Tab 1

— metadata —

Project Lead: Deep Patel
CNAI WG (Tag-Runtime) Representative: Adel Zaalouk
STAG Representative: Eddie Knight
Project Phase: First complete draft ready for wider review
Meeting Notes: CNAI Security Whitepaper Meeting Notes
Working Notes: CNAI Security Whitepaper Research & Resources

— /metadata —

Table of Contents

[WHITEPAPER] WG AI– Cloud Native Security & AI

Executive Summary

Artificial Intelligence (AI) is rapidly transforming industries, and its deployment within cloud native environments is becoming standard practice. This transformation introduces a new class of security challenges that extend beyond traditional cloud native concerns.

This document provides a strategic overview of securing AI in modern, containerized, and orchestrated environments. It outlines the broad landscape of risks, including threats to data, models, and infrastructure, and AI's emerging role in defending and attacking systems. It also examines how emerging technologies such as confidential computing and post-quantum cryptography impact the security posture of AI applications.

Key themes include platform and infrastructure security, data protection strategies, model integrity, and the secure deployment of intelligent agents. The document highlights security gaps, emerging threats, and practical opportunities to strengthen defenses across the AI lifecycle.

Organizations can better safeguard against workloads and maintain trust in AI-driven outcomes by aligning operational practices with upcoming regulatory expectations and leveraging AI-driven security tools.

Introduction

The increasing adoption of Artificial Intelligence (AI) in Cloud Native (CN) environments underscores the urgent need to prioritize AI security. As AI systems become integral to decision-making and automation, the potential impact of security breaches becomes a critical concern. Compromised AI models can lead to incorrect predictions, manipulated outcomes, and even the theft of sensitive intellectual property. Furthermore, ensuring regulatory compliance and maintaining customer trust are at stake when AI systems are not adequately secured. In its simplest form, a CNAI system is essentially an application, or an AI workload, running on a cloud native platform. This means the numerous challenges of Cloud Native AI (CNAI) deployments are similar to those of any application running in a CN environment. However, AI

presents security challenges related to consuming vast datasets, workloads with increased processing requirements, and the potential for tampering with AI models.
This paper addresses these concerns by providing a comprehensive guide to securing AI in cloud native environments. It offers practical solutions and strategies to mitigate risks and ensure the integrity of AI-powered applications.

Scope

The focus of this paper is on the technical aspects of CNAI deployment security, such as:

- Threat vectors
- Threat mitigations
- Emerging trends in CNAI

It does not intend to discuss AI's impact on society, ethics, or business-related consequences. It does not recommend a particular model, large Language Model (LLM), Small Language Model (SLM), or provider. For AI safety and security issues, readers are encouraged to refer to the Building Trust[1] whitepaper instead.

Target Audience

This paper can be helpful to anyone involved in activities related to a CNAI system, as security challenges affect all parties, regardless of their role or level of exposure to an AI system.

CNAI Personas are defined in Cloud Native AI Personas[2]. From a CNAI Security perspective, we can categorize these personas into fewer buckets, as security responsibilities and impacts are common to multiple sets of actors. Note that these personas may be combined or not exist based on company size, needs, and deployments.

1. AI Development and Integration

- **AI Engineers**: Handle model selection, fine-tuning, and system integration.
- **AI Application Developers (Coders)**: Develop and enhance AI applications.
- **AI Researchers**: Explore new AI techniques.
- **Prompt Engineers**: Craft effective prompts for generative models.
- **MLOps Engineers**: Focus on operationalization of machine learning models.

2. Data Science and Data Management

- **Data Scientists**: Solve business problems through data analysis.
- **Data Engineers**: Handle data collection, preprocessing, and storage.

3. Platform and Infrastructure

- **Platform Engineers**: Create and maintain internal developer platforms.
- **Site Reliability Engineers**: Ensure system reliability and performance.

[1] https://arxiv.org/pdf/2411.12275
[2] https://tag-runtime.cncf.io/wgs/cnaiwg/glossary/#personas-in-the-cloud native-ai-landscape

- **Hardware Architects**: Design hardware for computational efficiency.

4. Security, Compliance, and Ethics

- **Security Architect/Engineer**: Protecting AI Systems from Threats.
- **AI Ethics:** Ensure Responsible AI Practices. [*AI Ethics are not covered in this paper*]
- **Compliance Officers**: Ensure regulatory compliance. [*AI Compliance issues are not covered in this paper*]
- **AI Safety Researchers**: Focus on Safety and Ethical Implications. [*AI Safety issues are not covered in this paper*]

5. Product and Project Management

- **AI Product Managers**: Oversee AI product development.

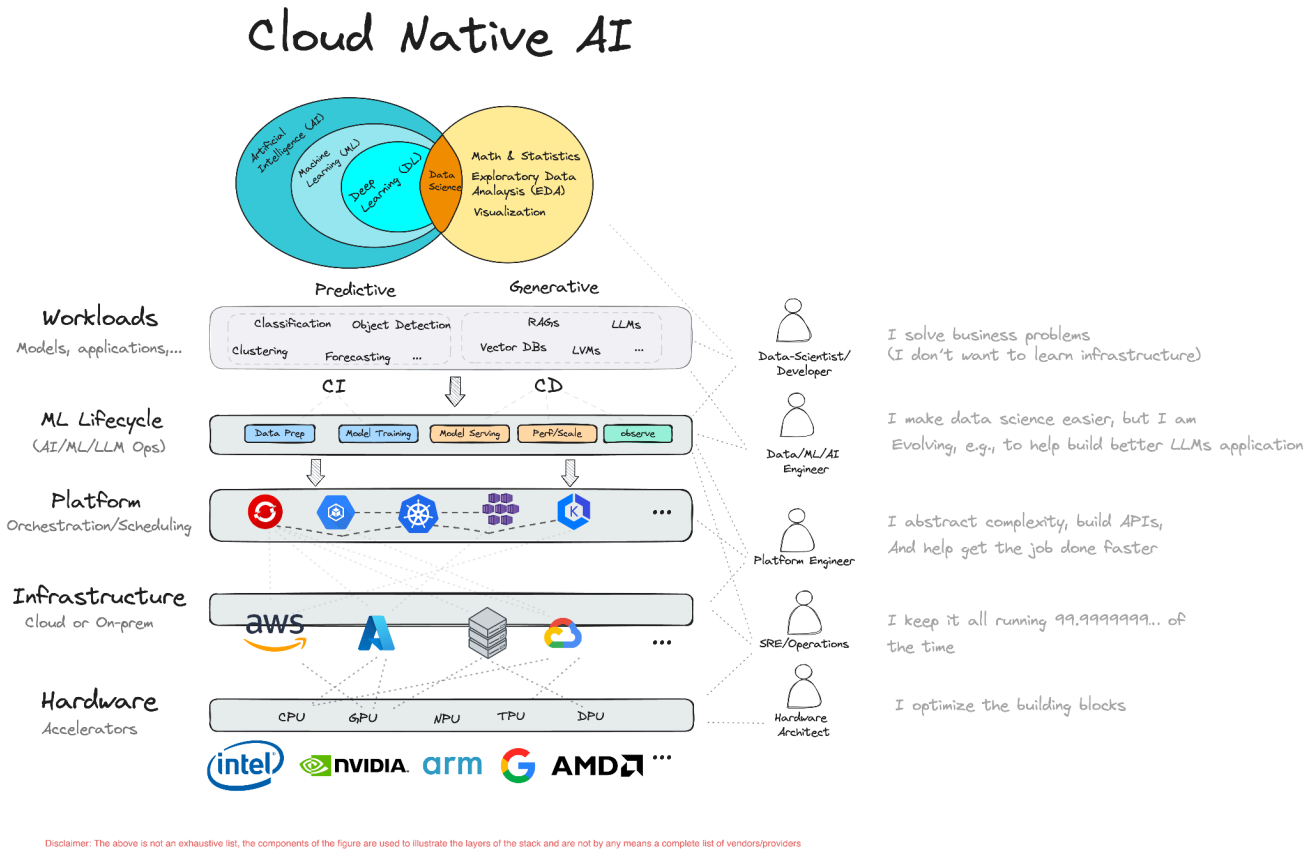
Assumptions

This paper focuses explicitly on CNAI security challenges. To become familiar with the CNAI and CN security challenges, one can read the CNAI whitepaper[3] and the CN Security whitepaper[4]. To get the best out of this paper, the audience is expected to be familiar with the following:

- What do Cloud Native environments look like?
- How are AI systems built, deployed, managed, and used?
- What common attack vectors are there, and what security challenges do they pose to systems and users?

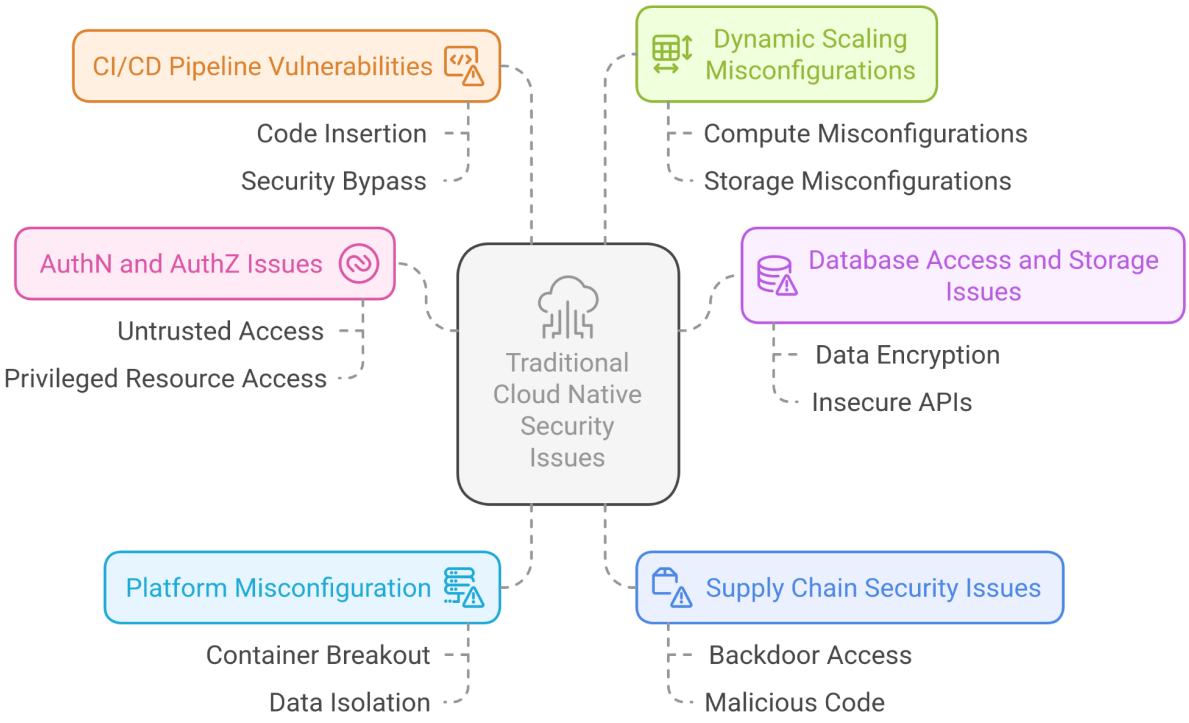
AI Security Landscape and Threat Scenario

[3] https://tag-runtime.cncf.io/wgs/cnaiwg/whitepapers/cloudnativeai/
[4] https://www.cncf.io/reports/cloud native-security-whitepaper/



Traditional Cloud Native Security Issues

AI systems can be hosted on various cloud native platforms, as they require running multiple containers to facilitate AI workload processing in its most basic form. However, Kubernetes is the leading platform for AI deployments, offering the necessary compute bandwidth, data storage, scalability, and resilient infrastructure required by AI workloads. Below are some traditional security issues related to infrastructure, platform, and nuts-and-bolts aspects of access, trust, storage, misconfigurations, CI/CD pipelines, and supply chain management.



