt-injection corpora are replayed against the bot through simulated PRs, issues, and comments. Detection effectiveness is tracked over time, and a regression blocks the model or prompt update that caused it.	3

Mappings & References:

- **AC.11.1:** AISVS C2.1; OWASP LLM Top 10 (2025) LLM01; OWASP Agentic Top 10 (2026) ASI01, ASI06.
- **AC.11.2:** AISVS C2.1; OWASP LLM Top 10 (2025) LLM01; OWASP Agentic Top 10 (2026) ASI01.
- **AC.11.3:** AISVS C7.1; OWASP LLM Top 10 (2025) LLM05; OWASP Agentic Top 10 (2026) ASI02, ASI05.
- **AC.11.4:** AISVS C9.3; OWASP Agentic Top 10 (2026) ASI02, ASI03, ASI05; NIST SP 800-204D (Workload isolation).
- **AC.11.5:** AISVS C9.5, C5.2.5; OWASP ASVS v5 V4 (Access Control); OWASP Agentic Top 10 (2026) ASI02, ASI03.
- **AC.11.6:** OWASP ASVS v5 V8 (Logging & Error Handling); OWASP LLM Top 10 (2025) LLM02; ISO/IEC 27001:2022 A.8.15, A.8.16.
- **AC.11.7:** GitHub Security Lab "Preventing pwn requests" series (Parts 1-4); OWASP Agentic Top 10 (2026) ASI01, ASI03, ASI09; OWASP CI/CD Top 10 CICD-SEC-01.
- **AC.11.8:** MITRE ATLAS (Indirect prompt injection); AISVS C2.1, C11.1; OWASP SAMM Security Testing (ST).

AC.12 CI/CD Pipeline Hardening Specific to AI Augmentation

Two kinds of CI/CD pipeline control are in scope for this family: those that AI augmentation *newly requires*, and those that AI augmentation *breaks*. Generic CI/CD hygiene is not in scope here; it is covered elsewhere. Short-lived credentials, immutable action pinning, branch protection, SLSA Build Track L3 provenance, and multi-party production approval are all addressed by OWASP ASVS v5 V10, the OWASP Top 10 CI/CD Security Risks (CICD-SEC-01 through CICD-SEC-10), NIST SP 800-204D, and SLSA v1.2. Adopters implement those baselines and verify them against the originating standards. We do not repeat that assessment here.

#	Description	Level
AC.12.1	**Verify that** workflows triggered by untrusted contributions (GitHub Actions `pull_request_target`, `workflow_run`, and equivalent fork-aware triggers in other CI systems) never check out, build, test, or otherwise execute untrusted code in a context that has repository write permissions or access to repository, organization, package-registry, cloud, or deployment secrets. Where a privileged follow-up step is needed, the untrusted contribution is first processed in an unprivileged `pull_request` workflow, and only validated passive artifacts are passed forward to a separate privileged workflow.	1
AC.12.2	**Verify that** secrets, credentials, and pipeline job tokens are not persisted into workspaces that process AI-touched or fork-originated untrusted code. For example, set `persist-credentials: false` on checkout where the platform supports it, and scrub CI runners of cached credentials before AI tooling runs.	1
AC.12.3	**Verify that** secrets are not exposed to workflows running code from forks or first-time contributors. Environment-protection rules (or the platform equivalent, such as protected variables and deployment approvals) require a manual approval before any secret-bearing job runs for those contributions. This control pairs with AC.11.7 and AC.13.2. Bot-level enforcement under AC.11.7 does not substitute for the platform-level enforcement required here.	1
AC.12.4	**Verify that** self-hosted or persistent runners used by AI tooling are ephemeral (destroyed after each job), network-segmented, and isolated from production credentials. Persistent or long-lived runners do not process fork PRs or AI-generated untrusted artifacts under any circumstances.	2
AC.12.5	**Verify that** changes to workflow definition files (`.github/workflows/*`, `.gitlab-ci.yml`, `Jenkinsfile`, Argo, Tekton, and equivalents) are detected on every PR and route through an elevated review path that includes a security reviewer, regardless of who the contributor is or whether AI was involved. AI agents must not be granted bypass authority over this review path.	2
AC.12.6	**Verify that** pipeline audit logs (workflow runs, secret access, runner registration, permission grants, OIDC token issuance) are streamed in real time to centralized security monitoring. Detection rules are tuned for AI-augmented threat patterns: bulk PR creation from new accounts, workflow-file modifications in fork PRs, unexpected secret access from AI-runner pools, and unusual egress (webhooks, paste sites, bin services) from AI workloads.	2
AC.12.7	**Verify that** artifacts produced by untrusted PR workflows are treated as untrusted passive data when a privileged follow-up workflow consumes them. The privileged workflow never executes binaries, scripts, packages, caches, or generated workflow fragments that originated in an untrusted contribution.	2
AC.12.8	**Verify that** the remediation of a vulnerable workflow includes invalidating or re-validating any PR that was opened before the fix landed. Without this step, a later commit to the same PR can pick up the stale workflow definition and route around the fix.	2

