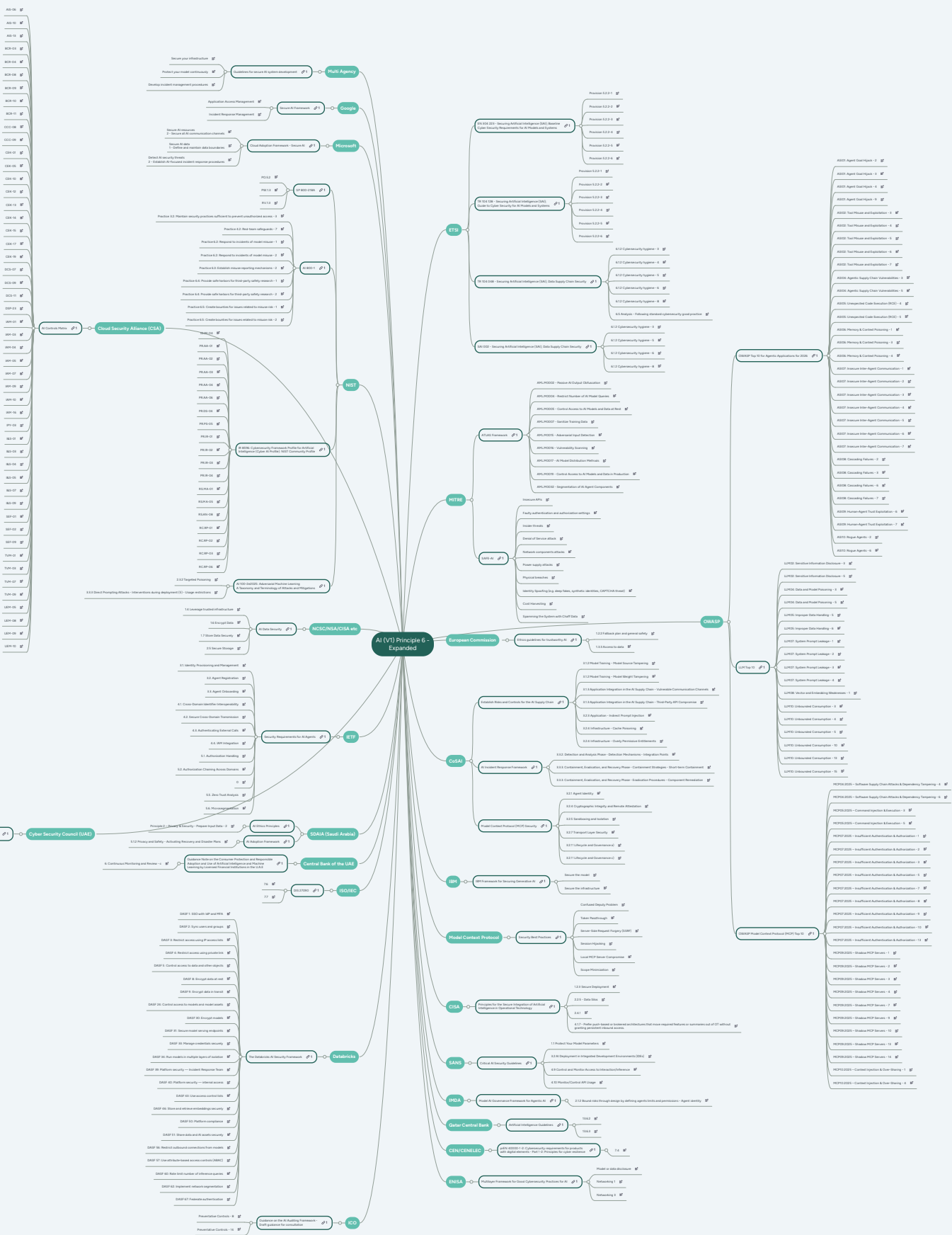




AI (V1) Principle 6 - Expanded

1. ETSI

1.1. EN 304 223 - Securing Artificial Intelligence (SAI); Baseline Cyber Security Requirements for AI Models and Systems

Link: https://www.etsi.org/deliver/etsi_en/304200_304299/304223/02.01.01_60/en_304223v020101p.pdf

Link: https://www.etsi.org/deliver/etsi_en/304200_304299/304223/02.01.01_60/en_304223v020101p.pdf

1.1.1. Provision 5.2.2-1

Developers and System Operators shall evaluate their organization's access control frameworks and identify appropriate measures to secure APIs, models, data, and training and processing pipelines.

1.1.2. Provision 5.2.2-2

If a Developer offers an API to external customers or collaborators, they shall apply controls that mitigate attacks on the AI system via the API. For example, placing limits on model access rate to limit an attacker's ability to reverse engineer or overwhelm defences to rapidly poison a model.

1.1.3. Provision 5.2.2-3

Developers shall also create dedicated environments for development and model tuning activities. The dedicated environments shall be backed by technical controls to ensure separation and principle of least privilege. In the context of AI, this is particularly necessary because training data shall only be present in the training and development environments where this training data is not based on publicly available data

1.1.4. Provision 5.2.2-4

Developers and System Operators shall implement and publish a clear and accessible vulnerability disclosure policy.

1.1.5. Provision 5.2.2-5

Developers and System Operators shall create, test and maintain an AI system incident management plan and an AI system recovery plan.

1.1.6. Provision 5.2.2-6

Developers and System Operators should ensure that, where they are using cloud service operators to help to deliver the capability, their contractual agreements support compliance with the above requirements.

1.2. TR 104 128 - Securing Artificial Intelligence (SAI); Guide to Cyber Security for AI Models and Systems

Link: https://www.etsi.org/deliver/etsi_tr/104100_104199/104128/01.01.01_60/tr_104128v010101p.pdf

Link: https://www.etsi.org/deliver/etsi_tr/104100_104199/104128/01.01.01_60/tr_104128v010101p.pdf

1.2.1. Provision 5.2.2-1

"Developers and System Operators shall evaluate their organization's access control frameworks and identify appropriate measures to secure APIs, models, data, and training and processing pipelines." (ETSI TS 104 223 [i.1])

Related threats/risks:

Inadequate access control can expose sensitive data, models, and pipelines to unauthorized access, increasing the risk of data leaks, model tampering, or unauthorized modifications.

Example Measures/Controls:

Establish Role-Based Access Controls (RBAC): Implement role-based access controls to limit access to AI models, data, and pipelines based on user roles and responsibilities, enforcing the principle of least privilege. This includes research environments. Protect API endpoints and data pipelines by implementing access controls, encryption, and authentication mechanisms to prevent unauthorized access.

1.2.2. Provision 5.2.2-2

"If a Developer offers an API to external customers or collaborators, they shall apply appropriate controls that mitigate attacks on the AI system via the API. For example, placing limits on model access rate to limit an attacker's ability to reverse engineer or overwhelm defences to rapidly poison a model." (ETSI TS 104 223 [i.1])

Related threats/risks:

Externally exposed APIs increase the risk of model extraction attacks, rapid data poisoning, and abuse, potentially compromising the integrity of the AI system.

Example Measures/Controls 1:

Implement API Rate Limiting: Enforce rate limits on API requests to prevent attackers from overwhelming the system, reverse engineering the model, or rapidly injecting malicious inputs.

Example Measures/Controls 2:

Use Behavioural Analysis for API Security: Implement behavioural analysis tools to detect abnormal API usage that could indicate malicious intent, such as model extraction or poisoning attempts.

Example Measures/Controls 3:

Deploy API Gateway with Security Features: Use an API gateway with security features like throttling, dynamic rate limiting, and any other attack detection features, authentication, and logging to manage and monitor access to external-facing APIs.

1.2.3. Provision 5.2.2-3

"Developers shall also create dedicated environments for development and model tuning activities. The dedicated environments shall be backed by technical controls to ensure separation and principle of least privilege. In the context of AI, this is particularly necessary because training data shall only be present in the training and development environments where this training data is not based on publicly available data." (ETSI TS 104 223 [i.1])

Related threats/risks:

Without separation and environment-specific controls, production-grade sensitive data and models can be exposed to unauthorized access via development or research workflows, leading to data leaks, tampering, unauthorized deployments, or compliance violations.

Example Measures/Controls:

Set Up Dedicated Development and Production Environments: Establish separate environments for development, testing, and production, ensuring data and models are only accessible where necessary. Restrict sensitive training data to the development environment, isolating it from production.

1.2.4. Provision 5.2.2-4

"Developers and System Operators shall implement and publish a clear and accessible vulnerability disclosure policy." (ETSI TS 104 223 [i.1])

Related threats/risks:

Without a defined vulnerability disclosure policy, security vulnerabilities can go unreported, exposing the organization to delayed or missed opportunities to patch critical issues.

Example Measures/Controls:

Develop and Publish a Vulnerability Disclosure Policy: Create a vulnerability disclosure policy that details how vulnerabilities can be reported, including timelines for acknowledgment and resolution.

1.2.5. Provision 5.2.2-5

"Developers and System Operators shall create, test, and maintain an AI system incident management plan and an AI system recovery plan." (ETSI TS 104 223 [i.1])

Related threats/risks:

Without a dedicated incident and recovery plan, AI systems can be slow to recover from disruptions, resulting in prolonged downtime or compromised model performance.

Example Measures/Controls:

Develop an AI-Specific Incident Management Plan: Create an incident management plan tailored to AI-specific threats, such as discovery of data poisoning, model drift, and adversarial attacks with recovery steps to a known good state, including procedures for validating model integrity.

1.2.6. Provision 5.2.2-6

"Developers and System Operators should ensure that, where they are using cloud service operators to help to deliver the capability, their contractual agreements support compliance with the above requirements." (ETSI TS 104 223 [i.1])

Related threats/risks:

Without clear understanding of cloud service agreements, organizations might not know what security measures they can expect or demand from the cloud provider, potentially leaving AI assets exposed to gaps in data protection or compliance failures.

Example Measures/Controls:

Define and Validate Security Clauses in Cloud Contracts: Ensure cloud service agreements explicitly outline security responsibilities, compliance standards, and support provisions, including data protection, access controls, incident response, and audit capabilities. Provide detailed documentation to stakeholders to bridge knowledge gaps about the cloud provider's obligations.

1.3. TR 104 048 - Securing Artificial Intelligence (SAI); Data Supply Chain Security

Link: https://www.etsi.org/deliver/etsi_tr/104000_104099/104048/01.01.01_60/tr_104048v010101p.pdf

Link: https://www.etsi.org/deliver/etsi_tr/104000_104099/104048/01.01.01_60/tr_104048v010101p.pdf

1.3.1. 6.1.2 Cybersecurity hygiene - 3

Any keys and passwords used to access data should be secured. Weak passwords and the reuse of compromised passwords are common enterprise security vulnerabilities and apply to both cloud and local storage. A robust password policy and multi-factor authentication should be in place.

1.3.2. 6.1.2 Cybersecurity hygiene - 4

Strong access controls should be in place, applying the principle of least privilege. These stand alongside limits to the number of queries allowed to be made against a model in a period of time.

1.3.3. 6.1.2 Cybersecurity hygiene - 5

Any organization using cloud storage should understand its responsibilities and the limits of what is provided by its CSP. This is particularly relevant where products move from development into critical operations, and may have inherited risk from the previous research phases.

1.3.4. 6.1.2 Cybersecurity hygiene - 6

A good Continuous Integration/Continuous Deployment (CI/CD) pipeline can improve the security of a resultant system, however, tools used in the pipeline should be updated regularly and access to repositories should be monitored.

1.3.5. 6.1.2 Cybersecurity hygiene - 8

A cyber incident response plan should be in place and audit processes should be established in order to support analysis of and learning from any security incidents that do take place.

1.3.6. 6.5 Analysis - Following standard cybersecurity good practice

Following standard cybersecurity good practice, including following the principle of least privilege when accessing data.

1.4. SAI 002 - Securing Artificial Intelligence (SAI); Data Supply Chain Security

Link: https://www.etsi.org/deliver/etsi_gr/SAI/001_099/002/01.01.01_60/gr_SAI002v010101p.pdf

Link: https://www.etsi.org/deliver/etsi_gr/SAI/001_099/002/01.01.01_60/gr_SAI002v010101p.pdf

1.4.1. 6.1.2 Cybersecurity hygiene - 3

Any keys and passwords used to access data should be secured. Weak passwords and the reuse of compromised passwords are common enterprise security vulnerabilities and apply to both cloud and local storage. A robust password policy and multi-factor authentication should be in place.

1.4.2. 6.1.2 Cybersecurity hygiene - 5

Any organization using cloud storage should understand its responsibilities and the limits of what is provided by its CSP. This is particularly relevant where products move from development into critical operations, and may have inherited risk from the previous research phases.

1.4.3. 6.1.2 Cybersecurity hygiene - 6

A good CI/CD (continuous integration/continuous deployment) pipeline can improve the security of a resultant system, however, tools used in the pipeline should be updated regularly and access to repositories should be monitored [i.11].