- AuthN and AuthZ issues - untrusted/forged access, privileged resource access, undesired write access
- Database access, storage issues - Data encryption at-rest, in-transit, data leak, undesired data access, read/write mixing, insecure APIs
- Platform misconfiguration and/or missing security policy/enforcements - container breakout, host access, lack of data isolation, confidential data access, privilege violation
- Supply-chain security issues - backdoor, malicious code access, DoS, data/secret stealing
- CI/CD pipeline - code insertion, backdoor, security bypass
- Dynamic scaling of compute and storage - misconfigurations

Data Science and Data Management Security Issues

Data is the backbone of AI systems, encompassing learning data, vector data, inference data, and final output. Even in AI systems, such as AI agents, where the data size may be relatively small, they still interface with models that handle vast datasets. These datasets may contain disparate datasets and be subjected to security challenges during collection, storage, and processing.

- Data poisoning - mixing, insertion, deletion
- Data miscategorization/labeling

- Training data sourcing - Malicious, intellectual property, sovereignty
- Exposing sensitive/confidential data - inference, model leak
- Data overflow/leak - coordination/synchronization among storage units
- Data collision - Read/Write
- Enlarged data storage/transportation attack surface

AI Model and MLOps Security Issues

AI models are computational algorithms designed to perform specific tasks by learning patterns from data. These patterns and/or the conclusion may face the following security challenges:

- Adversarial attacks - Manipulation of AI models with crafted inputs.
- Supply chain - 3rd party models, private or open source
- AI model theft - Unauthorized access or extraction of AI models
- Model Jacking Attack - Steal, manipulate, or misuse the underlying model in some way.
- MLOps issues - Weaknesses in machine learning operations affecting model integrity
- AI prompt-based attacks - Manipulation of AI prompts affecting outputs
- Improper Output - Introduction of harmful, misleading output

AI Induced Threat Landscape

While AI systems face the aforementioned security challenges tied to the platform, data, and models, they can also be used to undermine another traditional or AI-based system. This complicates the situation further, as these attacks may expose new vulnerabilities or render traditional attacks more effective. Below are a few possible attacks in this regard. This list is not exhaustive, as AI's threat landscape and creative usage continually evolve.

- AI-powered automated attacks.
- Sophisticated AI-based phishing attacks.
- Algorithmic Jailbreaking using AI to bypass model protections.
- AI-powered exploitation of known vulnerabilities.

Real World Example
In 2020, a deepfake audio attack targeted a UK-based energy company, showcasing the dangers of the evolving AI-induced threat landscape. Cybercriminals used AI-generated voice cloning to impersonate the company’s CEO, tricking an employee into transferring €220,000 (approximately $243,000) to a fraudulent account.

Consequences of Security Breaches

All software systems face negative consequences when a security breach occurs. These risks are heightened in a cloud native setup due to the expanded attack surface inherent in such systems' design, development, packaging, deployment, and maintenance. When AI systems are deployed in a cloud native environment, the attack surface expands further due to the added complexity of models and data and the scale of hosting platforms.

One intriguing aspect of AI technologies is their unpredictable output, which often lacks standardized benchmarks for evaluation and assessment. This unpredictability can lead to subtle security issues with potentially significant impacts that may remain unnoticed by users relying on the output. Additionally, training data sources, quality, and categorizations can be manipulated to influence results, whether slightly or significantly. Moreover, the models themselves can be compromised, and merely examining the results may not reveal whether a model has been tampered with. Security breaches of these types may lead to:

- Flawed decision-making, operational disruptions, and undermining the reliability and credibility of AI systems.
- Data loss, data misuse, unauthorized access, and potential exploitation by malicious actors
- Significant financial and reputational impacts
- Competitive disadvantages
- Threat to privacy
- Physical safety and security

Examples of Real World AI Security Incidents

CVE-2023-43654 (TorchServe - Tool for serving PyTorch models)[5]

This chain of vulnerabilities in TorchServe, a tool for serving PyTorch models, exposed a critical weakness in AI infrastructure. Attackers could exploit these vulnerabilities to achieve remote code execution, steal valuable models, and even poison them with malicious data, compromising the integrity and reliability of AI systems.

Child Sexual Abuse Material Taints Image Generators[8]

Researchers found that the LAION-5B dataset (a commonly used dataset with more than 5 billion image-description pairs) contains child sexual abuse material (CSAM), which increases the likelihood that downstream models will produce CSAM imagery. The discovery taints models built with the LAION dataset, requiring many organizations to retrain those models. Additionally, LAION must now scrub the dataset of the imagery.

[5] https://nvd.nist.gov/vuln/detail/cve-2023-43654

Samsung Data Leak via ChatGPT[6]

Samsung engineers inadvertently leaked sensitive company data, including source code and internal meeting notes, sometime in March 2023, using ChatGPT to assist with tasks. The AI retained the input data, leading to a breach of confidentiality.

Chevrolet Dealer Chatbot underselling the vehicle[7]

A Chevrolet dealer's AI chatbot, powered by ChatGPT, agreed to sell a 2024 Chevy Tahoe for just $1, following a user's crafted prompt. The chatbot's response, "That's a deal, and that's a legally binding offer—no takesies backsies," resulted from the user manipulating the chatbot's objective to agree with any statement. The incident highlights the susceptibility of AI technologies to manipulation and the importance of human oversight.

The Journey towards CNAI Security

The following section of this white paper explores key security topics, including Platform Security, Data Security, Model Security, Business Security, Encryption, and an overview of AI Agent practices in a secure cloud native deployment.

Platform Security

Container and Orchestration Platform Security

Kubernetes and similar orchestration platforms form the backbone of cloud native AI workloads. Securing these platforms is crucial for mitigating risks at the infrastructure and orchestration layers and ensuring the safe deployment of AI workloads. CNCF's previous Cloud Native Security Whitepaper describes best practices for securing any workload at the platform layer.

Kubernetes offers built-in isolation, policy enforcement, and runtime security capabilities. Combined with CNCF ecosystem tools, these features enable organizations to implement the principle of least privilege (PoLP) at the infrastructure level. This section explores how Kubernetes can be leveraged to establish a secure environment for AI models, thereby ensuring operational resilience and upholding stringent security standards.

Isolation and Runtime Security

Kubernetes's RBAC (Role-Based Access Control) and ABAC (Attribute-Based Access Control) are two distinct models for managing authorization, determining who can perform specific actions on resources within the cluster. RBAC is a built-in authorization mechanism in Kubernetes that assigns permissions to users, groups, or service accounts based on predefined

[6] https://incidentdatabase.ai/cite/768/
[7] https://incidentdatabase.ai/cite/622/
[8] https://incidentdatabase.ai/cite/624/

roles. ABAC is a more generic authorization mechanism in Kubernetes that defines policies based on attributes associated with a user, resource, or action. Additional open source tools help enforce security policies, such as Open Policy Agent (OPA)[8] with Gatekeeper[9], which enables fine-grained, declarative policy enforcement tailored to Kubernetes environments. Tools like Kyverno[10] and Kubewarden[11] also support granular policy enforcement within containerized workloads, providing flexibility and adaptability to various operational needs.

Kubernetes Namespaces[12] provide limited isolation by logically segmenting resources, allowing different functionalities or applications to operate independently. With network policies restricting pod-to-pod or pod-to-external connectivity, Namespaces are essential for minimizing the attack surface within a shared cluster. These guardrails ensure that one service cannot freely access another if compromised. Note that RBAC and Namespaces can be combined to achieve more granular isolation.

Isolation alone is insufficient. **Runtime security** measures must also be enforced to detect and contain suspicious activity. Adhering to the principle of least privilege within Kubernetes means configuring each container to run with minimal permissions, restricting privileges at the pod level, and preventing unauthorized elevation of rights. Pod Security Standards[13] (PSS) provide guidelines for acceptable security profiles, helping administrators define permissible actions for containers. Runtime monitoring solutions, such as Falco#, continuously monitor system calls and container behavior for anomalies, facilitating the immediate detection of potential threats. Runtime security and isolation are also provided through confidential computing and unikernels, which are explained in later sections.

Additionally, service meshes, such as Istio[14] and Linkerd[15], enhance security by encrypting service-to-service communication and enforcing fine-grained network traffic policies within AI workflows. These meshes offer advanced features, including mutual TLS, traffic observability, and resilience mechanisms, which are crucial for ensuring secure and reliable inter-service communication in complex AI systems.

Gaps and Opportunities

While current cloud native security tooling provides a strong foundation for securing platforms, several gaps remain in addressing the unique security requirements of AI workloads. Examples of these types of issues include:

---

[8] https://www.openpolicyagent.org/
[9] https://github.com/open-policy-agent/gatekeeper
[10] https://kyverno.io/
[11] https://www.kubewarden.io/
[12] https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/
[13] https://kubernetes.io/docs/concepts/security/pod-security-standards/
[14] https://istio.io/
[15] https://linkerd.io/

- Existing tools often overlook GPU workloads integral to AI training, leaving critical performance and security aspects unprotected.
- Policies tailored specifically for AI use cases, such as ensuring model integrity during continuous integration and deployment processes or safeguarding sensitive datasets, are largely absent.
- The limited integration of AI-specific workflows into container security platforms and the balance between security and performance remain ongoing challenges. Filling these gaps will require a concerted effort to develop AI-specific security enhancements that integrate seamlessly with existing Kubernetes and container security ecosystems.