Mappings & References:

- **AC.12.1:** OWASP CI/CD Top 10 CICD-SEC-01, CICD-SEC-04; GitHub Security Lab "Preventing pwn requests" series; NIST SP 800-204D (Pipeline isolation).
- **AC.12.2:** OWASP CI/CD Top 10 CICD-SEC-02, CICD-SEC-06; GitHub Docs (Automatic token authentication and permissions); NIST SP 800-53r5 AC-6 (Least Privilege).
- **AC.12.3:** OWASP CI/CD Top 10 CICD-SEC-01; GitHub Docs (Approving workflow runs from public forks; Protected environments); GitLab Docs (Protected variables).
- **AC.12.4:** OWASP CI/CD Top 10 CICD-SEC-06; NIST SP 800-204D (Runner isolation); ISO/IEC 27001:2022 A.8.22 (Segregation of Networks).
- **AC.12.5:** OWASP CI/CD Top 10 CICD-SEC-01; NIST SSDF PW.7; ISO/IEC 27001:2022 A.8.32.
- **AC.12.6:** OWASP ASVS v5 V8 (Logging); OWASP CI/CD Top 10 CICD-SEC-10 (Insufficient Logging and Visibility); ISO/IEC 27001:2022 A.8.16.
- **AC.12.7:** GitHub Security Lab "Preventing pwn requests" series; OWASP CI/CD Top 10 CICD-SEC-01; NIST SP 800-204D (Cross-workflow trust boundaries).
- **AC.12.8:** GitHub Security Lab "Preventing pwn requests" Part 4 (Alvaro Munoz, 2025); OWASP CI/CD Top 10 CICD-SEC-01; NIST SSDF RV.1.

AC.13 Adversarial AI Detection in Inbound Contributions

The previous families were about defending your own AI from misuse. This one flips the lens. Here the AI is on the attacker's side, and you are trying to spot the signal in inbound contributions and content. The scenario worth defending against is the one where an attacker uses AI to run fork-and-PR campaigns at scale, with malicious payloads tailored to the target repository.

#	Description	Level
AC.13.1	**Verify that** contribution-velocity and contributor-reputation analytics flag anomalies: bulk PR creation from newly created accounts, coordinated fork waves immediately preceding PRs, PR volumes that are inconsistent with human authorship, and reuse of payload patterns across unrelated repositories.	1
AC.13.2	**Verify that** PRs from first-time or low-reputation contributors require maintainer approval before any privileged workflow processes them. Privileged workflows here include AI review bots, secret-bearing jobs, and external-integration calls.	1
AC.13.3	**Verify that** automated PR pipeline gates detect known indicators of LLM-generated or LLM-assisted malicious payload patterns: registry-confusable or typosquatted dependency names, package references that do not resolve to any published version, and dependencies whose creation, first-publication, or maintainer-change timestamps look anomalous relative to the PR.	2
AC.13.4	**Verify that** detection rules are tagged to MITRE ATT&CK (T1195 Supply Chain Compromise and CI/CD-relevant sub-techniques) and to MITRE ATLAS techniques, maintained for the inbound contribution analysis use case, and reviewed against current threat intelligence.	2
AC.13.5	**Verify that** confirmed or high-confidence adversarial contributions trigger automated containment: block the PR, quarantine the fork, suspend the contributor, notify maintainers, and freeze affected workflow files. Triage decisions feed back into detection tuning.	3