1.4.4. 6.1.2 Cybersecurity hygiene - 8

A cyber incident response plan should be in place and audit processes should be established in order to support analysis of and learning from any security incidents that do take place [i.31].

2. MITRE

2.1. ATLAS Framework

Link: <https://atlas.mitre.org/mitigations>

Link: <https://atlas.mitre.org/mitigations>

2.1.1. AML.M0002 - Passive AI Output Obfuscation

Decreasing the fidelity of model outputs provided to the end user can reduce an adversary's ability to extract information about the model and optimize attacks for the model.

2.1.2. AML.M0004 - Restrict Number of AI Model Queries

Limit the total number and rate of queries a user can perform.

2.1.3. AML.M0005 - Control Access to AI Models and Data at Rest

Establish access controls on internal model registries and limit internal access to production models. Limit access to training data only to approved users.

2.1.4. AML.M0007 - Sanitize Training Data

Detect and remove or remediate poisoned training data. Training data should be sanitized prior to model training and recurrently for an active learning model.

Implement a filter to limit ingested training data. Establish a content policy that would remove unwanted content such as certain explicit or offensive language from being used.

2.1.5. AML.M0015 - Adversarial Input Detection

Detect and block adversarial inputs or atypical queries that deviate from known benign behavior, exhibit behavior patterns observed in previous attacks or that come from potentially malicious IPs. Incorporate adversarial detection algorithms into the AI system prior to the AI model.

2.1.6. AML.M0016 - Vulnerability Scanning

Vulnerability scanning is used to find potentially exploitable software vulnerabilities to remediate them.

File formats such as pickle files that are commonly used to store AI models can contain exploits that allow for arbitrary code execution. These files should be scanned for potentially unsafe calls, which could be used to execute code, create new processes, or establish networking capabilities. Adversaries may embed malicious code in model corrupt model files, so scanners should be capable of working with models that cannot be fully de-serialized. Model artifacts, downstream products produced by models, and external software dependencies should be scanned for known vulnerabilities.

2.1.7. AML.M0017 - AI Model Distribution Methods

Deploying AI models to edge devices can increase the attack surface of the system. Consider serving models in the cloud to reduce the level of access the adversary has to the model. Also consider computing features in the cloud to prevent gray-box attacks, where an adversary has access to the model preprocessing methods.

2.1.8. AML.M0019 - Control Access to AI Models and Data in Production

Require users to verify their identities before accessing a production model. Require authentication for API endpoints and monitor production model queries to ensure compliance with usage policies and to prevent model misuse.

2.1.9. AML.M0032 - Segmentation of AI Agent Components

Define security boundaries around agentic tools and data sources with methods such as API access, container isolation, code execution sandboxing, and rate limiting of tool invocation. This restricts untrusted processes or potential compromises from spreading throughout the system.

2.2. SAFE-AI

Link: https://atlas.mitre.org/pdf-files/SAFEAI_Full_Report.pdf

Link: https://atlas.mitre.org/pdf-files/SAFEAI_Full_Report.pdf

2.2.1. Insecure APIs

Insecure APIs can allow attackers unauthorized access, introduce malicious inputs, or disrupt AI systems. This includes risks like unauthorized data access and denial of service, as well as AI-specific threats such as manipulation of model inputs. AI systems often consist of both internal and externally facing APIs that need to be secured, so that data integrity is preserved as data is transmitted among the various components in the AI-enabled system. Standard mitigation strategies, such as data encryption, input validation, robust authentication and authorization mechanisms, are essential for ensuring security of both internal APIs and externally facing APIs.

Inference APIs are particularly vulnerable as they are often exposed to external users. Adversaries may exploit legitimate access to inference APIs to gather detailed information about model ontology, structure, and behavior, enabling black-box and white-box attacks. Attackers can refine adversarial techniques to bypass model defenses and evade detection capabilities to introduce malicious data, potentially leading to incorrect predictions or compromised decision-making.

2.2.2. Faulty authentication and authorization settings

Weak or improperly implemented authentication and authorization mechanisms can allow attackers to poison data, manipulate model input, or otherwise compromise the behavior of an AI component. Note that in AI-enabled systems, faulty settings for these mechanisms may be the result of deliberate attacks like model stealing and prompt extraction, or inadequate attention to data privacy during model development, testing, and deployment.

2.2.3. Insider threats

Insiders can exploit access privileges to engage in unauthorized activities, including data theft or sabotage of AI models and data. These insider attacks may be especially difficult to address for AI-enabled systems, since AI system development and documentation practices do not tend to employ the same process controls as traditional software development. Moreover, AI systems may require more frequent maintenance and triggers for conducting corrective maintenance due to factors like data, model, or concept drift. This points to the need for extra vigilance when it comes to things like data provenance and access control mechanisms and policies.

2.2.4. Denial of Service attack

Many AI-enabled systems require significant amounts of specialized computing resources. From an adversary's perspective, these computing resources can often be viewed as expensive bottlenecks that can be easily overloaded. Adversaries can exploit this vulnerability by flooding the system with inputs, or by intentionally crafting inputs that require heavy amounts of useless compute from the AI system. The increased computing load can eventually degrade or shut down the services supplied by the AI-enabled system. Mitigations include limiting the number of queries the AI system will handle at any one time and putting in place explicit monitors to detect adversarial input.

2.2.5. Network components attacks

The AI components in an AI-enabled system are often general-purpose capabilities that are customized for the needs of the system and its use cases. This means, in particular, that AI-enabled systems often do not have an adequately designed resilient security architecture. There are likely to be shortcomings regarding access controls and proper network configurations. Gaps in these capabilities need to be proactively identified and mitigated during the design, development and deployment phases of the AI lifecycle.

2.2.6. Power supply attacks

Power supply attacks are problematic for AI-enabled systems that need massive amounts of time and computing resources to process large volumes of complex data during some phase of the AI lifecycle. For example, this is a routine concern for AI pipelines that train modern deep learning systems such as large language models. While the AI architectures supporting these big data requirements are explicitly designed to provide safeguards like checkpoint and restore operations, the I/O bandwidth and storage needed to address these concerns are formidable. The AI concern here is arranging for the massive amount of resources needed for the safeguards, above and beyond what is needed to build and use the AI-enabled system itself.

2.2.7. Physical breaches

For AI-enabled systems, exploitation of the physical environment to attack the system can occur in a variety of ways. For example, an attacker may physically access the location where data is being collected and modify the collection process in ways that will compromise subsequent AI model training or performance. When the AI system receives input data from real world sensors, it may be possible to employ attacks that make malicious changes to the physical environment that will compromise system behavior (e.g., using physical domain patch-based attack to deceive a ML classification model). Mitigations include stringent approaches to detect physical tampering and unexpected vulnerabilities in AI components. Common anti-tamper technologies for software systems should be applied to AI-enabled systems if a physical breach is suspected. Since behavior patterns in AI components can be difficult to specify precisely, it may also be helpful to establish a variety of behavior baselines for AI components. Given a baseline of normal behavior, behavior analysis techniques could identify anomalous behavior patterns that might be useful indicators of tampering.

2.2.8. Identity Spoofing (e.g. deep fakes, synthetic identities, CAPTCHA threat)

The ability to generate new or altered identities using AI has become common place and represents a threat to some forms of identification and authentication (e.g., voice spoofing, keystroke dynamics, and biometrics). This is particularly true in phishing attacks designed to gain sensitive information that may put access control safeguards at risk. AI based systems can also mimic certain human inputs to break known CAPTCHA-types of systems, thereby increasing the vulnerability to compromise and fraud. Mitigations include stringent access controls, robust authentication and authorization mechanisms, and user education regarding the dangers of social engineering attacks like phishing.

2.2.9. Cost Harvesting

AI services tend to use large amounts of computing resources and consume a great deal of energy during response generation. Adversaries can maliciously increase the cost of running these services by flooding the system with useless queries, or by crafting computationally expensive inputs. For example, systems that rely on massive neural networks may be vulnerable to adversarial data (e.g., “sponge” examples) designed to activate large numbers of nodes in the hidden network layers.

2.2.10. Spamming the System with Chaff Data

Adversaries may spam AI applications with chaff data to flood them with false positives, overwhelming the system and increasing detections. This tactic forces analysts to waste time reviewing and correcting incorrect inferences, reducing their efficiency. Techniques include automated scripts, botnets, or tools like Faker to generate large volumes of synthetic data. By inundating the system with irrelevant data, adversaries aim to degrade performance and exhaust the resources.

3. OWASP

3.1. OWASP Top 10 for Agentic Applications for 2026

Link: <https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/>

Link: <https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/>

3.1.1. ASI01: Agent Goal Hijack - 2

Minimize the impact of goal hijacking by enforcing least privilege for agent tools and requiring human approval for high-impact or goal-changing actions.

3.1.2. ASI01: Agent Goal Hijack - 3

Define and lock agent system prompts so that goal priorities and permitted actions are explicit and auditable. Changes to changes in goals or reward definitions must go through configuration management and human approval.

3.1.3. ASI01: Agent Goal Hijack - 4

At run time, validate both user intent and agent intent before executing goal-changing or high-impact actions. Require confirmation - via human approval, policy engine, or platform guardrails whenever the agent proposes actions that deviate from the original task or scope. Pause or block execution on any unexpected goal shift, surface the deviation for review, and record it for audit.

3.1.4. ASI01: Agent Goal Hijack - 9

Incorporate AI Agents into the established Insider Threat Program to monitor any insider prompts intended to get access to sensitive data or to alter the agent behavior and allow for investigation in case of outlier activity.

3.1.5. ASI02: Tool Misuse and Exploitation - 3

Execution Sandboxes and Egress Controls. Run tool or code execution in isolated sandboxes. Enforce outbound allowlists and deny all non-approved network destinations.

3.1.6. ASI02: Tool Misuse and Exploitation - 4

Policy Enforcement Middleware (“Intent Gate”). Treat LLM or planner outputs as untrusted. A pre-execution Policy Enforcement Point (PEP/PDP) validates intent and arguments, enforces schemas and rate limits, issues short-lived credentials, and revokes or audits on drift.

3.1.7. ASI02: Tool Misuse and Exploitation - 5

Adaptive Tool Budgeting. Apply usage ceilings (cost, rate, or token budgets) with automatic revocation or throttling when exceeded.

3.1.8. ASI02: Tool Misuse and Exploitation - 6

Just-in-Time and Ephemeral Access. Grant temporary credentials or API tokens that expire immediately after use. Bind keys to specific user sessions to prevent lateral abuse.

3.1.9. ASI02: Tool Misuse and Exploitation - 7

Semantic and Identity Validation (“Semantic Firewalls”). Enforce fully qualified tool names and version pins to avoid tool alias collisions or typo squatted tools; validate the intended semantics of tool calls (e.g., query type or category) rather than relying on syntax alone. Fail closed on ambiguous resolution and prompt for user disambiguation.

3.1.10. ASI04: Agentic Supply Chain Vulnerabilities - 3

Containment and builds: Run sensitive agents in sandboxed containers with strict network or syscall limits; require reproducible builds.

3.1.11. ASI04: Agentic Supply Chain Vulnerabilities - 5

Inter-agent security: Enforce mutual auth and attestation via PKI and mTLS; no open registration; sign and verify all inter-agent messages.

3.1.12. ASI05: Unexpected Code Execution (RCE) - 4

Execution environment security: Never run as root. Run code in sandboxed containers with strict limits including network access; lint and block known-vulnerable packages and use framework [Page 23genai.owasp.org](https://github.com/OWASP/genai-owasp.org) sandboxes like mcp-run-python. Where possible, restrict filesystem access to a dedicated working directory and log file diffs for critical paths.

3.1.13. ASIO5: Unexpected Code Execution (RCE) - 5

Architecture and design: Isolate per-session environments with permission boundaries; apply least privilege; fail secure by default; separate code generation from execution with validation gates.

3.1.14. ASIO6: Memory & Context Poisoning - 1

Baseline data protection: Encryption in transit and at rest combined with least-privilege access.

3.1.15. ASIO6: Memory & Context Poisoning - 3

Memory segmentation: Isolate user sessions and domain contexts to prevent knowledge and sensitive data leakage.

3.1.16. ASIO6: Memory & Context Poisoning - 4

Access and retention: Allow only authenticated, curated sources; enforce context-aware access per task; minimize retention by data sensitivity.

3.1.17. ASIO7: Insecure Inter-Agent Communication - 1

Secure agent channels: Use end-to-end encryption with per-agent credentials and mutual authentication. Enforce PKI certificate pinning, forward secrecy, and regular protocol reviews to prevent interception or spoofing.

3.1.18. ASIO7: Insecure Inter-Agent Communication - 2

Message integrity and semantic protection: Digitally sign messages, hash both payload and context, and validate for hidden or modified natural-language instructions. Apply natural-language-aware sanitization and intent-diffing to detect goal, parameter tampering, hidden or modified natural-language instructions.

3.1.19. ASIO7: Insecure Inter-Agent Communication - 3

Agent-aware anti-replay: Protect all exchanges with nonces, session identifiers, and timestamps tied to task windows. Maintain short-term message fingerprints or state hashes to detect cross-context replays.