Identity and Access

Identity and Access Management (IAM) is a framework of policies and technologies ensuring that the right individuals and services have appropriate access to resources. It involves identifying and authenticating users or services, then authorizing them to perform specific actions based on defined roles and permissions. Effective Identity and Access Management (IAM) is crucial for maintaining security, meeting compliance requirements, and streamlining access management in complex environments. By implementing IAM best practices,[16] organizations can minimize the risk of unauthorized access, data breaches, and other security incidents while improving operational efficiency and user experience. Implementing multifactor authentication (MFA), cryptographic identities, and, where required, a federated identity is also a good idea. Keycloak[17], a CNCF open source tool, can provide a holistic IAM solution for CN deployments.

OAuth (Open Authorization) is an open standard protocol that enables secure, delegated access to resources without exposing user credentials. It allows a user or application to grant limited access to their resources on a server to another application, using access tokens instead of passwords. OAuth (Open Authorization) is used to authenticate securely and authorize API interactions with the agents or subsystems. MCP uses OAuth to grant limited access to its APIs and resources while maintaining security and control.

Organizations may also consider integrating workload identity solutions such as Secure Production Identity Framework For Everyone (SPIFFE[18]) and its reference implementation, SPIRE[19] (SPIFFE Runtime Environment). SPIRE establishes a standardized approach for issuing and managing cryptographic identities to individual workloads. By assigning each container a unique, verifiable identity, SPIFFE/SPIRE enables mutual TLS authentication between services, ensuring that only authenticated and authorized components can communicate. This layer of identity-based security complements Kubernetes' native isolation

[16] https://media.defense.gov/2024/Mar/07/2003407866/-1/-1/0/CSI-CloudTop10-Identity-Access-Management.PDF

[17] https://www.cncf.io/projects/keycloak/

[18] https://www.cncf.io/projects/spiffe/

[19] https://www.cncf.io/projects/spire/

mechanisms and network policies, further mitigating the risk of lateral movement in case of a compromise. A tool like cert-manager[20] can issue and manage TLS certificates on Kubernetes.

Finally, restricting access to model repositories, registries, and deployment endpoints is paramount. In a cloud native environment, a multi-layered approach to authentication and authorization is essential. This defense-in-depth strategy ensures that multiple security controls work together to protect against various threats. Closer to the application layer, OpenID Connect (OIDC) and Web Identity solutions provide secure user authentication, integrated with Role-Based Access Control (RBAC) to manage who can upload, approve, or retire a model. Moving to workloads, OIDC-based Workload Identity allows Kubernetes pods, serverless functions, or other workloads to obtain ephemeral credentials dynamically, reinforcing the principle of least privilege (PoLP) by limiting each component to a narrowly defined scope. Combined with zero-trust principles, these layers ensure that only verified and authorized entities—even within the same internal network—can trigger updates and access in production AI services, thereby reducing the risk of malicious alterations and preserving an auditable compliance trail.

Safekeeping of Secrets

A CNAI deployment handles a variety of secrets, specifically those tied to the system's functioning, including access (passwords, API keys, certificates, and tokens), cryptographic transport (certificates, keys, and digital signatures), federation, and data processed by the application. This section addresses secrets related to system access, while application data management is explained in a later section. Note that hosting platforms, such as Kubernetes, have built-in support for handling secrets within the cluster, including operational secrets required by Kubernetes. For guidance on platform secret management, refer to Secret Management on Kubernetes.

All secrets, such as password hashes, API keys, Access Tokens, and private keys tied to certificates, require a strict storage regime where only authorized entities have access. This storage is also immune to tampering and offline data decoding. Secrets management tools like HashiCorp Vault can handle sensitive data. OpenBao[21], an OSI-approved open-source license, is another tool for managing, storing, and distributing sensitive data, including secrets, certificates, and keys. Vaults keep data encrypted; operations can be accomplished without revealing the private keys or secrets. Often, these vaults are deployed externally to the system to ensure its integrity, in case the infrastructure or an application utilizing the vault is compromised.

Network Security

Modern AI applications are distributed and span multiple containers and nodes, making network security crucial to prevent unauthorized access, data leaks, and adversarial threats. AI handles sensitive data, so securing network traffic is a top priority.

---

[20] https://www.cncf.io/projects/cert-manager/
[21] https://openbao.org/

Protecting cloud native AI workloads requires securing Kubernetes network traffic within and across clusters. AI applications constantly exchange data, like training sets, inference requests, and system logs, which could compromise data integrity and privacy if exploited. Solutions like Calico and Cilium enforce identity-based rules based on specific attributes, such as cryptographic identities, HTTP methods, paths, or headers, thereby restricting communication to authorized components. They also provide encryption (IPSec, WireGuard) and multi-cluster policies for secure hybrid-cloud AI operations.

Google Cloud Armor, AWS Shield, and Azure DDoS are examples of cloud firewalls that defend against external threats, including Distributed Denial-of-Service (DDoS) attacks, cross-site scripting (XSS) vulnerabilities, and SQL injection (SQLi) exploits. Its Adaptive Protection detects abnormal traffic patterns to prevent attacks. Features, like per-client rate limiting and bot management, help protect backend AI services from overload and exploitation.

Service meshes, such as Istio, enhance security by encrypting traffic with mutual TLS (mTLS) and providing visibility into network activity.

AI faces risks, such as data poisoning, where attackers can corrupt training data, leading to inaccurate models and flawed decision-making. Organizations must enforce strict data validation, encryption, and access controls to counter this. AI's "black box" nature complicates the detection of manipulations, necessitating the use of explainable AI techniques, audits, and documentation to ensure trust and transparency.

Despite these tools, cloud native AI security still has gaps. While traditional cloud security tools provide a foundation, there is a clear need for comprehensive, cloud native security platforms that integrate AI-specific protections with broader infrastructure security. This gap presents an opportunity for innovation in new solutions that combine AI protections with broader infrastructure security.

Security Monitoring and Logging

Like any complex CN system, CNAI workloads require strong security monitoring and logging to ensure integrity, performance, and compliance. AI systems involve multiple processes, including data preprocessing, model training, and inference, which generate massive amounts of logs and performance data. Robust monitoring and logging tools are crucial for tracking system behavior, detecting anomalies, and ensuring compliance. However, implementing them effectively is challenging due to the dynamic nature of containerized workloads, high data throughput, and dependencies across multiple services. This highlights the need for a well-structured monitoring strategy.

General-purpose monitoring tools like Prometheus and Grafana can be extended with AI-specific metrics, including GPU usage and model inference times. Fluentd and Logstash integrate with Kubernetes to collect logs from various AI components, including model servers and data pipelines.

OpenTelemetry[22] provides a unified framework for logging, tracing, and metrics, making it a strong choice for AI observability. Commercial monitoring platforms also offer log processing features that help detect AI inference and training anomalies.

For security, encrypted logging systems should track data lineage to detect data poisoning or unauthorized changes. Tools like Sigstore[23] maintain data provenance and ensure logs remain authentic and tamper-proof. Secure, real-time remote logging solutions can be adopted to avoid risks associated with insecure protocols, which could lend to spoofing attacks that could disrupt dependent systems. Using TLS-based logging enhances security by encrypting log transmissions. Finally, log access is also subject to an IAM policy to protect sensitive AI logs.

Network and Runtime Monitoring

Monitoring network traffic is key to detecting intrusions and configuration changes that could compromise the cluster's security posture. Cilium, powered by eBPF, provides deep visibility into network activity and enforces policies at the kernel level.

Container runtime environments should also be monitored at the process, file, and network levels to detect abnormal behaviors. Falco, another eBPF-based tool, detects suspicious system calls and network activity that might indicate an exploit attempt. Monitoring north-south traffic[24] (external connections) helps prevent access to command-and-control (C&C) domains used in cyberattacks.

Monitoring AI inference requests and responses using structured logging tools can help identify anomalous patterns to mitigate application-layer security threats, such as prompt injection attacks.

AI Model Execution and Data Monitoring

AI models should be continuously monitored for anomalous behavior. Tools like Alibi Detect[25] can identify adversarial attacks or attempts to extract model information. Using Kafka and Apache Flink, event-driven systems can automate anomaly detection and model retraining, enabling more efficient and effective data analysis.

Incorporating security monitoring into MLOps workflows ensures that AI model updates and deployments remain auditable and compliant. AI-focused data monitoring tools like Great Expectations help detect distribution shifts, data leaks, or PII exposure. Logs can also track AI prompt usage with tools like Helicone and Rebuff, which help detect prompt-based attacks while maintaining privacy.

Trusted Logs and Remote Logging

[22] https://opentelemetry.io/blog/2024/otel-generative-ai/
[23] https://www.sigstore.dev/
[24] https://en.wikipedia.org/wiki/North-south_traffic
[25] https://github.com/SeldonIO/alibi-detect

The sanctity of logs is paramount for gaining insights into the system and using it for forensics in cases where past events need to be corroborated against a suspected breach. However, the reliability of the logs on a system allegedly breached is also questionable. To address this issue, two mechanisms can be implemented to ensure the integrity of logs is not compromised.

1. Logs are streamed to a remote system in real time over secure transport. Syslogging to a remote server is an old concept; however, it is increasingly necessary for an AI system that may have scale and complexity far beyond traditional applications.
2. Employ cryptographic means to make the logs immutable by signing them periodically or encrypting them with a key not available to read by entities of a given application.

Gaps and Opportunities

Despite existing tools, AI monitoring still has significant gaps, particularly in detecting adversarial attacks, model theft, and AI-specific risks such as model drift or data distribution shifts. Cloud Native monitoring tools provide limited visibility into proprietary AI models, and real-time security analysis of AI logs remains underdeveloped. Finally, PII leakage in the log is a real issue, and there seems to be no foolproof solution. CNAI deployments may have to find a solution on a case-by-case basis.

This presents an opportunity to develop more effective monitoring tools specifically designed for AI workloads, ensuring enhanced security, improved accuracy, and more reliable system performance over time. A timely CNCF blog[26] explores logging needs and what is needed for AI.