#	Description	Level
AC.13.6	**Verify that** PR analytics include structural AST profiling and stylometric or entropy-based heuristics tuned to identify LLM-generated code patterns. Detection in this category is still maturing, so compensating controls are accepted in place of high-precision automated detection: mandatory human review on flagged PRs, sandboxed execution of suspect payloads, and deferred merge until additional signals accrue.	3

Mappings & References:

- **AC.13.1:** OWASP CI/CD Top 10 CICD-SEC-01; NIST AI RMF MANAGE; MITRE ATLAS (Reconnaissance).
- **AC.13.2:** GitHub Docs (Approving workflow runs from public forks); OWASP CI/CD Top 10 CICD-SEC-01; NIST SSDF PW.4.
- **AC.13.3:** OWASP LLM Top 10 (2025) LLM03; OWASP CI/CD Top 10 CICD-SEC-03 (Dependency Chain Abuse); NIST SSDF PW.4.
- **AC.13.4:** MITRE ATT&CK T1195; MITRE ATLAS (Technique catalogue); OWASP SAMM Threat Assessment (TA).
- **AC.13.5:** NIST AI RMF MANAGE; ISO/IEC 27001:2022 A.5.25 (Assessment of Information Security Events); OWASP SAMM Incident Management (IM).
- **AC.13.6:** MITRE ATLAS (Adversarial ML output detection, research-edge); OWASP LLM Top 10 (2025) LLM03; NIST SSDF PW.8.

AC.14 Compromise Containment & Automated Remediation

Things go wrong eventually. When an AI-adjacent compromise (a prompt-injected bot, a leaked CI secret, a malicious AI-generated artifact in a build) is suspected or confirmed, the goal is to contain the damage and shorten the recovery.

#	Description	Level
AC.14.1	**Verify that** an incident-response playbook exists for AI-in-pipeline compromise. At minimum it covers: revoking AI-agent credentials, rotating every secret that touched the compromised workflow run, quarantining the compromised artifacts, notifying downstream consumers, notifying regulators where applicable, and preserving prompts, responses, and audit logs for forensics.	1
AC.14.2	**Verify that** any secret that touched a workflow run associated with a suspicious PR, a prompt-injection event, or an AI-agent anomaly is automatically rotated, and that downstream issuers (cloud IAM, package registries, signing-key custodians) are notified of the rotation.	1
AC.14.3	**Verify that** AI agent identities (keys, tokens, OIDC trust grants) can be rapidly revoked and quarantined, with a target time-to-revoke that is written down and tested at least once a year.	2
AC.14.4	**Verify that** build provenance and AI BOM records are used during incident response to identify every downstream artifact produced under the suspect AI agent or the compromised pipeline run, so that recall, rebuild, or quarantine can be targeted.	2
AC.14.5	**Verify that** automated remediation is tested in tabletop or live-fire exercises at least once a year. The scenarios include a prompt-injected reviewer bot, fork-PR secret exfiltration, and an AI-generated malicious workflow file.	3

Mappings & References:

- **AC.14.1:** ISO/IEC 27001:2022 A.5.24, A.5.26; NIST AI RMF MANAGE; OWASP SAMM Incident Management (IM).
- **AC.14.2:** OWASP ASVS v5 V6 (Cryptography), V14; OWASP CI/CD Top 10 CICD-SEC-06; NIST SSDF RV.2.
- **AC.14.3:** AISVS C9.4 (Agent and Orchestrator Identity); NIST SP 800-207 (Zero Trust Architecture); ISO/IEC 27001:2022 A.5.18 (Access Rights).
- **AC.14.4:** OWASP SCVS (Bill-of-materials analysis); CycloneDX ML-BOM tracing; NIST SSDF RV.1.
- **AC.14.5:** NIST SSDF RV.1; ISO/IEC 27001:2022 A.5.28 (Collection of Evidence); OWASP SAMM Incident Management (IM).