3.1.20. ASIO7: Insecure Inter-Agent Communication - 4

Protocol and capability security: Disable weak or legacy communication modes. Require agent-specific trust negotiation and bind protocol authentication to agent identity. Enforce version and capability policies at gateways or middleware.

3.1.21. ASIO7: Insecure Inter-Agent Communication - 5

Limit metadata-based inference: Reduce the attack surface for traffic analysis by using fixed-size or padded messages where feasible, smoothing communication rates, and avoiding deterministic communication schedules. These lightweight measures make it harder for attackers to infer agent roles or decision cycles from metadata alone, without requiring heavy protocol redesign.

3.1.22. ASIO7: Insecure Inter-Agent Communication - 6

Protocol pinning and version enforcement: Define and enforce allowed protocol versions (e.g., MCP, A2A, gRPC). Reject downgrade attempts or unrecognized schemas and validate that both peers advertise matching capability and version fingerprints.

3.1.23. ASIO7: Insecure Inter-Agent Communication - 7

Discovery and routing protection. Authenticate all discovery and coordination messages using cryptographic identity. Secure directories with access controls and verified reputations, validate identity and intent end-to-end, and monitor for anomalous routing flows.

3.1.24. ASIO8: Cascading Failures - 2

Isolation and trust boundaries: Sandbox agents, least privilege, network segmentation, scoped APIs, and mutual auth. to contain failure propagation.

3.1.25. ASIO8: Cascading Failures - 3

JIT, one-time tool access with runtime checks: Issue short-lived, task-scoped credentials for each agent run and validate every high-impact tool invocation against a policy-as-code rule before executing it. This ensures a compromised or drifting agent cannot trigger chain reactions across other agents or systems.

3.1.26. ASIO8: Cascading Failures - 6

Rate limiting and monitoring: Detect fast-spreading commands and throttle or pause on anomalies.

3.1.27. ASI08: Cascading Failures - 7

Implement blast-radius guardrails such as quotas, progress caps, circuit breakers between planner and executor.

3.1.28. ASI09: Human-Agent Trust Exploitation - 6

Content provenance and policy enforcement: Attach verifiable metadata-source identifiers, timestamps, and integrity hashes-to all recommendations and external data. Enforce digital signature validation and runtime policy checks that block actions lacking trusted provenance or exceeding the agent's declared scope.

3.1.29. ASI09: Human-Agent Trust Exploitation - 7

Separate preview from effect: Block any network or state-changing calls during preview context and display a risk badge with source provenance and expected side effects.

3.1.30. ASI10: Rogue Agents - 2

Isolation & Boundaries: Assign Trust Zones with strict inter-zone communication rules and deploy restricted execution environments (e.g., container sandboxes) with API scopes based on least privilege.

3.1.31. ASI10: Rogue Agents - 6

Require periodic behavioral attestation: challenge tasks, signed bill of materials for prompts and tools, and per-run ephemeral credentials with one-time audience binding. All signing and attestation mechanisms assume hardened cryptographic key management (e.g., HSM/KMS-backed keys, least- Page 38genai.owasp.org privilege access, rotation and revocation). Keys must never be directly available to agents; instead, orchestrators should mediate signing operations so that a compromised agent cannot simply exfiltrate or misuse long-lived keys

3.2. LLM Top 10

Link: <https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/>

Link: <https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/>

3.2.1. LLM02: Sensitive Information Disclosure - 3

Limit access to sensitive data based on the principle of least privilege. Only grant access to data that is necessary for the specific user or process.

3.2.2. LLM02: Sensitive Information Disclosure - 5

Train models using decentralized data stored across multiple servers or devices. This approach minimizes the need for centralized data collection and reduces exposure risks.

3.2.3. LLM04: Data and Model Poisoning - 3

Implement strict sandboxing to limit model exposure to unverified data sources. Use anomaly detection techniques to filter out adversarial data.

3.2.4. LLM04: Data and Model Poisoning - 5

Ensure sufficient infrastructure controls to prevent the model from accessing unintended data sources.

3.2.5. LLM05: Improper Data Handling - 5

Use parameterized queries or prepared statements for all database operations involving LLM output.

3.2.6. LLM05: Improper Data Handling - 6

Employ strict Content Security Policies (CSP) to mitigate the risk of XSS attacks from LLM-generated content.

3.2.7. LLM07: System Prompt Leakage - 1

Avoid embedding any sensitive information (e.g. API keys, auth keys, database names, user roles, permission structure of the application) directly in the system prompts. Instead, externalize such information to the systems that the model does not directly access.

3.2.8. LLM07: System Prompt Leakage - 2

Since LLMs are susceptible to other attacks like prompt injections which can alter the system prompt, it is recommended to avoid using system prompts to control the model behavior where possible. Instead, rely on systems outside of the LLM to ensure this behavior. For example, detecting and preventing harmful content should be done in external systems.

3.2.9. LLM07: System Prompt Leakage - 3

Implement a system of guardrails outside of the LLM itself. While training particular behavior into a model can be effective, such as training it not to reveal its system prompt, it is not a guarantee that the model will always adhere to this. An independent system that can inspect the output to determine if the model is in compliance with expectations is preferable to system prompt instructions.

3.2.10. LLM07: System Prompt Leakage - 4

Critical controls such as privilege separation, authorization bounds checks, and similar must not be delegated to the LLM, either through the system prompt or otherwise. These controls need to occur in a deterministic, auditable manner, and LLMs are not (currently) conducive to this. In cases where an agent is performing tasks, if those tasks require different levels of access, then multiple agents should be used, each configured with the least privileges needed to perform the desired tasks.

3.2.11. LLM08: Vector and Embedding Weaknesses - 1

Implement fine-grained access controls and permission-aware vector and embedding stores. Ensure strict logical and access partitioning of datasets in the vector database to prevent unauthorized access between different classes of users or different groups.

3.2.12. LLM10: Unbounded Consumption - 3

Apply rate limiting and user quotas to restrict the number of requests a single source entity can make in a given time period.

3.2.13. LLM10: Unbounded Consumption - 4

Monitor and manage resource allocation dynamically to prevent any single user or request from consuming excessive resources.

3.2.14. LLM10: Unbounded Consumption - 5

Set timeouts and throttle processing for resource-intensive operations to prevent prolonged resource consumption.

3.2.15. LLM10: Unbounded Consumption - 10

Implement restrictions on the number of queued actions and total actions, while incorporating dynamic scaling and load balancing to handle varying demands and ensure consistent system performance.

3.2.16. LLM10: Unbounded Consumption - 13

Implement strong access controls, including role-based access control (RBAC) and the principle of least privilege, to limit unauthorized access to LLM model repositories and training environments.

3.2.17. LLM10: Unbounded Consumption - 15

Implement automated MLOps deployment with governance, tracking, and approval workflows to tighten access and deployment controls within the infrastructure.

3.3. OWASP Model Context Protocol (MCP) Top 10

Link: <https://owasp.org/www-project-mcp-top-10/>

Link: <https://owasp.org/www-project-mcp-top-10/>

3.3.1. MCP04:2025 – Software Supply Chain Attacks & Dependency Tampering - 4

Version Pinning & Approved Registries

- Pin component versions — avoid “latest”
- Use internal package mirrors or registries
- Block direct downloads from the public internet

3.3.2. MCP04:2025 – Software Supply Chain Attacks & Dependency Tampering - 6

Sandbox Third-Party Plugins

- Run plugins in constrained environments (e.g., WASM, container isolation)
- Restrict filesystem + network access

3.3.3. MCP05:2025 – Command Injection & Execution - 3

Sandbox All Tools

- Run tools inside containers, micro-VMs, gVisor/Kata, or jailed users.
- Enforce timeouts, resource limits, and read-only file systems. Isolate high-risk tools (file system, network, DB) into separate sandboxes.

3.3.4. MCP05:2025 – Command Injection & Execution - 5

Strong Validation at Tool Boundaries Validate agent output against schemas before execution. Use parameterized SQL/APIs — never interpolate input. Reject unsafe patterns: chained commands, redirection, wildcards, command substitution.

3.3.5. MCP07:2025 – Insufficient Authentication & Authorization - 1

Strong Authentication for All Entities

- Require mutual TLS (mTLS) between MCP clients, agents, and servers.
- Use short-lived, scoped tokens (JWT/OAuth2-style) tied to specific sessions and permissions.
- Enforce token binding to agent identity (e.g., signed agent attestation).
- Validate every token on the server side — never trust client-provided claims.

3.3.6. MCP07:2025 – Insufficient Authentication & Authorization - 2

Implement Fine-Grained Authorization

- Adopt RBAC (roles) or ABAC (attributes) models: Example: “Agent X may read customer data but not execute tools.”
- Evaluate permissions per request, not per session.
- Deny-by-default: any unrecognized agent or scope should be blocked automatically.

3.3.7. MCP07:2025 – Insufficient Authentication & Authorization - 3

Token Lifecycle Management

- Enforce expiration, rotation, and revocation policies for all tokens.
- Store tokens securely (vaulted or encrypted).
- Detect and block replayed or duplicated tokens.

3.3.8. MCP07:2025 – Insufficient Authentication & Authorization - 5

Centralized Identity & Access Management

- Integrate MCP authentication with organizational IAM or OIDC providers.
- Require federated identity for all user-driven and system-driven actions.
- Centralize policy enforcement through a Policy Decision Point (PDP).

3.3.9. MCP07:2025 – Insufficient Authentication & Authorization - 7

Secure-by-Default Configurations

- Disable guest or anonymous access in all MCP endpoints.
- Prevent local testing servers from exposing endpoints publicly.
- Enforce environment-specific credentials for dev/test/prod.

3.3.10. MCP07:2025 – Insufficient Authentication & Authorization - 8

Revoke all compromised or static tokens immediately.

3.3.11. MCP07:2025 – Insufficient Authentication & Authorization - 9

Rotate all service credentials and enforce unique per-agent identities.

3.3.12. MCP07:2025 – Insufficient Authentication & Authorization - 10

Enable mTLS and strict API key binding.

3.3.13. MCP07:2025 – Insufficient Authentication & Authorization - 13

Add temporary compensating controls: IP restrictions, manual approvals for sensitive actions.

3.3.14. MCP09:2025 – Shadow MCP Servers - 1

Establish Central MCP Governance & Registry

- Create a centralized MCP registry where every instance must be registered before deployment.
- Tie registration to CI/CD pipelines — any unregistered instance should fail deployment.
- Maintain metadata: owner, purpose, version, endpoints, compliance state, and contact.
- Require approval and risk classification for each new MCP instance.

3.3.15. MCP09:2025 – Shadow MCP Servers - 2

Implement Discovery & Continuous Scanning

- Use network discovery tools (Nmap, Shodan internal equivalents, CSPM, or EASM tools) to detect open MCP ports and endpoints.
- Deploy passive network sensors to identify MCP traffic patterns (unique protocol identifiers, routes).
- Integrate discovery results with asset inventories and vulnerability management platforms.
- Automate shadow MCP detection scans weekly with alerts to the security operations team.

3.3.16. MCP09:2025 – Shadow MCP Servers - 3

Define Baseline Configuration Templates

- Publish secure-by-default MCP configuration templates for teams:
- Enforce authentication and authorization (mTLS, OAuth).
- Disable unauthenticated tool calls and external access by default.
- Include preconfigured logging, rate-limits, and monitoring agents.
- Block deployment of MCP instances that deviate from approved templates.

3.3.17. MCP09:2025 – Shadow MCP Servers - 4

Enforce Identity & Access Management (IAM) Controls

- Require all MCP instances to integrate with central IAM providers (SSO, LDAP, or OIDC).
- Use service identities bound to teams and enforce role-based access.
- Apply network segmentation (VPC-level controls, firewall rules) to limit exposure.

3.3.18. MCP09:2025 – Shadow MCP Servers - 7

Policy & Enforcement

- Integrate MCP governance into corporate IT and AI Acceptable Use Policies (AUPs).
- Require sign-off from information security before deployment of any model-serving or context protocol infrastructure.
- Periodically audit compliance and enforce disciplinary or procedural action for unauthorized setups.

3.3.19. MCP09:2025 – Shadow MCP Servers - 9

Contain the detected shadow MCP (disable network access, snapshot for forensics).

3.3.20. MCP09:2025 – Shadow MCP Servers - 10

Identify owners and isolate associated credentials or API keys.

3.3.21. MCP09:2025 – Shadow MCP Servers - 13

Enforce registration and compliance checks before re-enabling access.

3.3.22. MCP09:2025 – Shadow MCP Servers - 14

Update network segmentation and discovery coverage to prevent recurrence.

3.3.23. MCP10:2025 – Context Injection & Over-Sharing - 1

Use Ephemeral Contexts

- Make context windows short-lived and per session by default.
- Enforce automatic deletion after task completion.
- Avoid persistent memory unless explicitly sanctioned and governed.

3.3.24. MCP10:2025 – Context Injection & Over-Sharing - 4

Context Expiry and TTL Enforcement

Define time-to-live (TTL) policies such as:

Session end

30 minutes

24 hours max Automatically purge expired contexts and embeddings.