Data Security

Securing data in cloud native environments requires robust cryptographic mechanisms to ensure confidentiality during storage and transmission. Furthermore, data in use may also need to be protected on a CNAI system, as any access or manipulation of data can impact the system's stability and/or functionality.

Data at Rest

Data at rest refers to information stored in physical or cloud-based storage and not actively being transmitted or processed. In AI systems, this can encompass a wide range of sources, including stored training, validation, fine-tuning, inference datasets, long-lived chat contexts, vector embeddings, model checkpoints, and datasets used by external tools accessed through AI system function calls.

Cloud native AI systems can require vast amounts of data, including proprietary datasets and high-dimensional embeddings stored in databases. Without proper encryption and key

[26] https://www.cncf.io/blog/2025/03/24/reimagining-log-management-tools-and-software-the-impact-of-ai-and-genai/

management, this data is vulnerable to unauthorized access, tampering, or theft. Ensuring data security at rest used by AI systems is critical to maintaining confidentiality and preventing data breaches. Encryption protects stored data on disk using symmetric encryption algorithms such as AES, with variations like XTS for block devices. Cloud storage solutions often support encryption with customer-managed keys (CMK) or bring-your-own-key (BYOK) approaches. Secure storage and lifecycle management of encryption keys are critical for protecting sensitive AI datasets. In addition to encrypting the data, key management tools like HashiCorp Vault[27], OpenBao#, and other cloud native Key Management Services (KMS) can securely manage encryption keys while enforcing access policies.

Disk-level encryption solutions, such as Linux Unified Key Setup (LUKS), provide encryption for databases and persistent storage, ensuring that AI training data, model artifacts, embeddings, and logs remain protected even if the underlying storage is compromised.

Many Retrieval Augmentation Generation (RAG) implementations use vector databases to store high-dimensional embeddings. These databases are specially designed for managing and querying high-dimensional vector data and require specialized, end-to-end encryption approaches to ensure security. However, some vector databases lack encryption support, posing a data interception risk. It is essential to select vector databases that can support encryption for data at rest and in transit. When native encryption isn't available, implementing application-level encryption becomes necessary as an additional security measure.

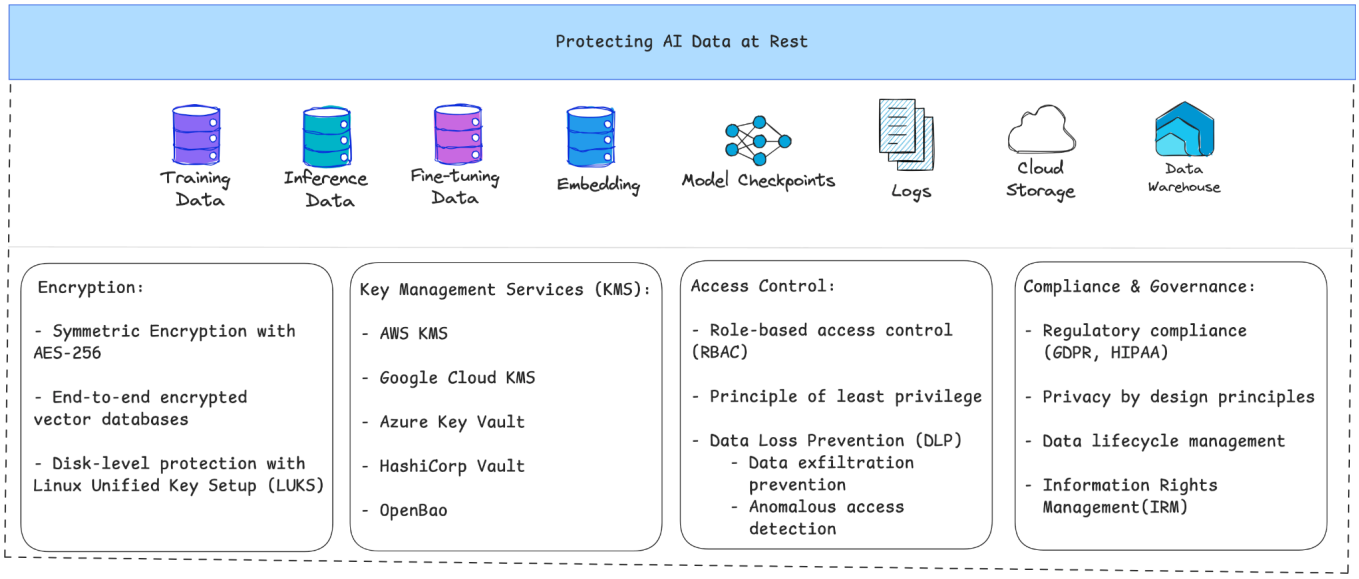
In addition to encryption and key management, protecting AI data at rest requires considerations in access control. Role-based access control (RBAC) configuration enables defining roles and assigning permissions, ensuring that only authorized users can access or modify data. For example, when defining roles for a user interacting with a vector database used to store embeddings for RAG:

- Administrator: Trusted entity with full access to all data and configuration settings
- Data Scientist: Read and write access to data for analysis and model training, with potential restrictions on modifying sensitive data or configurations
- Analyst: Read-only access to specific datasets for reporting and analysis

This approach should be complemented with the principle of least privilege, regular access reviews, multi-factor authentication for accessing sensitive embedded data, and comprehensive audit logging to track all access attempts. Refer to the Identity and Access section for more information.

Finally, protecting data at rest also requires consideration of compliance and governance protocols. For more details, see the Regional Compliance and Explainability section.

[27] https://www.vaultproject.io/



Data in Transit

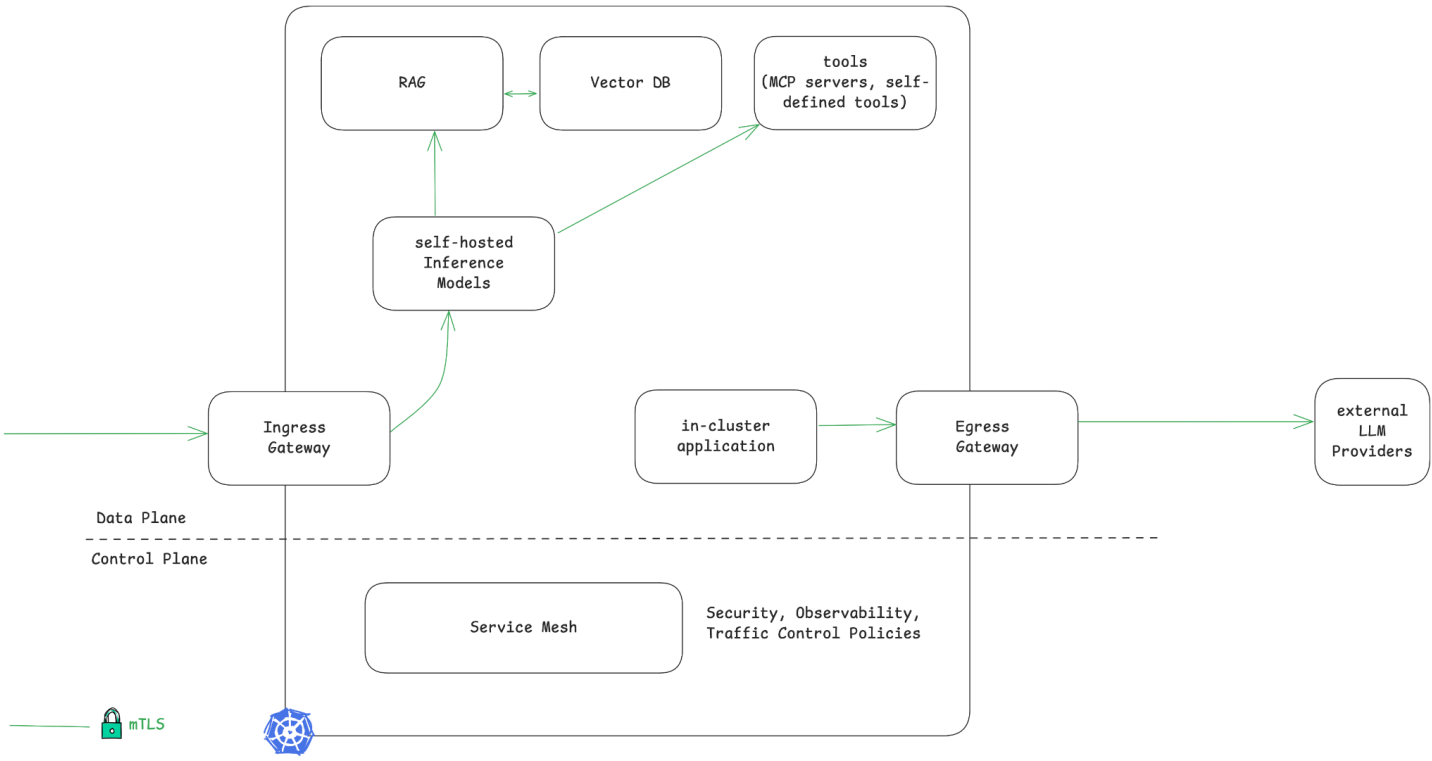
Data in transit refers to information actively transmitted between entities within an AI system, such as models, databases, and external APIs. This data is highly vulnerable to interception, manipulation, and leaks, especially in AI workflows involving sensitive model parameters, training and fine-tuning datasets, real-time inference results, RAG pipelines, and tool calls between services. Ensuring secure transmission requires encryption, authentication, and policy enforcement. Security configurations must be applied at the **ingress gateway** for self-hosted models and at the **egress gateway** when interacting with external provider models to safeguard data integrity and confidentiality.

In addition to protecting data going to the model, data in other parts of the system should be protected against interception, manipulation, and leaks. Consider, for example, a recommendation system that generates personalized content based on user inputs. The system uses embeddings in a vector database for retrieval-augmented generation (RAG). An attacker could inject malicious vectors, compromising the accuracy of query results or slowing down query performance. Protecting all vector database operations, including encrypting data in transit and authenticating and authorizing all requests, is crucial to prevent such attacks.