4. European Commission

4.1. Ethics guidelines for trustworthy AI

Link: <https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai>

Link: <https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai>

4.1.1. 1.2.2 Fallback plan and general safety

AI systems should have safeguards that enable a fallback plan in case of problems. This can mean that AI systems switch from a statistical to rule-based procedure, or that they ask for a human operator before continuing their action. It must be ensured that the system will do what it is supposed to do without harming living beings or the environment.

4.1.2. 1.3.3 Access to data

In any given organisation that handles individuals' data (whether someone is a user of the system or not), data protocols governing data access should be put in place. These protocols should outline who can access data and under which circumstances. Only duly qualified personnel with the competence and need to access individual's data should be allowed to do so.

5. CoSAI

5.1. Establish Risks and Controls for the AI Supply Chain

Link: <https://github.com/cosai-oasis/ws1-supply-chain/blob/main/risks-and-controls-for-the-ai-supply-chain-v1.md>

Link: <https://github.com/cosai-oasis/ws1-supply-chain/blob/main/risks-and-controls-for-the-ai-supply-chain-v1.md>

5.1.1. 3.1.2 Model Training - Model Source Tampering

Unauthorized modification of model architecture code to introduce backdoors:
Two-person code review requirements, secure source repositories with access controls

5.1.2. 3.1.2 Model Training - Model Weight Tampering

Unauthorized modification of model weights after training, during checkpoints, or in pre-trained models:
Access controls for model storage, cryptographic model signatures

5.1.3. 3.1.3 Application Integration in the AI Supply Chain - Vulnerable Communication Channels

Interception or manipulation of model-application traffic:
End-to-end encryption, certificate validation, response signing

5.1.4. 3.1.3 Application Integration in the AI Supply Chain - Third-Party API Compromise

Security failures in external services integrated into the application:
API security assessments, traffic monitoring, fallback mechanisms

5.1.5. 3.2.3 Application - Indirect Prompt Injection

Unauthorized instructions or malicious commands introduced to the model through external data sources during inference operations, including RAG systems, agentic frameworks, web search functionality, tool outputs, and other integrated data providers:
Implement comprehensive access control frameworks with strong authentication mechanisms for all external data sources. Deploy advanced content filtering systems with specific detection capabilities for injection patterns across all integrated data channels.

5.1.6. 3.2.4 Infrastructure - Cache Poisoning

Malicious manipulation of cached data or results within the AI infrastructure, where attackers introduce corrupted information into temporary storage mechanisms used to optimize model performance:
Implement comprehensive cache validation protocols, cryptographic verification of cached content, and regular cache refresh cycles with integrity checks.

5.1.7. 3.2.4 Infrastructure - Overly Permissive Entitlements

Excessive access rights within the AI infrastructure allowing components or users to access, modify, or control resources beyond operational requirements, creating unnecessary attack surfaces:
Implement comprehensive least-privilege architecture, regularly audit entitlement configurations, enforce just-in-time access protocols, and establish proper service account isolation.

5.2. AI Incident Response Framework

Link: <https://github.com/cosai-oasis/ws2-defenders/blob/main/incident-response/AI%20Incident%20Response.md>

Link: <https://github.com/cosai-oasis/ws2-defenders/blob/main/incident-response/AI%20Incident%20Response.md>

5.2.1. 3.3.2. Detection and Analysis Phase - Detection Mechanisms - Integration Points

- SIEM integration
- Correlation with network security
- Authentication event linking

5.2.2. 3.3.3. Containment, Eradication, and Recovery Phase - Containment Strategies - Short-term Containment

- Emergency prompt filters
- Rate limiting/access restrictions
- Component isolation
- User/IP blocking

5.2.3. 3.3.3. Containment, Eradication, and Recovery Phase - Eradication Procedures - Component Remediation

- Data-Level: Remove poisoned data, reconstruct embeddings
- Model-Level: Roll back to secure versions, enhance guardrails
- Deployment-Level: Update prompts, strengthen controls
- Output-Level: Implement filtering, content moderation

5.3. Model Context Protocol (MCP) Security

Link: <https://github.com/cosai-oasis/ws4-secure-design-agentic-systems/blob/main/model-context-protocol-security.md>

Link: <https://github.com/cosai-oasis/ws4-secure-design-agentic-systems/blob/main/model-context-protocol-security.md>

5.3.1. 3.2.1 Agent Identity

All requests should be traceable across the entire execution chain: the end user or initiating agent, any intermediate MCP servers, and the tools or services that performed the resulting actions. Standards are emerging to define the identity of agents and servers. One of these is SPIFFE / SPIRE, which provides cryptographic workload identities that can be granted authorization to resources. The SPIFFE ID can be used in token exchange as the subject or actor depending on the flow.

Secure identity, authentication, and authorization across the agentic and MCP ecosystem is an extremely active area of research and development. We will provide a much deeper analysis of the problem space in a subsequent white paper.

5.3.2. 3.2.4 Cryptographic Integrity and Remote Attestation

Hardware Trusted Execution Environments (TEE) like Intel TDX, and AMD-SEV/SNP provide stronger isolation and can provide protection against runtime tampering of trusted servers, such as tool poisoning due to server compromise. Combined with remote attestation, TEEs can isolate MCP servers and clients from compromised hardware, malicious administrators of server infrastructure, and certain classes of co-tenancy threats. For containerized deployments, consider the use of confidential containers (CoCo) that run containers in TEEs, and use remote attestation to verify the trustworthiness of the TEEs and what is running in them.

5.3.3. 3.2.5 Sandboxing and Isolation

Agents and MCP servers should be executed with least privilege. MCP servers that interact with the host environment (e.g. by accessing files, running commands, issuing network connections), or that execute LLM-generated code, should always run in a sandbox to mitigate against potential safety and security threats.

LLM-generated code and commands may contain hallucinations, bugs, or vulnerabilities, and should not run with full user privileges. MCP servers are commonly deployed in containers for ease of use, but containers should not be relied upon as a strong security boundary. Consider additional sandboxing (gVisor, Kata Containers, SELinux sandboxes) for stronger isolation.

5.3.4. 3.2.7 Transport Layer Security

MCP is structured around distinct communication layers that facilitate robust interaction between systems. At the transport layer, MCP leverages two primary communication methods:

stdio Transport:

A direct, pipe-based stream communication channel, typically used for intra-process or tightly integrated inter-process communications. JSON-RPC messages flow directly via standard input/output streams. This transport is most commonly used for local servers.

HTTP Streaming Transport:

A generalized HTTP-based transport channel supporting bidirectional JSON-RPC communication via streamed request-response patterns. This transport is most commonly used for remote servers.

At the higher-level protocol layer, MCP employs JSON-RPC 2.0 to standardize the formatting and processing of commands and responses communicated across these transport channels. JSON-RPC ensures structured messaging, enabling interoperability and clarity of communication across diverse platforms.

However, these transport and protocol layers, when improperly secured or configured, can expose MCP clients and servers to multiple vulnerabilities.

5.3.5. 3.2.11 Lifecycle and Governance a)

Organizations must:

- implement mandatory code signing verification for all MCP servers before installation,
- use private package repositories with security scanning and approval workflows,
- deploy software composition analysis (SCA) tools to detect vulnerable dependencies,
- implement allow-lists of approved MCP servers with documented security reviews,
- Run MCP servers and clients in TEEs and use remote attestation to verify prior to interactions,
- use cryptographic hash verification for package integrity,
- and deploy binary authorization that prevents execution of unsigned or unverified code.

5.3.6. 3.2.11 Lifecycle and Governance c)

Lifecycle management demands:

- maintaining centralized inventory of all deployed MCP servers with metadata (version, owner, purpose),
- implementing automated discovery to detect shadow or unauthorized MCP deployments,
- deploying lifecycle policies that automatically deprecate or remove outdated servers,
- using configuration management tools (Ansible, Puppet, Chef) to maintain consistent deployments,
- implementing rollback capabilities for problematic updates,
- and deploying update management processes with testing and staged rollout.

6. IBM

6.1. IBM Framework for Securing Generative AI

Link: <https://www.ibm.com/products/tutorials/ibm-framework-for-securing-generative-ai>

Link: <https://www.ibm.com/products/tutorials/ibm-framework-for-securing-generative-ai>

6.1.1. Secure the model

Within model development, you're building applications in a new way, and that often involves introducing new, exploitable vulnerabilities that attackers can use as entry points into the environment and, in turn, into your AI models. Considering that organizations have historically struggled with managing a growing debt of known vulnerabilities found within their environments, this risk will carry over to AI.

Developing AI applications often starts with data science teams repurposing pretrained, open-source machine learning (ML) models from online model repositories, which often lack comprehensive security controls. However, the value they provide organizations, such as dramatically reducing the time and effort required for generative AI adoption, often outweighs that risk, ultimately passing it on to the enterprise. The general scarcity of security around ML models, coupled with the increasingly sensitive data that ML models are exposed to, means that attacks targeting these models have a high potential for damage.

The primary attack techniques during model development are supply chain attacks due to the heavy reliance on pretrained, open-source ML models from online model repositories used to accelerate development efforts. Attackers have the same access to these online repositories and can deploy a backdoor or malware into them. Once uploaded back into the repository, they can become an entry point to anyone that downloads the infected model. If these models are infected, it can be incredibly difficult to detect. Organizations must be very cautious about where they consume models and how trusted the source is.

Application programming interface (API) attacks are another concern. Organizations without the resources or expertise to build their own large language models (LLMs) rely on APIs to consume the capabilities of prepackaged, pretrained models. Attackers recognize this will be a major consumption model for LLMs and will look to target the API interfaces to access and exploit data being transported across the APIs.

Attackers may also seek to exploit LLM agents or plug-ins with excessive permissions to access open-ended functions or downstream systems that can perform privileged actions in business workflows. If an attacker can compromise privileges that are granted to AI agents, the damage could be destructive.

Organizations' focus should include:

- Continuously scanning for vulnerabilities, malware and corruption across the AI/ML pipeline

- Discovering and hardening API and plug-in integrations to third-party models

- Configuring enforcing policies, controls and RBAC around ML models, artifacts and data sets so that no one person or thing has access to all the data or all of the model functions

6.1.2. Secure the infrastructure

One of the first lines of defense is a secure infrastructure. Organizations should leverage existing expertise to optimize security, privacy and compliance standards across distributed environments hosting the AI systems. It's essential that they harden network security, access control, data encryption, and intrusion detection and prevention around AI environments. They should also consider investing in new security defenses specifically designed to protect AI.

7. Model Context Protocol

7.1. Security Best Practices

Link: https://modelcontextprotocol.io/specification/2025-11-25/basic/security_best_practices

Link: https://modelcontextprotocol.io/specification/2025-11-25/basic/security_best_practices

7.1.1. Confused Deputy Problem

Attackers can exploit MCP proxy servers that connect to third-party APIs, creating "confused deputy" vulnerabilities. This attack allows malicious clients to obtain authorization codes without proper user consent by exploiting the combination of static client IDs, dynamic client registration, and consent cookies.

To prevent confused deputy attacks, MCP proxy servers MUST implement per-client consent and proper security controls as detailed below.

7.1.2. Token Passthrough

"Token passthrough" is an anti-pattern where an MCP server accepts tokens from an MCP client without validating that the tokens were properly issued to the MCP server and passes them through to the downstream API.

MCP servers MUST NOT accept any tokens that were not explicitly issued for the MCP server.

7.1.3. Server-Side Request Forgery (SSRF)

Server-Side Request Forgery (SSRF) is an attack where an attacker can induce an MCP client to make HTTP requests to unintended destinations, potentially accessing internal network resources, cloud metadata endpoints, or other protected services.

MCP clients deployed to a server MUST consider SSRF risks and implement appropriate mitigations when fetching OAuth-related URLs. Which protections are appropriate depend on your network environment.

7.1.4. Session Hijacking

Session hijacking is an attack vector where a client is provided a session ID by the server, and an unauthorized party is able to obtain and use that same session ID to impersonate the original client and perform unauthorized actions on their behalf.

To prevent session hijacking and event injection attacks, the following mitigations should be implemented:

MCP servers that implement authorization MUST verify all inbound requests. MCP Servers MUST NOT use sessions for authentication.

MCP servers MUST use secure, non-deterministic session IDs. Generated session IDs (e.g., UUIDs) SHOULD use secure random number generators. Avoid predictable or sequential session identifiers that could be guessed by an attacker. Rotating or expiring session IDs can also reduce the risk.

MCP servers SHOULD bind session IDs to user-specific information. When storing or transmitting session-related data (e.g., in a queue), combine the session ID with information unique to the authorized user, such as their internal user ID. Use a key format like `session_id: user_id`. This ensures that even if an attacker guesses a session ID, they cannot impersonate another user as the user ID is derived from the user token and not provided by the client.

MCP servers can optionally leverage additional unique identifiers.

7.1.5. Local MCP Server Compromise

Local MCP servers are MCP Servers running on a user's local machine, either by the user downloading and executing a server, authoring a server themselves, or installing through a client's configuration flows. These servers may have direct access to the user's system and may be accessible to other processes running on the user's machine, making them attractive targets for attacks.

If an MCP client supports one-click local MCP server configuration, it MUST implement proper consent mechanisms prior to executing commands.

7.1.6. Scope Minimization

Poor scope design increases token compromise impact, elevates user friction, and obscures audit trails.

Implement a progressive, least-privilege scope model:

- Minimal initial scope set (e.g., `mcp:tools-basic`) containing only low-risk discovery/read operations

- Incremental elevation via targeted `WWW-Authenticate scope=...` challenges when privileged operations are first attempted

- Down-scoping tolerance: server should accept reduced scope tokens; auth server MAY issue a subset of requested scopes

Server guidance:

- Emit precise scope challenges; avoid returning the full catalog

- Log elevation events (scope requested, granted subset) with correlation IDs

Client guidance:

- Begin with only baseline scopes (or those specified by initial `WWW-Authenticate`)

- Cache recent failures to avoid repeated elevation loops for denied scopes

8. CISA

8.1. Principles for the Secure Integration of Artificial Intelligence in Operational Technology

Link: <https://www.cisa.gov/sites/default/files/2026-01/joint-guidance-principles-for-the-secure-integration-of-artificial-intelligence-in-operational-technology-508cV2.pdf>

Link: <https://www.cisa.gov/sites/default/files/2026-01/joint-guidance-principles-for-the-secure-integration-of-artificial-intelligence-in-operational-technology-508cV2.pdf>

8.1.1. 1.2.3 Secure Deployment

Deploy the AI system using methods that maintain its security posture, including using proper network segmentation and access control, as well as verifying and validating that the AI system works as intended.

8.1.2. 2.2.5 - Data Silos

Address the complexity of integrating AI systems in OT environments due to OT/IT network segregation, proprietary protocols, formats, diversity of OT products, and a multitude of suppliers.

8.1.3. 2.4.1

When integrating AI into OT environments, critical infrastructure owners and operators should carefully evaluate the existing infrastructure to ensure compatibility and security.

8.1.4. 4.1.7 - Prefer push-based or brokered architectures that move required features or summaries out of OT without granting persistent inbound access.

Where data must traverse to business networks, use one-way transfer patterns and audited staging buffers. This method for implementing segmentation helps operators maintain existing segmentation best practices, such that the AI system is not a persistent attack path into OT.

9. SANS

9.1. Critical AI Security Guidelines

Link: <https://sansorg.egnyte.com/dl/bvkYQxrW8QMj>

Link: <https://sansorg.egnyte.com/dl/bvkYQxrW8QMj>

9.1.1. 1.1 Protect Your Model Parameters

It is critical to ensure traditional security controls, such as principle of least privilege and strong access controls with accountability, have been implemented. Should an unauthorized individual be able to replace or modify a deployed model, untold damage can result. Although this applies to any kind of AI model developed and deployed by an enterprise—including training models, which are highly susceptible to poisoning and attacks—consider the example of a generative agentic system leveraging a large language model (LLM)

9.1.2. 3.2 AI Deployment in Integrated Development Environments (IDEs)

IDEs such as VSCode, Windsurf, or Cursor are fully integrated with models or offer LLM integration as a highly desirable option. Although these integrations can significantly increase the efficiency and output of developers, users can inadvertently expose proprietary algorithms, models, API keys, and datasets through AI-powered features. Organizations should explore IDEs with local-only LLM integrations to mitigate risk exposure when local-only LLM integration becomes available. This control ensures that sensitive data remains secure and protected.

9.1.3. 4.9 Control and Monitor Access to Interaction/Inference

Depending on the use case and deployment of the LLM application, authentication and access controls should be implemented where appropriate. For public-facing applications, such as help or chatbots to aid or guide website visitors, there is no need to require authentication for user interaction. For external-facing applications, the front-end must authenticate to the LLM back-end, and as such, these calls should have audit and access logging enabled. Similarly, internal LLM applications, or those containing sensitive data, should be used with authentication and access controls, with auditing enabled by default in an enterprise environment. The ability to interact with an LLM application and have the model perform inference should be restricted, according to business use cases. Unless absolutely necessary, unauthenticated and/or unmonitored access to LLM APIs or front-ends should not be allowed.

9.1.4. 4.10 Monitor/Control API Usage

Abuse of LLMs can occur via multiple means. Prompt injection protection and detection methods primarily focus on the content of the input. Content validation, monitoring, and filtering also should be complemented with usage and behavior monitoring focused on the interactions themselves. LLM API keys should be properly managed under robust, secure software development policies, such as no hardcoding of keys in applications. Observing API usage for misuse is critical. Anomalous spikes in API usage can serve as an effective detection method for abuse, while rate limiting should be considered to restrict the number and cadence of interactions allowed. In addition to rate limiting, organizations also should consider other forms of behavior/anomaly detection. Although not limited to interactions through APIs, adversaries can easily automate inputs to exposed API endpoints, making them more susceptible to volumetric attacks. To mitigate this risk, internal training or inference API endpoints should not be public facing.

10. IMDA

10.1. Model AI Governance Framework for Agentic AI

Link: <https://www.imda.gov.sg/-/media/imda/files/about/emerging-tech-and-research/artificial-intelligence/mgf-for-agentic-ai.pdf>

Link: <https://www.imda.gov.sg/-/media/imda/files/about/emerging-tech-and-research/artificial-intelligence/mgf-for-agentic-ai.pdf>

10.1.1. 2.1.2 Bound risks through design by defining agents limits and permissions - Agent identity

Identity management and access control is one of the key means in which organisations enable traceability and accountability today for humans. As agents become more autonomous, identity management has to be extended to agents as well to track individual agent behaviour and establish who holds accountability for each agent.

This is an evolving space, and gaps exist today in terms of handling agent identity robustly. For example, current authorisation systems typically have pre-defined, static scopes. However, to operate safely in more complex scenarios, agents require fine-grained permissions that may change dynamically depending on the context, risk levels, and task objectives. Current authentication systems are also typically based on a single, unique individual. Such systems face difficulty in handling complex agent setups, such as when agents act for multiple human users with different permissions, or recursive delegation scenarios where agents spin up multiple sub-agents.

In the interim, organisations should consider these best practices to enable agent control and traceability:

- Identification: An agent should have its own unique identity, such that it can identify itself to the organisation, its human user, or other agents. However, an agent's identity may need to be tied to a supervising agent, a human user, or an organisational department for accountability and tracking. Additionally, the different capacities in which an agent acts (e.g. independently or on behalf of a specified human user) should also be recorded.
- Authorisation: An agent can have pre-defined permissions based on its role or the task at hand, or its permissions may be dynamically set by its authorising human user, or a combination of both. As a rule of thumb, the human user should not be able to set permissions for the agent greater than what the human user is himself authorised to do. Such delegations of authority should be clearly recorded.

11. Qatar Central Bank

11.1. Artificial Intelligence Guidelines

Link: https://www.qcb.gov.qa/Services/Financial%20Technology/QCB_Artificial_Intelligence_Guideline.pdf

Link: https://www.qcb.gov.qa/Services/Financial%20Technology/QCB_Artificial_Intelligence_Guideline.pdf

11.1.1. 13.6.2

The AI System must have built in guard rails or specific limits that the AI cannot override.

11.1.2. 13.6.3

Entities must review the guard rails and specific limits on a frequent set schedule or in response to external volatility spikes.

12. CEN/CENELEC

12.1. prEN 40000-1-2: Cybersecurity requirements for products with digital elements - Part 1-2: Principles for cyber resilience

Link: <https://genorma.com/en/standards/pren-40000-1-2>

Link: <https://genorma.com/en/standards/pren-40000-1-2>

12.1.1. 7.4

Cybersecurity architecture and design

13. ENISA

13.1. Multilayer Framework for Good Cybersecurity Practices for AI

Link:

<https://www.enisa.europa.eu/sites/default/files/publications/Multilayer%20Framework%20for%20Good%20Cybersecurity%20Practices%20for%20AI.pdf>

Link:

<https://www.enisa.europa.eu/sites/default/files/publications/Multilayer%20Framework%20for%20Good%20Cybersecurity%20Practices%20for%20AI.pdf>

13.1.1. Model or data disclosure

Model or data disclosure can be protected by applying proper access control and federated learning to minimise the risk of data breaches. Similarly, to reduce the level of compromise of ML application components, these should be compliant with protection policies, fully integrated to existing security operations and asset management processes, and evaluated according to the level of security of their foundation blocks (e.g. libraries that are responsible for the algorithm implementation). Finally, to prevent failure or malfunction of ML applications, employed algorithms should have their bias reduced, should be properly evaluated to ensure that they are resilient to the environment in which they will operate and should encompass explainability strategies.

13.1.2. Networking 1

Have you developed / plan to develop national incident management or handling procedures considering AI?

13.1.3. Networking 3

Is there collaboration with the national CSIRTs/ CERTs and ISACs for the efficient handling of AI related incidents?

14. ICO

14.1. Guidance on the AI Auditing Framework - Draft guidance for consultation

Link: <https://ico.org.uk/media2/about-the-ico/consultations/2617219/guidance-on-the-ai-auditing-framework-draft-for-consultation.pdf>

Link: <https://ico.org.uk/media2/about-the-ico/consultations/2617219/guidance-on-the-ai-auditing-framework-draft-for-consultation.pdf>

14.1.1. Preventative Controls - 8

Document policy / processes for breach reporting and escalation.

14.1.2. Preventative Controls - 14

Have an API access policy in place which monitors volume and patterns of requests to identify and report suspicious activity.

15. Databricks

15.1. The Databricks AI Security Framework

Link: <https://www.databricks.com/sites/default/files/2025-02/databricks-ebook-dasf-2.pdf>

Link: <https://www.databricks.com/sites/default/files/2025-02/databricks-ebook-dasf-2.pdf>

15.1.1. DASF 1: SSO with IdP and MFA

Implementing single sign-on with an identity provider's (IdP) multi-factor authentication is critical for secure authentication. It adds an extra layer of security, ensuring that only authorized users access the Databricks Platform.

15.1.2. DASF 2: Sync users and groups

Synchronizing users and groups from your identity provider (IdP) with Databricks using the SCIM standard facilitates consistent and automated user provisioning for enhancing security.

15.1.3. DASF 3: Restrict access using IP access lists

Configure IP access lists to restrict authentication to Databricks from specific IP ranges, such as VPNs or office networks, and strengthen network security by preventing unauthorized access from untrusted locations.

Secure Egress Gateway (SEG) is a component of Databricks Platform Security that allows an administrator to implement policies that restrict access to internal or external endpoints. Currently, SEG only applies to Serverless runtimes.

15.1.4. DASF 4: Restrict access using private link

Use AWS PrivateLink, Azure Private Link or GCP Private Service Connect to create a private network route between the customer and the Databricks control plane or the control plane and the customer's compute plane environments to enhance data security by avoiding public internet exposure.

15.1.5. DASF 5: Control access to data and other objects

Implementing Unity Catalog for unified permissions management and assets simplifies access control and enhances security.

15.1.6. DASF 8: Encrypt data at rest

Databricks supports customer-managed encryption keys to strengthen data at rest protection and greater access control.

15.1.7. DASF 9: Encrypt data in transit

Databricks supports TLS 1.2+ encryption to protect customer data during transit. This applies to data transfer between the customer and the Databricks control plane and within the compute plane. Customers can also secure inter-cluster communications within the compute plane per their security requirements.

15.1.8. DASF 24: Control access to models and model assets

Organizations commonly encounter challenges in tracking and controlling access to ML models, auditing their usage, and understanding their evolution in complex machine learning workflows. Unity Catalog integrates with the MLflow Model Registry across model lifecycles. This approach simplifies the management and oversight of ML models, proving particularly valuable in environments with multiple teams and diverse projects.

15.1.9. DASF 30: Encrypt models

Databricks Platform secures model assets and their transfer with TLS 1.2+ in-transit encryption. Additionally, Unity Catalog's managed model registry provides encryption at rest for persisting models, further enhancing security.

15.1.10. DASF 31: Secure model serving endpoints

Model serving involves risks of unauthorized data access and model tampering, which can compromise the integrity and reliability of machine learning deployments. Mosaic AI Model Serving addresses these concerns by providing secure-by-default REST API endpoints for MLflow machine learning models, featuring autoscaling, high availability and low latency.

15.1.11. DASF 33: Manage credentials securely

Databricks Secrets stores your credentials and references them in notebooks, scripts, configuration properties and jobs. Integrating with heterogeneous systems requires managing a potentially large set of credentials and safely distributing them across an organization. Instead of directly entering your credentials into a notebook, use Databricks Secrets to store your credentials and reference them in notebooks and jobs to prevent credential leaks through models. Databricks secret management allows users to use and share credentials within Databricks securely. You can also choose to use a third-party secret management service, such as AWS Secrets Manager or a third-party secret manager.