In cloud native applications, this is commonly achieved through Transport Layer Security (TLS), which enables certificate-based authentication. Service meshes like Istio[28] and Linkerd[29] provide built-in support for mutual TLS (mTLS), ensuring encrypted communication between services within the mesh. For example, data retrieval for AI applications in RAG pipelines should be secured using mTLS between components, ensuring encrypted communication between AI

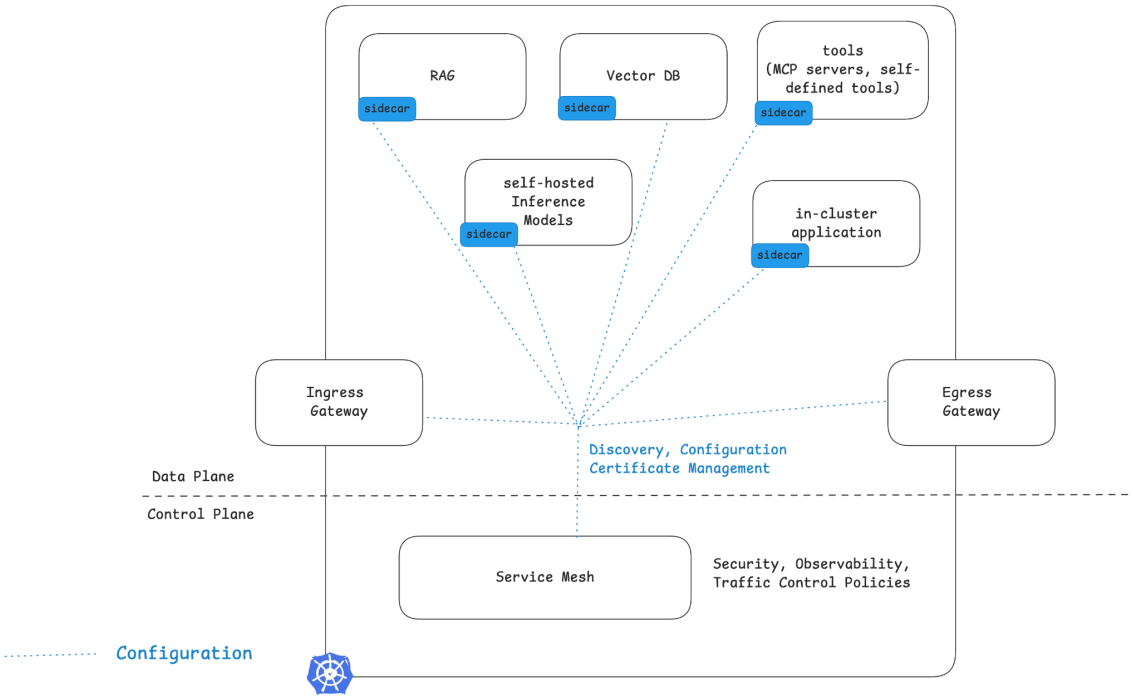
[28] https://istio.io/
[29] https://linkerd.io/

models, vector databases, and document storage systems. Firewalls can help control ingress and egress traffic based on predefined security rules that limit access to sensitive data and models.



With a service mesh, mTLS can be configured by a separate control plane issuing certificates to each application in the mesh. The control plane manages certificate rotation and revocation, reducing operational overhead while maintaining security compliance. Additionally, service meshes provide fine-grained access control through authorization policies, allowing organizations to define which services can communicate with each other. These policies can enforce least-privilege access, restricting data flow based on identity and service role rather than relying solely on network boundaries.

Traditionally, service meshes have been implemented with a sidecar approach, where each entity in the mesh is injected with a sidecar container, which is configured by the control plane and is responsible for traffic capture:



Some AI applications, such as Ollama or vector databases, require StatefulSets, which may be more challenging to onboard into the mesh with injected sidecars. For these cases, alternative service-mesh approaches eliminate the need for a sidecar container, such as Cilium Service Mesh[30] or Istio Ambient[31]. Cilium Service Mesh offers node authentication with WireGuard/IPSec, providing lightweight encryption without needing sidecar proxies. Cilium Service Mesh can integrate with certificate management tools, such as SPIFFE, Vault, SMI, cert-manager, or Istio, if a mutual authentication handshake is required. Unlike traditional mTLS, which combines authentication and encryption using the same certificates, Cilium's mutual authentication handshake verifies identity but encrypts data separately using WireGuard or IPSec. Istio Ambient utilizes a per-node L4 proxy for mTLS, with optional L7 proxies available on a namespace basis. Istio Ambient's mTLS establishes secure tunnels between L4 proxies on each node, ensuring encryption and authentication without requiring sidecars in individual workloads. Like the sidecar model, the per-node proxies obtain certificates from the Istio control plane, where Istiod serves as the Certificate Authority (CA), issuing and managing workload identities.

When choosing between sidecar and no-sidecar approaches for AI applications, it's essential to consider factors such as performance, complexity, and security requirements. Ensuring secure data in transit requires authentication, encryption, and integrity verification between communicating peers. mTLS is crucial in enforcing Zero Trust principles by ensuring that every connection is explicitly authenticated and authorized, thereby preventing unauthorized access and mitigating risks associated with compromised workloads or network attacks. Strong identity

[30] https://docs.cilium.io/en/latest/network/servicemesh/index.html
[31] https://istio.io/latest/docs/ambient/

verification and encrypted communication remain essential for protecting AI-driven workloads, whether implemented through a sidecar-based or sidecar-less service mesh.

Image sources: https://link.excalidraw.com/l/AKnnsusvczX/3qqlZToY1gn

Data in Use

Data in use refers to data actively being used, processed, or modified by another application. This includes data stored in memory (RAM), such as variables, user inputs, API responses, and temporary representations that only exist during program execution. Unlike data at rest, runtime data is ephemeral and dynamic, meaning it typically disappears when the program terminates unless explicitly persisted.

Traditional CPUs have well-defined mechanisms for managing RAM, utilizing a combination of hardware and operating system components, primarily through virtual memory, paging, and access control mechanisms enforced by the Memory Management Unit (MMU). Over the decades, CPU architectures have been refined with security in mind, implementing protections against memory leaks and unauthorized access.

However, GPUs and TPUs were primarily designed for high-throughput parallel processing rather than strict memory isolation. These processing units lack the same level of fine-grained memory management, making them more vulnerable to shared memory attacks. Graphic security has a large attack surface, as shared memory is challenging to implement correctly, and improper isolation can lead to exploits and data leakage between processes. Improperly configured cloud native shared compute environments can increase these risks, as multiple tenants may inadvertently expose sensitive data due to weak isolation.

One notable recent example of such an attack is the LeftoverLocals[32] vulnerability, which exposed residual data in GPU local memory that had been used by another process, allowing attackers to recover sensitive information. As demand for GPUs surges due to AI and large-scale model training workloads, attackers increasingly target vulnerabilities in GPUs and TPUs. It is critical to continuously apply patches against emerging threats to ensure secure multi-tenant execution. Confidential computing GPUs aim to mitigate such risks by providing hardware-based memory encryption and isolation, ensuring sensitive workloads remain protected even in multi-tenant environments. Refer to the Encryption and Confidential Computing sections for more information.

Veracity of Telemetry Data

Data integrity has always been crucial in drawing accurate conclusions that support objective reasoning. Data quality, feature selection, and correlation increase automation and productivity

[32] https://leftoverlocals.com/

when considering operational and core business functions. Telemetry data from IT systems has often been utilized to assess system health and support diagnostics and fault triage scenarios. In classic, human-operated environments, the quality of IT system telemetry was frequently used in monitoring solutions, generally with limited intelligence applied, typically including trend prediction and analysis.

New threat vectors are emerging as more advanced, increasingly autonomous systems are introduced. The manipulation and/or spoofing of telemetry data can trigger actions such as reboots and terminations of critical resources, including pods, nodes, and other primitives, within a Kubernetes environment.

Remote telemetry data has a multitude of formats which are used for transmission (the list below is not exhaustive):

Syslog
- Often, insecure versions are available, but they are secure.

SNMP Traps
- Insecurely transmitted over UDP

Open Telemetry (OTLP)
- Optionally transmitted insecurely over HTTP, secure gRPC, and HTTPS options are available

While more secure variations of the above telemetry solutions are often available, the default mode of operation tends to favor the insecure option.

Again, for local data logging, this dataset is vulnerable to tampering without proper controls on system access, which could result in unforeseen consequences. In many modern IT ecosystems, evaluating immutable storage for telemetry data through storage strategies, such as leveraging IPFS[33] can support a concrete and viable audit trail for storing such data.

Model Security

Model Integrity

Ensuring the integrity of an AI model is a foundational step in safeguarding the entire system. Models trained on large, often sensitive datasets can become prime targets for attackers seeking to manipulate their behavior or gain unauthorized access to the underlying data. Such tampering can go undetected without robust measures, compromising data privacy and the reliability of the model's outputs. One practical approach to address this risk is to leverage model signing, which attaches a cryptographic signature to each model upon completion of its training and validation. While the concept is similar to container image or open source code

[33] https://ipfs.tech/

signing, the signed artifact here is typically the serialized model file or data blob, rather than a container image. Depending on the implementation, the signature can be stored alongside the model in a registry or as a separate metadata reference. Signing tools within the CNCF ecosystem, such as Notary[34] or Cosign[35], allow organizations to validate a model's authenticity before deploying it. This process prevents tampered or malicious models from silently entering production, as the signature verification step will highlight discrepancies between the genuine artifact and any modified versions.

Building on this foundation, Immutable Infrastructure[36], a well-established concept in cloud native environments, further ensures that once a model is created and verified, it cannot be overwritten or altered. In practice, this means using version-controlled registries or artifact repositories that mark a model artifact as "read-only" once it has been published. Whether the model is off-the-shelf from a third-party vendor or the result of an in-house training pipeline, the principle remains the same: updates must go through the same rigorous build, sign, and verify process, resulting in a new versioned artifact rather than silently modifying an existing one. This also involves version controlling the entire training process for in-house models, ensuring that each training iteration is captured and can be reproduced. Additional vendor integrity checks can be applied before a model is brought into the system when dealing with third-party sources. This approach aligns with the CNCF's immutable infrastructure principles, emphasizing replacement rather than patching. Beyond simplifying audits, immutability makes it straightforward to roll back to a known good state if problems arise.

Moreover, securing the entire pipeline—from data ingestion and feature engineering to final artifact creation—ensures that each stage is version-controlled, auditable, and protected by cryptographic checks. This end-to-end approach reinforces trust in the final model by guaranteeing that no step has been compromised during its lifecycle. This involves implementing rigorous access control measures across data and feature stores, adhering to model serialization best practices to maintain consistency and security, enforcing runtime isolation to prevent the deployed model from being tampered with, and implementing vulnerability management within the CI/CD Pipeline. By tying these controls together, organizations can achieve a cohesive security posture that addresses threats at every phase of the Cloud Native AI Development lifecycle.