15.1.12. DASF 34: Run models in multiple layers of isolation

Databricks Serverless Compute provides a secure-by-design model serving service featuring defense-in-depth controls like dedicated VMs, network segmentation, and encryption for data in transit and at rest. It adheres to the principle of least privilege for enhanced security.

15.1.13. DASF 39: Platform security — Incident Response Team

Databricks has established a formal incident response plan that outlines key elements such as roles, responsibilities, escalation paths and external communication protocols. The platform handles over 9TB of audit logs daily, aiding customer and Databricks security investigations. A dedicated security incident response team operates an internal Databricks instance, consolidating essential log sources for thorough security analysis. Databricks ensures continual operational readiness with a 24/7/365 on-call rotation. Additionally, a proactive hunting program and a specialized detection team support the incident response program and require periodic AI audits and establish protocols for incident reporting, including logs review, performance monitoring, and procedures to report and address misuse.

15.1.14. DASF 40: Platform security — internal access

Databricks personnel, by default, do not have access to customer workspaces or production environments. Access may be temporarily requested by Databricks staff for purposes such as investigating outages, security events or supporting deployments. Customers have the option to disable this access. Additionally, staff activity within these environments is recorded in customer audit logs. Accessing these areas requires multi-factor authentication, and employees must connect to the Databricks VPN.

15.1.15. DASF 43: Use access control lists

Databricks access control lists (ACLs) enable you to configure permissions for accessing and interacting with workspace objects, including folders, notebooks, experiments, models, clusters, pools, jobs, Delta Live Tables pipelines, alerts, dashboards, queries and SQL warehouses.

15.1.16. DASF 46: Store and retrieve embeddings securely

Mosaic AI Vector Search is a vector database that is built into the Databricks Data Intelligence Platform and integrated with its governance and productivity tools. A vector database is a database that is optimized to store and retrieve embeddings. Embeddings are mathematical representations of the semantic content of data, typically text or image data. Embeddings are usually generated by feature extraction models for text, image, audio or multi-modal data, and are a key component of many GenAI applications that depend on finding documents or images that are similar to each other. Examples are RAG systems, recommender systems, and image and video recognition.

Databricks implements the following security controls to protect your data:

Every customer request to Vector Search is logically isolated, authenticated and authorized

Mosaic AI Vector Search encrypts all data at rest (AES-256) and in transit (TLS 1.2+) and optionally can be encrypted with Customer Managed Keys (CMK).

15.1.17. DASF 50: Platform compliance

Develop your solutions on a platform created using some of the most rigorous security and compliance standards in the world. Get independent audit reports verifying that Databricks adheres to security controls for ISO 27001, ISO 27018, SOC 1, SOC 2, FedRAMP, HITRUST, IRAP, etc.

Use platform with established data policy with trust and safety commitments.

15.1.18. DASF 51: Share data and AI assets securely

Databricks Delta Sharing lets you share data and AI assets securely in Databricks with users outside your organization, whether those users use Databricks or not.

15.1.19. DASF 56: Restrict outbound connections from models

Egress Control enables you to control outbound connections from your Model Serving compute resources.

With this feature, you can restrict access to the internet while allowing access via Unity Catalog Connections or Private Link. Further, this feature blocks direct access to cloud storage (over the shared S3 gateway) to ensure that all data access occurs via Unity Catalog-controlled paths to reduce the risk of data exfiltration.

15.1.20. DASF 57: Use attribute-based access controls (ABAC)

Attribute-based access controls (ABAC) allow data stewards to set policies on data and AI assets using various criteria like user-defined tags, workspace details, location, identity and time. Whether it's restricting sensitive data to authorized personnel or adjusting access dynamically based on project needs, ABAC ensures security measures are applied with detailed accuracy. Implement attribute-based access controls (ABAC) to define access policies based on attributes or characteristics of the user or the resource being accessed. Use row level filters and column masking for fine-grained access controls.

15.1.21. DASF 60: Rate limit number of inference queries

Enforce request rate limits to manage traffic at the endpoint level on a per-user and per-endpoint basis, effectively controlling access levels and volume.

15.1.22. DASF 62: Implement network segmentation

Establish network and security policies to define and enforce egress rules from models and outbound network connections from your serverless compute resources.

15.1.23. DASF 67: Federate authentication

OAuth token federation is a simpler and more secure method for authenticating to Databricks, especially for automated workloads. Token federation allows applications to authenticate to Databricks using tokens from your trusted IdP, eliminating the need to store Databricks secrets like static tokens or passwords. Databricks OAuth token federation enables your users and applications to securely access Databricks AI APIs using tokens from your identity provider (IdP). Workloads authenticate to Databricks as a service principal in the Databricks account, using workload identity tokens issued by the automation environment. The Databricks SDKs and Databricks CLI automatically fetch these workload identity tokens and exchange them for Databricks OAuth tokens, which eliminates the need manage and rotate Databricks secrets.

16. ISO/IEC

16.1. DIS 27090

Link: <https://www.iso.org/obp/ui/en/#iso:std:iso-iec:27090:dis:ed-1:v1:en>

Link: <https://www.iso.org/obp/ui/en/#iso:std:iso-iec:27090:dis:ed-1:v1:en>

16.1.1. 7.6

Development environment protection

16.1.2. 7.7

Throttling model use

17. Central Bank of the UAE

17.1. Guidance Note on the Consumer Protection and Responsible Adoption and Use of Artificial Intelligence and Machine Learning by Licensed Financial Institutions in the U.A.E

Link: <https://rulebook.centralbank.ae/en/rulebook/guidance-note-consumer-protection-and-responsible-adoption-and-use-artificial-intelligence>

Link: <https://rulebook.centralbank.ae/en/rulebook/guidance-note-consumer-protection-and-responsible-adoption-and-use-artificial-intelligence>

17.1.1. 6. Continuous Monitoring and Review - c

Mechanisms should be in place to detect, report and remediate any performance issues, biases or unintended consequences that may arise over time.

18. SDAIA (Saudi Arabia)

18.1. AI Ethics Principles

Link: <https://sdaia.gov.sa/en/SDAIA/about/Documents/ai-principles.pdf>

Link: <https://sdaia.gov.sa/en/SDAIA/about/Documents/ai-principles.pdf>

18.1.1. Principle 2 – Privacy & Security - Prepare Input Data - 2

The confidentiality of data ensures that information is accessible only to those who are authorized to access the information and that there are specific controls that manage the delegation of authority.

18.2. AI Adoption Framework

Link: <https://sdaia.gov.sa/en/SDAIA/about/Files/AIAdoptionFramework.pdf>

Link: <https://sdaia.gov.sa/en/SDAIA/about/Files/AIAdoptionFramework.pdf>

18.2.1. 5.1.2 Privacy and Safety - Activating Recovery and Disaster Plans

Implementing professional recovery plans helps mitigate the impact of unexpected outages. These plans include automated backups, ready-to-launch backup sites, and automated response mechanisms that restore systems within minutes, ensuring business continuity and protecting critical data.

19. IETF

19.1. Security Requirements for AI Agents

Link: <https://www.ietf.org/archive/id/draft-ni-a2a-ai-agent-security-requirements-00.html>

Link: <https://www.ietf.org/archive/id/draft-ni-a2a-ai-agent-security-requirements-00.html>

19.1.1. 3.1. Identity Provisioning and Management

Identity provisioning and management are the process of creating and assigning a verifiable digital identity to an agent.

Initial Trust Establishment: Initial trust can be established through one or more of the following trust anchors, including, but are not limited to: a manufacturer-embedded immutable credential like an IDevID certificate; a hardware root of trust like a Trusted Platform Module (TPM) or Hardware Security Module (HSM); identity documents like an AWS Instance Identity Document or an Azure Managed Service Identity token. This step verifies the agent's execution environment (device, container, etc.) as trustworthy, allows the device or container to join the network, thereby enabling secure operations for all subsequent steps.

Credential Request: During a credential request, the agent must provide multiple proofs of its legitimacy, such as a Certificate Signing Request or a Proof of Possession by signing with the corresponding private key, as well as remote attestation by collecting and submitting evidence to a RATS (Remote Attestation Procedures) Verifier. Additionally, to define the agent's operational scope, the request should incorporate user identity context, binding the credential to a specific human user or an organizational role.

Credential Issuance: The ACA validates proofs and requests from the above two steps, if passed, it issues an agent-specific credential that may include its owner or requester identity, capabilities, locator, acceptable validation methods for the ARS.

Credential Lifecycle Management: The ACA not only issues credentials but also defines and enforces revocation policies. These policies are triggered by specific events, such as a detected security compromise, the agent's scheduled decommissioning, or a key rotation.

19.1.2. 3.2. Agent Registration

After receiving a credential from the ACA, the agent then sends it to the ARS to authenticate itself and start the registration process.

Authentication: The ARS must verify the legitimacy of the credential submitted by the agent. It must be signed or otherwise endorsed by the ACA.

Registration: The ARS then checks if the information signed by the ACA, such as the agent's capabilities, exactly matches the registration request sent by the agent. Upon successful validation, the ARS assigns the agent a unique identifier and establishes an agent record that links the identifier to its attributes.

Record Management: This step automatically removes expired credentials and synchronizes with the ACA to ensure timely revocation of credentials, preventing the use of invalid or compromised credentials.

19.1.3. 3.3. Agent Onboarding

Agent onboarding differs between campus and cloud environments. On campus, agents use protocols like EAP-TLS for network access. In the cloud, the process involves injected sidecars, which register agents to the central service mesh registry automatically to enable communication and management.

19.1.4. 4.1. Cross-Domain Identifier Interoperability

Different domains may use distinct identifier schemas. Possible methods include:

- pre-configured schema translation
- cross-domain identifier synchronization
- a universal parsing framework or system

19.1.5. 4.2. Secure Cross-Domain Transmission

Mutual TLS (mTLS) connection starts from the external requesting agent to the master agent. The master agent terminates the mTLS connection and parses the application layer requests. In this case, the master agent functions as an OAuth resource server, and manages internal task orchestration.

19.1.6. 4.3. Authenticating External Calls

The master agent then verifies the identity of the requesting agent, and whether or not it has permission to the requested service or agent. Different authentication methods might be possible:

- API keys
- Username-password
- Pre-shared secrets
- Assertions (for example, JWT Authorization Grant[I-D.draft-ietf-oauth-identity-chaining-06])

which can even be combined with AND/OR logic. During this process, the master agent might be able to identify the caller endpoint type:

- human user via browser or app
- human user via API
- AI agents
- Hardware or equipment via an IoT API

19.1.7. 4.4. IAM Integration

Since the agent may inherit its access rights from its owner or user, when authenticating requests, the validation might require integration of IAM systems for redirected verification.

19.1.8. 5.1. Authorization Handling

The master agent acts as the role of OAuth 2.1 resource server. It must validate access tokens as described in OAuth 2.1 Section 5.2. If validation fails, it must respond according to OAuth 2.1 Section 5.3 error handling requirements.

19.1.9. 5.2. Authorization Chaining Across Domains

In an agentic AI use case, a request may traverse multiple resource servers in multiple trust domains before completing. It will be common that the requesting agent from domain A needs to access the resource server (master agent) of domain B. During this process, the following information should be preserved:

Original requesting agent identity

Authorization context

- Scope
- Resource
- Audience
- Grant type
- Assertion

19.1.10. 0

Within a domain, there might exist different types of heterogeneous systems or legacy systems that require different authentication methods. They could be API endpoints, microservices, tools or databases. The exact authentication methods are determined by the service itself, for example,

- bearer tokens
- API keys
- pre-shared secrets
- username-passwords
- X.509 certificates, etc.

As a result, the master agent also works as an intermediary credential manager that converts the formats, scopes, identity of the credential, bridging the gap between heterogeneous systems and platforms.

Examples include:

Static secrets (API keys) to be exchanged to short-lived, on demand credentials (bearer tokens)

19.1.11. 5.5. Zero Trust Analysis

The above information can be used as rich context that allow zero trust access control. Remote attestation results of the requesting agent could also be part of access policy decision point's inputs. Remote attestation results of the requesting agent could include the following information:

- RoT and trust anchors
- Identifiers
- Affiliations
- Posture assessment results
- Capabilities

The overall information will be used as input of Policy Engine (PE) and Policy Decision Point (PDP).

19.1.12. 5.6. Microsegmentation

Microsegmentation may be enforced to prevent lateral movement of security risks. Possible granularity of microsegmentation includes:

- per IP segment/subnet
- per each workload
- per tags and attributes (of workload), etc.

There should be policy enforcement points (PEP) at the perimeter of each segment. Each PEP can receive software-defined security policies issued by PE/PDP.

20. NCSC/NSA/CISA etc

20.1. AI Data Security

Link: https://media.defense.gov/2025/May/22/2003720601/-1/-1/0/CSI_AI_DATA_SECURITY.PDF

Link: https://media.defense.gov/2025/May/22/2003720601/-1/-1/0/CSI_AI_DATA_SECURITY.PDF

20.1.1. 1.4 Leverage trusted infrastructure

Use a trusted computing environment that leverages Zero Trust architecture. [10] Provide secure enclaves for data processing and keep sensitive information protected and unaltered during computations. This approach fosters a secure foundation for data privacy and security in AI data workflows by isolating sensitive operations and mitigating risks of tampering. Trusted computing infrastructure supports the integrity of data processes, reduces risks associated with unverified or altered data, and ultimately creates a more robust and transparent AI ecosystem. Trusted environments are essential for AI applications where data accuracy directly impacts their decision-making processes.

20.1.2. 1.6 Encrypt Data

Adopt advanced encryption protocols proportional to the organizational data protection level. This includes securing data at rest, in transit, and during processing. AES-256 encryption is the de facto industry standard and is considered resistant to quantum computing threats. [12] [13][13] Use protocols, such as TLS with AES-256 or postquantum encryption, for data in transit. Refer to NIST SP 800-52r2, "Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations" [14] for more details.

20.1.3. 1.7 Store Data Securely

Store data in certified storage devices that enforce NIST FIPS 140-3 [15] compliance, ensuring that the cryptographic modules used to encrypt the data provide high-level security against advanced intrusion attempts. Note that Security Level 3 (defined in NIST FIPS 140-2 [16]) provides robust data protection; however, evaluate and determine the appropriate level of security based on organizational needs and risk assessments

20.1.4. 2.5 Secure Storage

After ingest, data needs to be stored in a database that adheres to the best practices for digital signatures, data integrity, and data provenance that are described in detail above. Note that an append-only cryptographically signed database should be used where feasible, but there may be a need to delete older material that is no longer relevant. Each time a data element is updated (e.g., resized, cropped, flipped, etc.) for augmentation purposes in a non-temporary fashion, then the updated data should be stored as a new entry with documented changes. The database's certificate should be verified at the time the database is accessed for a training run. If the database does not pass the certificate check, abort the training and conduct a comprehensive database audit to discover any data modifications.

21. NIST

21.1. SP 800-218A

Link: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218A.pdf>

Link: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218A.pdf>

21.1.1. PO.5.2

Secure and harden development endpoints (endpoints for software designers, developers, testers, builders, etc.) to perform development tasks using a risk-based approach.

21.1.2. PW.1.3

Where appropriate, build in support for using standardized security features and services (e.g., enabling software to integrate with existing log management, identity management, access control, and vulnerability management systems) instead of creating proprietary implementations of security features and services.

21.1.3. RV.1.3

Have a policy that addresses vulnerability disclosure and remediation, and implement the roles, responsibilities, and processes needed to support that policy.

Include AI model vulnerabilities in organization vulnerability disclosure and remediation policies.

Make users of AI models aware of their inherent limitations and how to report any cybersecurity problems that they encounter.

21.2. AI 800-1

Link: <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.800-1.ipd2.pdf>

Link: <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.800-1.ipd2.pdf>

21.2.1. Practice 3.2: Maintain security practices sufficient to prevent unauthorized access - 3

Apply appropriate protections against insider threats, such as limiting access to model weights within the organization or implementing two-party control systems.

21.2.2. Practice 4.2: Red-team safeguards - 7

Consider each level of access to a model that a threat actor might have, ranging from limited access through an API to direct access to the code and parameters that define the model, and determine the minimum level of access (if any) that allows the red team to accomplish its goal.

21.2.3. Practice 6.2: Respond to incidents of model misuse - 1

Establish clear organizational responsibilities for accountable incident response processes.

21.2.4. Practice 6.2: Respond to incidents of model misuse - 2

Plan for responses to plausible novel scenarios of misuse, such as strengthening safeguards or implementing new ones.

21.2.5. Practice 6.3: Establish misuse reporting mechanisms - 2

Establish formal processes to adjudicate concerns from the public, employees, and contractors in a timely fashion.

21.2.6. Practice 6.4: Provide safe harbors for third-party safety research - 1

Publish a clear vulnerability disclosure policy for model flaws that outlines how such flaws should be shared with the developer and the public, and how the organization will respond to reported flaws.

21.2.7. Practice 6.4: Provide safe harbors for third-party safety research - 2

Publish a clear safe harbor policy that commits to not pursuing legal action against or restricting the accounts of external safety researchers that act in good faith and comply with the vulnerability disclosure policy.

21.2.8. Practice 6.5: Create bounties for issues related to misuse risk - 1

Establish a program to incentivize researchers to find model flaws and disclose them according to the vulnerability disclosure policy established in Practice 6.4.

21.2.9. Practice 6.5: Create bounties for issues related to misuse risk - 2

Consider referring to norms and best practices of existing bug bounty programs, such as those instituted by software vendors, to guide program development.

21.3. IR 8596: Cybersecurity Framework Profile for Artificial Intelligence (Cyber AI Profile): NIST Community Profile

Link: <https://csrc.nist.gov/pubs/ir/8596/iprd>

Link: <https://csrc.nist.gov/pubs/ir/8596/iprd>

21.3.1. ID.IM-04

Incident response plans and other cybersecurity plans that affect operations are established, communicated, maintained, and improved

21.3.2. PR.AA-01

Identities and credentials for authorized users, services, and hardware are managed by the organization

21.3.3. PR.AA-02

Identities are proofed and bound to credentials based on the context of interactions

21.3.4. PR.AA-03

Users, services, and hardware are authenticated

21.3.5. PR.AA-04

Identity assertions are protected, conveyed, and verified

21.3.6. PR.AA-06

Physical access to assets is managed, monitored, and enforced commensurate with risk

21.3.7. PR.DS-04

Backups of data are created, protected, maintained, and tested

21.3.8. PR.PS-05

Installation and execution of unauthorized software are prevented

21.3.9. PR.IR-01

Networks and environments are protected from unauthorized logical access and usage

21.3.10. PR.IR-02

The organization's technology assets are protected from environmental threats

21.3.11. PR.IR-03

Mechanisms are implemented to achieve resilience requirements in normal and adverse situations

21.3.12. PR.IR-04

Adequate resource capacity to ensure availability is maintained

21.3.13. RS.MA-01

The incident response plan is executed in coordination with relevant third parties once an incident is declared

21.3.14. RS.MA-05

The criteria for initiating incident recovery are applied

21.3.15. RS.AN-08

An incident's magnitude is estimated and validated

21.3.16. RC.RP-01

The recovery portion of the incident response plan is executed once initiated from the incident response process

21.3.17. RC.RP-02

Recovery actions are selected, scoped, prioritized, and performed

21.3.18. RC.RP-03

The integrity of backups and other restoration assets is verified before using them for restoration

21.3.19. RC.RP-06

The end of incident recovery is declared based on criteria, and incident-related documentation is completed

21.4. AI 100-2e2025: Adversarial Machine Learning A Taxonomy and Terminology of Attacks and Mitigations

Link: <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2025.pdf>

Link: <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2025.pdf>

21.4.1. 2.3.2 Targeted Poisoning

Targeted poisoning attacks are notoriously challenging to defend against. Jagielski et al. [180] showed an impossibility result for subpopulation poisoning attacks. To mitigate some of the risks associated with such attacks, model developers may protect training data through traditional cybersecurity measures such as access controls, use methods for data sanitization and validation, and use mechanisms for dataset provenance and integrity attestation [267]. Ma et al. [230] proposed the use of differential privacy (DP) as a defense (which follows directly from the definition of differential privacy), but differentially private ML models may also have lower accuracy than standard models, and the trade-off between robustness and accuracy needs to be considered in each application. See Section 4.1.1 for further discussion on the trade-offs between the attributes of Trustworthy AI systems.

21.4.2. 3.3.3 Direct Prompting Attacks - Interventions during deployment (5) - Usage restrictions

Other interventions have focused on choices about how models are offered to users: for example, the efficacy of some attacks can be reduced by limiting the inference parameters that are accessible to users (e.g., temperature or logit bias), as well as the richness of the model generations returned (e.g., logit probabilities) [250]. Additionally, limiting the release of public information [252, 266] and artifacts [249] and restricting the total number of model queries available to users [251] may make attacks more challenging. These techniques may have additional drawbacks in limiting positive use cases.

22. Cyber Security Council (UAE)

22.1. National Cyber Security Policy for Artificial Intelligence

Link: https://csc.gov.ae/documents/38662/0/National+Cyber+Security+Policy+for+Artificial+Intelligence_v1.1.pdf/4e02b32e-9f62-948d-4bc8-b580d596451b?t=1766994254544

Link: https://csc.gov.ae/documents/38662/0/National+Cyber+Security+Policy+for+Artificial+Intelligence_v1.1.pdf/4e02b32e-9f62-948d-4bc8-b580d596451b?t=1766994254544

22.1.1. 2.2.6

The entity shall apply robust network security controls to AI/ML systems to ensure confidentiality, integrity, and availability, tailored to the systems' specific needs and characteristics.

22.1.2. 2.3.4

The entity shall enforce robust access controls on AI/ML training data to ensure data confidentiality, integrity, and privacy, while supporting accountability and auditability.

22.1.3. 2.4.2

The entity shall ensure the operational resilience of AI/ML systems by incorporating redundancy and failover mechanisms to maintain availability during disruptions.

22.1.4. 2.4.3

The entity shall proactively address AI/ML system through business continuity and disaster recovery planning processes and regularly testing restoration capability and continuity scenarios.

22.1.5. 2.6.2

The entity shall establish a robust and efficient incident reporting and management system for AI/ML cyber security incidents that ensures prompt response, compliance with legal and contractual obligations, and supports continuous learning and improvement.

22.1.6. 3.1.4 Change Management and Reporting - 4

Where possible, the entity should have a rollback plan in place to restore the system to its previous state in case a change introduces significant issues or risks.

22.1.7. 3.2.2 Security Configuration Management - 1

The entity should establish and enforce secure configuration baselines for all AI/ML supporting infrastructure and applications.

22.1.8. 3.2.2 Security Configuration Management - 3

The entity should establish procedures for the secure backup and recovery of AI/ML system configurations to minimize downtime and data loss in the event of a security incident or system failure.

22.1.9. 3.2.4 Vulnerability Management for AI/ML Systems - 1

The entity should implement a robust vulnerability management program to identify, assess, and remediate security vulnerabilities in AI/ML systems.

22.1.10. 3.2.4 Vulnerability Management for AI/ML Systems - 4

The entity should regularly review and update the vulnerability management program to align with changes in AI/ML technologies, threat landscape, and regulatory changes.

22.1.11. 3.2.5 Application Security - 3

The entity should implement a robust access control mechanism in AI/ML applications to limit the access to AI/ML data and functions based on the principle of least privilege.

22.1.12. 3.2.6 Network Security for AI/ML Infrastructure - 1

The entity should establish and enforce a network security policy specifically tailored for AI/ML infrastructure.

22.1.13. 3.2.6 Network Security for AI/ML Infrastructure - 2

The entity should segment high risk AI/ML systems from the rest of the network to minimize the risk of lateral movement in case of a breach.

22.1.14. 3.2.6 Network Security for AI/ML Infrastructure - 3

Network traffic to and from AI/ML systems should be securely encrypted and continuously monitored for any signs of intrusion or abnormal behavior.

22.1.15. 3.2.6 Network Security for AI/ML Infrastructure - 4

The entity should implement firewall rules and Intrusion Detection/Prevention Systems (IDS/IPS), or other security technologies specifically configured for the unique network behaviors and requirements of AI/ML systems.

22.1.16. 3.2.6 Network Security for AI/ML Infrastructure - 5

The entity should ensure that any remote access to AI/ML infrastructure is secure and controlled, utilizing secure protocols and methods such as multi-factor authentication.

22.1.17. 3.3.4 Access to Training Data Control - 1

The entity should enforce strict access control measures for AI/ML training data, ensuring that only authorized individuals and processes can access this data and only it for its intended purpose.

22.1.18. 3.3.5 AI/ML Data Protection at Rest and in Motion - 2

The entity should implement secure methods for the transfer of AI/ML data, using only approved and secure protocols based on the classification of the data.

22.1.19. 3.3.5 AI/ML Data Protection at Rest and in Motion - 3

The entity should enforce strong key management practices for encryption keys used to protect AI/ML data.

22.1.20. 3.4.2 Fail-Safe & Backup for AI/ML Systems - 2

The entity should design AI/ML systems, based on their criticality, with failover mechanisms to ensure system availability during disruptions, such as redundant infrastructure deployment in geographically diverse data centers, load balancing, and automatic failover and switching.

22.1.21. 3.4.2 Fail-Safe & Backup for AI/ML Systems - 3

The entity should employ backup measures for AI/ML systems as appropriate, including periodic and secure backup of training data, model parameters, and system configurations.

22.1.22. 3.4.2 Fail-Safe & Backup for AI/ML Systems - 4

The entity should regularly test and validate the effectiveness of fail-safe and backup measures to ensure their readiness in the event of a system failure or other disruptions.

22.1.23. 3.4.3 Continuity Planning for AI/ML Systems - 1

The entity should consider AI/ML systems in business continuity planning and disaster recovery strategies.