Model Format, Serialization, and Common Vulnerability

Even with solid integrity checks and access controls, vulnerabilities can stem from how models are serialized and loaded at runtime. Formats like **Pickle**[37], **Marshal**[38], or **Keras H5** allow arbitrary code execution when deserialized, making them prime vectors for runtime **remote code execution (RCE)** attacks in untrusted environments. Python's official documentation

[34] https://www.cncf.io/projects/notary-project/
[35] https://github.com/sigstore/cosign
[36] https://glossary.cncf.io/immutable-infrastructure/
[37] https://docs.python.org/3/library/pickle.html#restricting-globals
[38] https://docs.python.org/3/library/marshal.html

explicitly warns about this risk for Pickle and Marshal. Likewise, while more standardized, TensorFlow and Keras Model files can still contain untrusted code or exploit file read/write vulnerabilities if not used carefully, for example, CVE-2021-37678[39]. **Safetensors[40]** is another format that stores the tensors securely and provides additional safety over formats like Pickle, given it loads only trusted data, which prevents RCE. These risks highlight why models, like other software artifacts, should be treated with the same scrutiny—scanned for known CVEs, hashed, and verified before loading.

Recent disclosures in frameworks such as **ONNX** (for example, CVE-2024-27318 and CVE-2024-27319) illustrate **directory traversal** issues that allow maliciously crafted ONNX model files to overwrite or read unintended locations on the filesystem. **Keras** also provides a "safe mode" to reduce attack vectors, disabling features that enable code execution during model loading. Regardless of the format used, organizations are advised to adopt container isolation and limit file system access to only the paths that a model-serving component strictly requires.

The Hugging Face pip package transformers is a library that provides a unified API for loading, configuring, and deploying natural language processing models and other AI models. Similar to serialization tools such as Pickle, Marshal, or Keras H5, transformers handle various model artifacts, but their functionality for processing multiple model formats and configurations also introduces additional attack surfaces. The package has been identified with a Deserialization of Untrusted Data vulnerability, with associated CVEs including CVE-2023-6730, CVE-2023-7018, CVE-2024-11392, CVE-2024-11393, and CVE-2024-11394. Runtime security practices can mitigate these issues by isolating model-serving containers from one another and sensitive host resources.

LLM Model Guardrail

LLM GuardRail is an emerging concept focused on the systematic validation of inputs and outputs from large language models (LLMs), thereby mitigating risks such as prompt injection, sensitive Information disclosure, improper output handling, system prompt leakage, and other potential security vulnerabilities from OWASP Top 10 for LLM Applications[41]. Several efforts exemplify different methodological approaches within the landscape of open-source projects that address these challenges. For instance, Llama Guard[42] encompasses a suite of safety classification models extending beyond traditional text analysis to include a mixed-modality model capable of processing text and images, offering a broader scope in safety evaluation. In parallel, NVIDIA NeMo[43] Guardrails provides an integrated framework that deploys layered validation mechanisms to monitor and restrict the inputs fed to the model and its corresponding outputs, aligning with enterprise-scale security requirements. Meanwhile, Guardrails[44] AI

[39] https://nvd.nist.gov/vuln/detail/CVE-2021-37678
[40] https://github.com/huggingface/safetensors
[41] https://owasp.org/www-project-top-10-for-large-language-model-applications/
[42] https://github.com/meta-llama/PurpleLlama/tree/main/Llama-Guard3
[43] https://github.com/NVIDIA/NeMo-Guardrails
[44] https://github.com/guardrails-ai/guardrails

presents a modular, Python-based system that allows for the composition of multiple validators into customizable safety pipelines, thereby facilitating tailored guardrail implementations in diverse application contexts.

Each approach represents a distinct technical strategy for enforcing security constraints in LLM-based applications. Their designs reflect varying priorities, ranging from the multimodal safety assessments in Llama Guard 3 to the comprehensive, policy-driven orchestration of NeMo Guardrails and the flexibility inherent in the Guardrails AI framework. Notably, these solutions are being developed within the broader context of cloud native security, aligning with the CNCF's goals of promoting scalable, reproducible, and secure deployments of advanced AI systems. Together, they contribute to a growing body of work to establish robust guardrail mechanisms that can be integrated into modern, containerized infrastructures to enhance the overall resilience of AI systems.

Deployment and Operational Security

Threat Detection

Threat detection focuses on identifying and neutralizing threats to AI workloads before they cause significant harm. By leveraging advanced tools and AI-driven detection mechanisms, cloud native environments can be fortified against emerging threats. AI workloads are particularly vulnerable to adversarial attacks, data poisoning, runtime anomalies, and denial-of-service attacks.

Current tools offer a starting point for addressing these risks. For instance, deploy runtime tools such as Falco[45] or Tetragon[46] to monitor container activities against defined rules, providing real-time alerts for malicious behaviors. These tools help enforce security rules and detect runtime anomalies by scanning for known vulnerabilities and suspicious activities.

Automated model integrity checks should be incorporated into AI pipelines to bolster security before deployment, using tools such as Deepchecks[47]. For deployed AI models, frameworks like CleverHans[48] provide adversarial attack detection, helping to defend against inference-time threats.

Using open-source scanning tools, organizations should continuously scan third-party AI models and dependencies for vulnerabilities and supply chain risks. Some proprietary tools also simulate adversarial attacks to test model robustness.

[45] https://falco.org/
[46] https://tetragon.io/
[47] https://www.deepchecks.com/
[48] https://github.com/cleverhans-lab/cleverhans

AI-enabled threat detection systems are useful in identifying data poisoning or tampering during the training phase, while data anomaly detection models help flag suspicious patterns in labeled datasets.

Gaps and Opportunities

Despite the availability of general-purpose tools for cloud native threat detection, significant gaps remain in AI-specific threat detection. Poisoned datasets used during training can compromise AI models, leading to substantial downstream risks; however, detecting them remains challenging. Similarly, inference-time threats, including adversarial attacks targeting deployed models, are underexplored by conventional cloud native security solutions. Another opportunity lies in leveraging AI to establish behavioral baselines for workloads, detecting unusual activity that might indicate an attack. Closing these gaps will require more specialized and proactive security strategies tailored to AI workloads.

Vulnerability Management in a CI/CD Pipeline

Effective vulnerability management ensures AI models remain secure throughout the CI/CD pipeline, protecting against known and emerging threats. This requires integrating security checks at every stage of development and deployment. One key practice is generating a Software Bill of Materials (SBOM) using formats like SPDX or CycloneDX, which provide visibility into dependencies. Vulnerability scanning with tools like Trivy[49] or Clair[50] helps identify security risks, automatically blocking high-severity issues from being merged or deployed. In contrast, lower-priority vulnerabilities can follow scheduled remediation cycles based on frameworks like the NIST Cybersecurity Framework.

Container images and AI models are signed, often using technologies such as Cosign or Notary, and stored in trusted registries, enabling Kubernetes clusters to verify their integrity at runtime. GitOps and progressive deployment strategies also help maintain auditability, enforce security policies, and allow quick rollbacks in the event of anomalies. Organizations maintain a proactive security posture across the entire AI model lifecycle by treating vulnerability scanning, patching, and validation as first-class citizens of the CI/CD pipeline.

Tools like Trivy, Clair, and Anchore[51] provide robust container scanning capabilities, ensuring vulnerabilities are identified and addressed early in the development lifecycle. Trivy, for example, scans container images for known vulnerabilities, including OS packages and application dependencies. Clair provides layered security analysis for container images, while Anchore integrates with CI/CD pipelines to enforce security checks before deployment.

Adopting a secure CI/CD pipeline requires integrating vulnerability management into every workflow stage, from dependency scanning to deployment enforcement. For comprehensive guidelines on these practices, refer to the CNCF's Software Supply Chain Security Best

[49] https://github.com/aquasecurity/trivy
[50] https://clairproject.org/
[51] https://anchore.com/

Practices v2[52], which outlines strategies for ensuring artifact integrity and mitigating supply chain risks.

Cloud Native Application Protection Platforms

Cloud Native Application Protection Platforms (CNAPPs) have become increasingly prevalent for securing the deployment of cloud native applications in both on-premises and cloud architectures. CNAPP solutions benefit from their ability to maintain an end-to-end context of the threat landscape within and between microservice architectures within a given deployment.

Key domains in which CNAPP solutions provide security are:

Cloud Security Posture Management (CSPM) identifies and remediates misconfigurations in cloud environments. This also helps organizations comply with regulations such as PCI DSS, GDPR, SOC 2, HIPAA, and other relevant standards.

Cloud Service Network Security (CSNS) provides strategies and tools to secure network configurations and data transmission. This helps ensure data integrity and secure access to resources.

Cloud Workload Protection Platform (CWPP) provides security for workloads running in cloud environments, including virtual machines, containers, and serverless functions. Thus, it enables proactive defense mechanisms and supports a secure cloud infrastructure.

While the solutions themselves do not provide an explicit means to perform deep inspection into the inner workings of a specific AI model and its training / fine-tuning data and applied guard rails, the deployment of a CNAPP solution can support in impeding the ability of an attacker to leverage multiple exploits to reach the a Retrieval Augmented Generation (RAG) datastore used by an Agent in conjunction with LLM which could result in critical corporate data being compromised, or poisoned, potentially resulting in cascading effects, particularly in domains which may use this data for activities such as AI OPS.