22.1.24. 3.4.3 Continuity Planning for AI/ML Systems - 2

The entity should establish measures to ensure the continuity of AI/ML systems, considering potential disruptions from both technical and non-technical factors, including data corruption, loss of critical resources, and sustained outages.

22.1.25. 3.4.3 Continuity Planning for AI/ML Systems - 3

In cases where AI/ML systems are critical to operations, redundant systems or alternative processes should be available and ready for immediate deployment.

22.1.26. 3.4.3 Continuity Planning for AI/ML Systems - 4

The entity should have a defined process to recover AI/ML systems from disruptions and outline key personnel, processes, resources, and critical dependencies necessary for the recovery of AI/ML systems.

22.1.27. 3.4.3 Continuity Planning for AI/ML Systems - 5

The entity should periodically test the continuity and recovery plan for AI/ML systems and train relevant staff to ensure they are adequately prepared.

22.1.28. 3.5.2 Defending Against AI/ML Attacks - 5

AI/ML systems should be designed and configured to resist, contain, and recover from attacks, minimizing potential damage and disruption.

22.1.29. 3.6.2 Incident Reporting and Management for AI/ML - 1

The entity should establish a robust incident reporting and management process for AI/ML security incidents to ensure they are appropriately identified, logged, investigated, and resolved.

22.1.30. 3.6.2 Incident Reporting and Management for AI/ML - 4

The entity should consider automated response mechanisms for AI/ML security incidents to mitigate the impact of attacks and prevent further escalation such as isolating affected systems, adjusting security controls, or blocking network traffic, based on predefined conditions and incident severity levels.

23. Microsoft

23.1. Cloud Adoption Framework - Secure AI

Link: <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/scenarios/ai/secure>

Link: <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/scenarios/ai/secure>

23.1.1. Secure AI resources 2 - Secure all AI communication channels

Exposed communication paths between AI components allow data interception and system compromise. Properly secured channels prevent unauthorized access and protect sensitive information in transit. Implement managed identities for secure authentication without stored credentials, use virtual networks to isolate AI communications, and deploy Azure API Management to secure Model Context Protocol server endpoints.

23.1.2. Secure AI data 1 - Define and maintain data boundaries

Clear data boundaries ensure AI workloads access only data appropriate for their intended audience and use case. Use Microsoft Purview to classify data sensitivity and define access policies. Implement Azure role-based access control (RBAC) to restrict data access by workload and user group. Use Azure Private Link to create network-level data isolation between AI applications.

23.1.3. Detect AI security threats 2 - Establish AI-focused incident response procedures

Undetected security incidents can lead to data loss, model compromise, or service disruption that damages business operations. Specialized incident response procedures address the unique characteristics of AI security events. Build and test incident response plans that address AI-specific threats and continuously monitor for indicators of compromise in AI systems. Establish clear escalation procedures for different types of AI security incidents.

24. Google

24.1. Secure AI Framework

Link: <https://www.saif.google/secure-ai-framework>

Link: <https://www.saif.google/secure-ai-framework>

24.1.1. Application Access Management

Ensure that only authorized users and endpoints can access specific resources for authorized actions.

24.1.2. Incident Response Management

Manage response to AI security and privacy incidents.

25. Multi Agency

25.1. Guidelines for secure AI system development

Link: <https://www.ncsc.gov.uk/files/Guidelines-for-secure-AI-system-development.pdf>

Link: <https://www.ncsc.gov.uk/files/Guidelines-for-secure-AI-system-development.pdf>

25.1.1. Secure your infrastructure

You apply good infrastructure security principles to the infrastructure used in every part of your system's life cycle. You apply appropriate access controls to your APIs, models and data, and to their training and processing pipelines, in research and development as well as deployment. This includes appropriate segregation of environments holding sensitive code or data. This will also help mitigate standard cyber security attacks which aim to steal a model or harm its performance.

25.1.2. Protect your model continuously

Attackers may be able to reconstruct the functionality of a model or the data it was trained on, by accessing a model directly (by acquiring model weights) or indirectly (by querying the model via an application or service). Attackers may also tamper with models, data or prompts during or after training, rendering the output untrustworthy.

You protect the model and data from direct and indirect access, respectively, by:

Implementing standard cyber security best practices

Implementing controls on the query interface to detect and prevent attempts to access, modify, and exfiltrate confidential information

To ensure that consuming systems can validate models, you compute and share cryptographic hashes and/or signatures of model files (for example, model weights) and datasets (including checkpoints) as soon as the model is trained. As always with cryptography, good key management is essential.

Your approach to confidentiality risk mitigation will depend considerably on the use case and the threat model. Some applications, for example those involving very sensitive data, may require theoretical guarantees that can be difficult or expensive to apply. If appropriate, privacy-enhancing technologies (such as differential privacy or homomorphic encryption) can be used to explore or assure levels of risk associated with consumers, users and attackers having access to models and outputs.

25.1.3. Develop incident management procedures

The inevitability of security incidents affecting your AI systems is reflected in your incident response, escalation and remediation plans. Your plans reflect different scenarios and are regularly reassessed as the system and wider research evolves. You store critical company digital resources in offline backups. Responders have been trained to assess and address AI-related incidents. You provide high-quality audit logs and other security features or information to customers and users at no extra charge, to enable their incident response processes.

26. Cloud Security Alliance (CSA)

26.1. AI Controls Matrix

Link: <https://cloudsecurityalliance.org/artifacts/ai-controls-matrix>

Link: <https://cloudsecurityalliance.org/artifacts/ai-controls-matrix>

26.1.1. AIS-06

Establish and implement strategies and capabilities for secure, standardized, and compliant application deployment. Automate where possible.

26.1.2. AIS-10

Define and implement processes, procedures, and technical measures to secure APIs. Review and update for any improvements at least annually or after significant system changes.

26.1.3. AIS-13

Implement sandboxing techniques to execute AI tools and plugins in isolated environments to prevent unintended interactions with critical systems or data and limit the possibility of lateral movement.

26.1.4. BCR-03

Establish strategies to reduce the impact of business disruptions, and improve resiliency and recovery from business disruptions.

26.1.5. BCR-04

Establish, document, approve, communicate, apply, evaluate and maintain a business continuity plan based on the results of the operational resilience strategies and capabilities.

26.1.6. BCR-08

Periodically perform backups. Ensure the confidentiality, integrity and availability of the backup, and verify restoration from backup for resiliency.

26.1.7. BCR-09

Establish, document, approve, communicate, apply, evaluate and maintain a disaster response plan to recover from natural and man-made disasters. Update the plan at least annually or upon significant changes.

26.1.8. BCR-10

Exercise the disaster response plan annually or upon significant changes, including, if possible, participation of local emergency authorities.

26.1.9. BCR-11

Supplement business-critical equipment with both locally redundant and geographically dispersed equipment located at a reasonable minimum distance in accordance with applicable industry standards.

26.1.10. CCC-08

Implement a procedure for the management of exceptions, including emergencies, in the change and configuration process. Align the procedure with the requirements of GRC-04: Policy Exception Process.

26.1.11. CCC-09

Define and implement a process to proactively roll back changes to a previous known good state in case of errors or security concerns.

26.1.12. CEK-01

Establish, document, approve, communicate, apply, evaluate and maintain policies and procedures for Cryptography, Encryption and Key Management. Review and update the policies and procedures at least annually or upon significant changes.

26.1.13. CEK-05

Establish a standard change management procedure, to accommodate changes from internal and external sources, for review, approval, implementation and communication of cryptographic, encryption and key management technology changes.

26.1.14. CEK-10

Generate Cryptographic keys using industry accepted cryptographic libraries specifying the algorithm strength and the random number generator used.

26.1.15. CEK-12

Rotate cryptographic keys in accordance with the calculated cryptoperiod, which includes provisions for considering the risk of information disclosure and legal and regulatory requirements.

26.1.16. CEK-13

Define, implement and evaluate processes, procedures and technical measures to revoke and remove cryptographic keys prior to the end of its established cryptoperiod, when a key is compromised, or an entity is no longer part of the organization, which include provisions for legal and regulatory requirements.

26.1.17. CEK-14

Define, implement, and evaluate processes, procedures, and technical measures to securely destroy cryptographic keys when they are no longer needed, which include provisions for legal and regulatory requirements.

26.1.18. CEK-15

Define, implement and evaluate processes, procedures and technical measures to create keys in a pre-activated state when they have been generated but not authorized for use, which include provisions for legal and regulatory requirements.

26.1.19. CEK-17

Define, implement and evaluate processes, procedures and technical measures to deactivate keys at the time of their expiration date, which include provisions for legal and regulatory requirements.

26.1.20. CEK-19

Define, implement and evaluate processes, procedures and technical measures to use compromised keys to encrypt information only in controlled circumstance, and thereafter exclusively for decrypting data and never for encrypting data, which include provisions for legal and regulatory requirements.

26.1.21. DCS-07

Design and implement physical security perimeters to safeguard personnel, data, and information systems.

26.1.22. DCS-09

Allow only authorized personnel access to secure areas, with all ingress and egress points restricted, documented, and monitored by physical access control mechanisms. Retain access control records on a periodic basis as deemed appropriate by the organization.

26.1.23. DCS-11

Train datacenter personnel to safely manage adverse events, including but not limited to unauthorized ingress and egress attempts.

26.1.24. DSP-23

Regularly validate the consistency and conformity of training, fine-tuning or augmentation data. Implement dataset versioning to ensure traceability and enforce restrictions to prevent unauthorized changes.

26.1.25. IAM-01

Establish, document, approve, communicate, implement, apply, evaluate and maintain policies and procedures for identity and access management. Review and update the policies and procedures at least annually, or upon significant changes.

26.1.26. IAM-03

Manage, store, and regularly review the inventory of identities, and monitor their level of access.

26.1.27. IAM-04

Employ the separation of duties principle when implementing information system access.

26.1.28. IAM-05

Employ the least privilege principle when implementing information system access.

26.1.29. IAM-07

De-provision or modify identity access in a timely manner.

26.1.30. IAM-09

Define, implement and evaluate processes, procedures and technical measures for the segregation of privileged access roles.

26.1.31. IAM-10

Define and implement an access process to ensure privileged access roles and rights are granted for a time limited period, and implement procedures to prevent the accumulation of segregated privileged access.

26.1.32. IAM-16

Define, implement and evaluate processes, procedures and technical measures to verify access to data and system functions is authorized.

26.1.33. IPY-03

Implement cryptographically secure and standardized network protocols for the management, import and export of data, according to industry standards.

26.1.34. I&S-01

Establish, document, approve, communicate, apply, evaluate and maintain policies and procedures for infrastructure and virtualization security. Review and update the policies and procedures at least annually, or upon significant changes.

26.1.35. I&S-03

Monitor, encrypt and restrict communications between environments to only authenticated and authorized connections, as justified by the business. Review these configurations at least annually, and support them by a documented justification of all allowed services, protocols, ports, and compensating controls.

26.1.36. I&S-04

Harden host and guest OS, hypervisor or infrastructure control plane, according to their respective best practices, and supported by technical controls, as part of a security baseline.

26.1.37. I&S-05

Separate production and non-production environments.

26.1.38. I&S-07

Use secure and encrypted communication channels when migrating servers, services, applications, or data to hosted environments. Such channels must include only up-to-date and approved protocols.

26.1.39. I&S-09

Define, implement and evaluate processes, procedures and defense-in-depth techniques for protection, detection, and timely response to network-based attacks.

26.1.40. SEF-01

Establish, document, approve, communicate, apply, evaluate and maintain policies and procedures for Security Incident Management, E-Discovery, and Forensics. Review and update the policies and procedures at least annually or upon significant changes.

26.1.41. SEF-02

Establish, document, approve, communicate, apply, evaluate and maintain policies and procedures for the timely management of security incidents. Review and update the policies and procedures at least annually, or upon significant changes.

26.1.42. SEF-09

Define incident categories and severity levels for AI systems, and determine response procedures for each, including automated response where applicable.

26.1.43. TVM-01

Establish, document, approve, communicate, apply, evaluate and maintain policies and procedures to identify, report and prioritize the remediation of vulnerabilities and threats, in order to protect systems against vulnerability exploitation. Review and update the policies and procedures at least annually or upon significant changes.

26.1.44. TVM-03

Define, implement and evaluate processes, procedures and technical measures to enable both scheduled and emergency responses to vulnerability identifications, based on the identified risk.

26.1.45. TVM-07

Define, implement and evaluate processes, procedures and technical measures based on identified risks to support scheduled and emergency responses to vulnerability identification.

26.1.46. TVM-09

Define and implement a process for tracking and reporting vulnerability identification and remediation activities that includes stakeholder notification.

26.1.47. UEM-05

Define, implement and evaluate processes, procedures and technical measures to enforce policies and controls for all endpoints permitted to access systems and/or store, transmit, or process organizational data.

26.1.48. UEM-08

Protect information from unauthorized disclosure on managed endpoint devices with storage encryption.

26.1.49. UEM-09

Configure managed endpoints with anti-malware detection and prevention technology and services.

26.1.50. UEM-10

Configure managed endpoints with properly configured software firewalls.