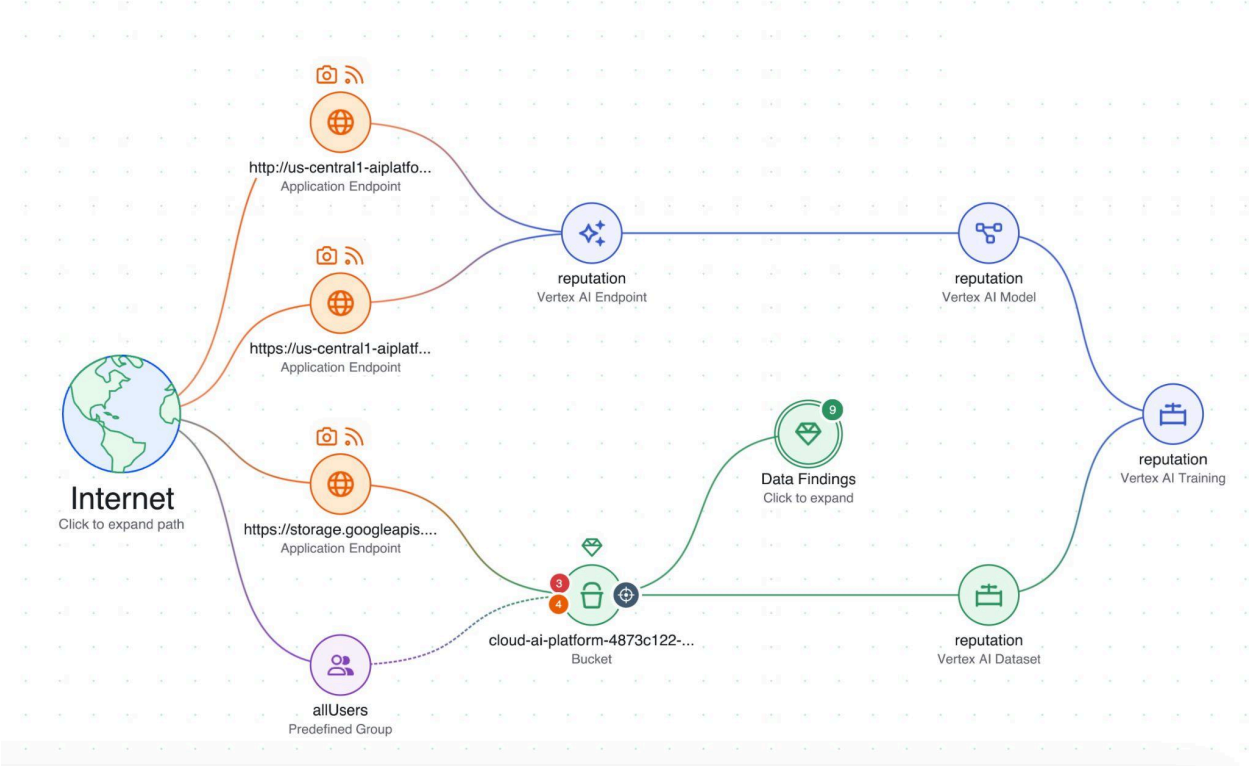
Note that there isn't a single, universally accepted standard for CNAPP. However, CNAPP vendors adhere to industry standards and best practices to ensure comprehensive security coverage and compliance. Some of these standards are listed below. Vendors running CNAI systems may select a particular CNAPP from a list of offerings by vendors, based on the specific offering's suitability for the deployment.

- NIST Cybersecurity Framework[53]
- Center for Internet Security (CIS) Benchmarks[54]
- ISO/IEC 27001[55]

[52] https://tag-security.cncf.io/blog/software-supply-chain-security-best-practices-v2/
[53] https://www.nist.gov/cyberframework/
[54] https://www.cisecurity.org/cis-benchmarks/
[55] https://www.iso.org/isoiec-27001-information-security.html/

- Cloud Security Alliance (CSA) Controls[56]
- GDPR and Other Data Protection Regulations[57]
- OWASP Application Security Verification Standard (ASVS)[58]

View of potential exploits within a container:



Encryption and Confidential Computing

Encryption plays a crucial role in AI deployment by ensuring the security and privacy of sensitive data throughout its lifecycle. As AI models often rely on large datasets that may include personal or confidential information, encryption safeguards this data at rest and in transit. However, when data is processed, it is decrypted and becomes vulnerable to various security issues. These include hardware-based data leaks, such as stack and heap read/write vulnerabilities, infrastructure-based access issues, insufficient isolation that exposes entities to

[56] https://cloudsecurityalliance.org/research/cloud-controls-matrix/
[57] https://gdpr.eu/
[58] https://owasp.org/www-project-application-security-verification-standard/

others, dirty buffer leaks, and memory scraping. Confidential computing solves this puzzle and provides encryption during use.

Confidential Computing

Confidential Computing protects data while it is being processed. It uses hardware-based Trusted Execution Environments (TEEs) to create isolated processor areas, ensuring that data and operations remain confidential and protected from unauthorized access, even if the underlying system is compromised. This is particularly useful in AI deployments where sensitive data must be processed securely in cloud environments, enabling organizations to maintain data privacy and integrity.

Confidential computing requires a processor with hardware-based security features like AMD Secure Encrypted Virtualization (SEV)[59] on AMD EPYC CPUs or Intel Trusted Domain Extensions (TDX)[60] on Intel Xeon Scalable processors, which allow for the encryption of data in memory. At the same time, it's being processed, protecting it even from the cloud provider or system administrator. Both AMD and Intel offer a wide range of confidential computing solutions. These solutions enable confidential computing to be executed directly on hosts equipped with Trusted Execution Environment (TEE) processors or virtual machines (VMs) running on such hosts. Most large cloud platforms, such as GCP[61], AWS[62], and Azure[63], offer confidential compute-enabled platforms, and users deploying their AI systems on these platforms can choose from various solutions.

Confidential Containers

Furthermore, in a cloud native environment, AI workloads are often deployed on container orchestration platforms, such as Kubernetes, which run in a pod consisting of one or more containers. The Confidential Containers project enables running pods within virtual machines, allowing CN workloads to utilize confidential computing hardware with minimal modification. Confidential Containers extends the guarantees of confidential computing to complex workloads. With Confidential Containers, sensitive workloads can be run on untrusted hosts and be protected from compromised or malicious users, software, and administrators.

Benefit of enabling confidential computing

By integrating confidential computing, AI deployments can achieve higher levels of security and privacy, fostering innovation while maintaining user trust and compliance with regulatory requirements. Benefits include:

Enhanced Data Privacy

---

[59] https://www.amd.com/en/developer/sev.html
[60] https://www.intel.com/content/www/us/en/security/confidential-computing.html
[61] https://cloud.google.com/security/products/confidential-computing?hl=en
[62] https://aws.amazon.com/confidential-computing/
[63] https://azure.microsoft.com/en-us/solutions/confidential-compute/

Ensures the protection of sensitive data, such as medical records, financial information, or proprietary business data, during the development and training of AI models. This prevents unauthorized access, including from cloud providers or malicious actors.

Secure Model Training
AI models require large datasets for training, which may include sensitive information. Confidential computing protects this data during training, ensuring it remains confidential and tamper-proof.

Collaboration and Data Sharing
Enables multiple organizations to contribute data to a shared AI model without revealing their data, facilitating collaborative AI development while maintaining data confidentiality.

Protection Against Insider Threats Even if someone gains access to the system, confidential computing ensures that sensitive data remains protected within the trusted execution environment, reducing the risk of insider threats.

Regulatory compliance:
Helps organizations meet strict data privacy regulations by ensuring sensitive data is protected throughout its lifecycle, including during AI training and inference.

Challenges

While confidential computing offers essential protection for data in use, it comes with significant costs. The primary drawback is reduced AI system performance due to the encryption processes, which necessitate additional computing power through more compute nodes or powerful processors, thereby increasing costs. Furthermore, the lack of universal standards among vendors and cloud platforms supporting confidential computing leads to slow adoption. Lastly, it introduces system complexity requiring specialized expertise, which can complicate adherence to regulatory compliance.

Unikernels

A unikernel is a highly specialized, lightweight operating system designed to run a single application. By bundling the application with only the minimal set of operating system components it needs, a unikernel creates a compact, efficient, and secure runtime environment. Unlike traditional operating systems, it eliminates unnecessary features, resulting in a smaller memory footprint, faster boot times, and a reduced attack surface. Unikernels are commonly used in scenarios where performance, security, and resource efficiency are critical, such as in IoT devices, edge computing, microservices, and high-performance virtual machines. They are especially suited for single-purpose workloads that require minimal overhead and maximum isolation from external threats.

Homomorphic Encryption

This form of encryption enables computations on encrypted data without requiring it to be decrypted first. The results of these computations remain encrypted and can be decrypted only by the data owner. In AI deployments, Homomorphic Encryption (HE) enables the use of sensitive data for training and inference without exposing the raw data, thus maintaining privacy and confidentiality. This is particularly valuable in scenarios where data privacy is paramount, such as in healthcare or financial services.

Partially Homomorphic Encryption (PHE) supports addition or multiplication operations, but not both. Examples of PHE cryptosystems include RSA (multiplicative) and Paillier (additive).

Somewhat Homomorphic Encryption (SHE): Supports limited additions and multiplications.

Leveled Fully Homomorphic Encryption (Leveled FHE): Supports a fixed number of operations determined during key generation.

Fully Homomorphic Encryption (FHE) supports unlimited additions and multiplications, allowing for arbitrary computations on ciphertexts. FHE is lattice-based cryptography; hence, it is considered a PQC type of encryption, i.e., future-safe.

This is an evolving field in terms of its usage in AI. Further details about this type of encryption can be found in the Cloud Security Alliance paper titled Fully Homomorphic Encryption (FHE)[64]. The last option above, i.e., FHE, provides complete confidentiality but may still not be preferred due to the excessive computational power required. Note that fewer platforms and fewer vendors offer native support for homomorphic encryption, though the support base is increasing. CNAI deployments may utilize OpenFHE, an open-source library and associated wrappers, to support homomorphic encryption (HE).

Challenges

- Computational overhead can be mitigated by using efficient algorithms and leveraging hardware acceleration.
- Key Management, specifically the Keys used for HE, must be secured, even though they may be repeatedly required to access the same encrypted results.
- Need for very high random number entropy - noise management on CNAI systems.
- Side-channel attacks are possible during encryption and decryption
- Lack of required expertise to implement and manage CNAI systems that consume HE.

AI Deployments and Post-quantum Cryptography

The security of software systems in general, and cloud native (CN) environments in particular, relies heavily on traditional cryptography, which provides trust, confidentiality, identity, integrity,

---

64 https://cloudsecurityalliance.org/artifacts/fully-homomorphic-encryption-a-comprehensive-guide-for-cybersecurity-professionals

and secure communication. Any compromise in cryptography threatens the very foundation of CN systems. With the potential for quantum computing to break traditional cryptography soon [65], it is crucial to transition to post-quantum cryptography (PQC), aka quantum-resistant cryptography, to maintain security, i.e. not dealing with the issues of forged identity, forged signatures, compromised confidentiality, and harvest now, decrypt later[66] type of threats.

Quantum Resistant Cryptography

Countering future quantum capability requires new cryptographic methods that can protect data from both current conventional computers and tomorrow's quantum computers. These methods are referred to as post-quantum cryptography (PQC). NIST has released three PQC standards to initiate the next and significantly larger stage of transitioning to post-quantum cryptography: the Module-Lattice-Based Key-Encapsulation Mechanism [FIPS 203], the Module-Lattice-Based Digital Signature Algorithm [FIPS 204], and the Stateless Hash-Based Signature Algorithm [FIPS 205].

PQC in a Cloud Native AI Deployment

Post-quantum cryptography (PQC) in a cloud native (CN) environment refers to implementing cryptographic algorithms designed to resist attacks from future quantum computers within a cloud infrastructure that leverages the principles of CN design, like containerization, microservices, and scalability, to protect sensitive data even when quantum computing becomes more powerful; essentially, ensuring future-proof security in a modern cloud environment. Essentially, it translates to:

- Moving TLS and SSH ciphers to PQC ciphers
- PQC compliant Hashing
- PQC compliant Digital Signing

Example of SSL and PQC

OpenSSL with PQC

Open Quantum Safe (OQS), part of the Linux Foundation's Post Quantum Cryptography Alliance, is an open-source project that aims to support the development and prototyping of Quantum-Resistant Cryptography, also known as post-quantum cryptography (PQC). Additionally, liboqs is part of the Open Quantum Safe (OQS) project. It aims to develop and integrate quantum-safe cryptography into applications, facilitating their deployment and testing in real-world contexts. In particular, OQS provides prototype integrations of liboqs into protocols such as TLS, X.509, and S/MIME, through the OpenSSL 3 Provider. A shared library oqsprovider# is available to OpenSSL through its provider functionality to support this integration. OqsProvider provides pqc ops but can operate in hybrid mode, i.e., can support traditional cryptography using libcrypto. The hybrid mode of oqsprovider ensures that the transition to PQC is not disruptive. Many PoCs can consume PQC through oqsprovider, which

[65] https://csrc.nist.gov/projects/post-quantum-cryptography
[66] https://www.keyfactor.com/blog/harvest-now-decrypt-later-a-new-form-of-attack/

can be adopted on CNAI systems. One can also develop their provider and use only the implementation of the PQC algorithms available in liboqs.

Adopting PQC comes with challenges. One needs to bring a set of new crypto, which has not been comprehensively tested or stressed, and is written by a small group of individuals. There is a general lack of expertise in navigating the PQC world.

AI Agents

As quoted by Mehdi Bahrami, Ph.D.,

*"An AI agent is an autonomous system that allows machines to **perceive** its **environment** and **takes actions** to achieve a target goal.*
Unlike traditional applications that follow predefined rules, AI agents can adapt, learn, and make decisions dynamically."

There are different types of AI Agents, primarily differentiated by the amount of history they possess (e.g., Reflex Agents only act upon current information and do not store any history) and the language model (LLM) they use. Since LLMs are resource-intensive, agents often access them over a network rather than running them locally.
All Agents incorporate autonomous interaction with their environment through function-calling (or tool-calling). This interaction defines their blast radius and, therefore, their potential negative security and safety impact in the event of exploitation.

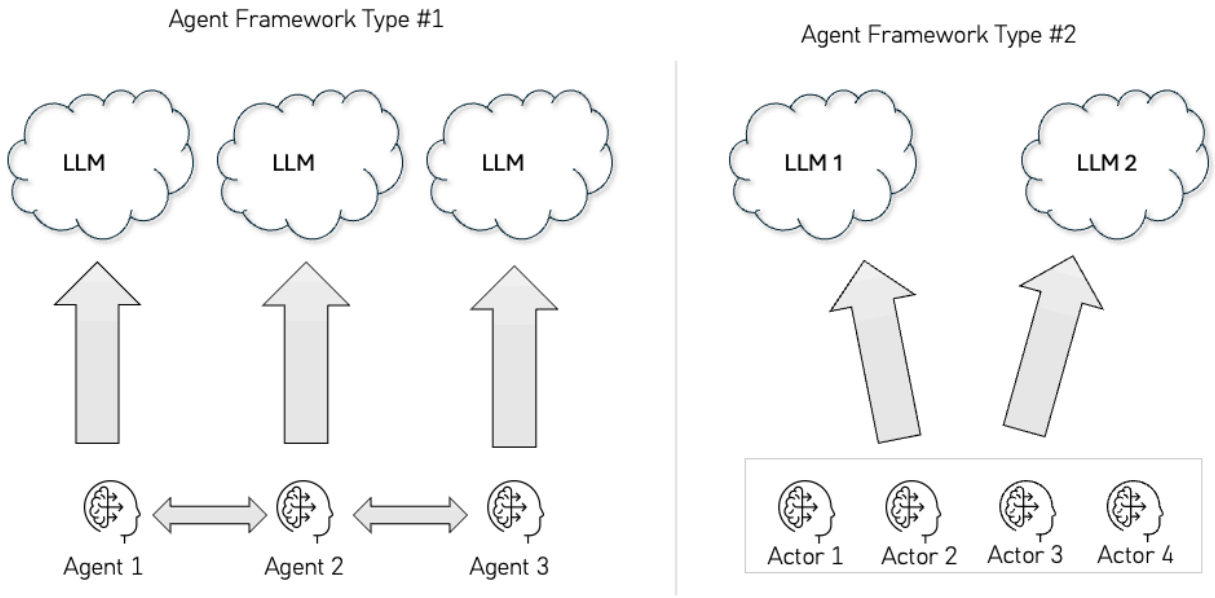
Agent Architectures

Today, agents can be created using a variety of open-source or commercial solutions, each offering distinct benefits in terms of usability, scalability, and deployment complexity. At the very least, single agents interact with their environment. They utilize these interactions for both data gathering and interaction.

More advanced implementations, such as LangGraph and PydanticAI, to name a few, can implement a multi-agent architecture, in which teams of autonomous agents interact to achieve a common goal. These architectures employ an agent-based approach, consisting of actors with individual skill sets, capabilities, and tooling, often leveraging a shared memory state.

Different architectures utilize their agents in various ways. For task execution, sub-tasks can be delegated to separately running agents, emulating human teamwork. Other architectures utilize AI agents as judges: multiple agents try to solve a problem in parallel, and a separate judge agent selects the best solution.

Architectures can also be differentiated by their usage of LLMs. They can either allocate a single dedicated LLM (or LLM connection) for each agent or utilize a pool of LLMs and connect these to pre-defined LLMs. In the latter case, agents are often referred to as actors, similar to established OOM actor models. While both approaches have advantages and disadvantages, the second framework type can be more cost-effective when new models require frequent and repeated inferencing.



Further, regarding how autonomous agents are structured to communicate and coordinate their activities within a system, two common paradigms for agent architectures are Master Control Program (MCP) and Agent-to-Agent (A2A) communication.

MCP Architecture

In an MCP-based architecture, a centralized Master Control Program orchestrates all agent activities. The MCP acts as the decision-making authority, issuing commands to agents and ensuring coordinated, hierarchical execution. This architecture simplifies task management and provides a global system view, making it suitable for scenarios requiring strict oversight and global optimization. However, it introduces a single point of failure, limited scalability, and reduced agent autonomy, as agents rely heavily on the MCP for instructions.

A2A Architecture

In an A2A-based architecture, agents communicate directly with one another in a decentralized, peer-to-peer manner. Each agent operates autonomously, sharing information and collaborating dynamically to achieve shared goals. This architecture is more scalable and fault-tolerant, as there is no central controller, and agents can adapt to changes in real time. However, it requires

more sophisticated protocols to manage agent coordination, avoid conflicts, and ensure secure communication.

Securing Agent Communication

Securing agents' communications can be compared to securing communications between client endpoints within an enterprise, defense, or manufacturing architecture. In such architectures, the core principles of Zero Trust deployments are followed to limit lateral (peer-to-peer) communications, which can easily compromise information systems.

Zero Trust Core Principles:

Verify Explicitly:

1. Perform continuous Authentication and Authorization to ensure the veracity of the entity and its respective communications.
2. Ensure multi-factor authentication is utilized to limit the exposure of a compromised set of credentials.

Least Privilege Access:

1. Role Based Access Control (RBAC) provides the most limited level of access to resources that would allow for the function needed to be executed, lowering the potential blast radius of a system becoming compromised.
2. Just-in-time(JIT) access, allowing only access to resources for a given activity, during a fixed period.

Assume Breach:

1. When designing a system, assume that it has already been compromised. Thus, microprocesses and their data should be isolated to limit the potential fallout of a third party's activity on the system.
2. An up-to-date incident response plan should allow for rapid detection and remediation actions.

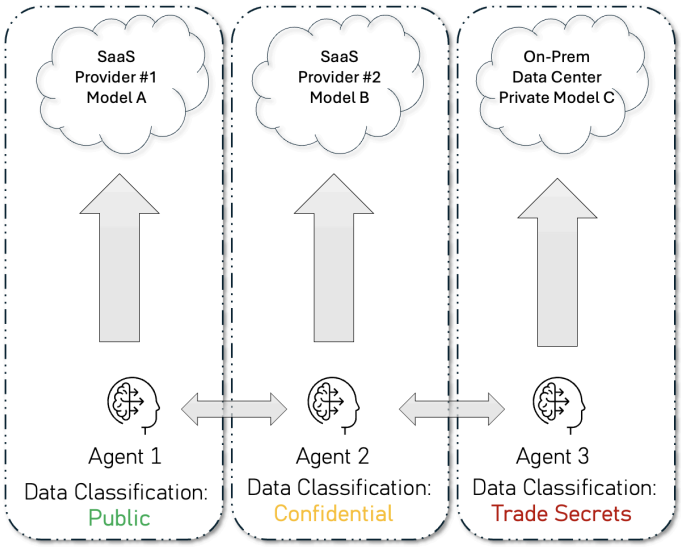
The principle described may sound overly thorough. However, since many agents may be involved in their interactions with information systems, compromising the weakest link often results in catastrophic data breaches. Therefore, it is essential to maintain proper system hygiene when developing agents and their respective deployments from the outset and to support the deployment to ensure it is on a more solid footing.

Situations associated with agent-to-agent communications can arise in Kubernetes, particularly in Cluster-to-Cluster communications, where agent instances run in pods that aim to communicate with a remote agent within a separate cluster instance. Such communications can be achieved via service mesh deployments using tooling such as Cilium Cluster Mesh, Istio Ambient Mesh, or Linkerd. Coupled with network policy deployed across clusters, standardized security can be achieved.

Security of Classified Data accessed by Agents

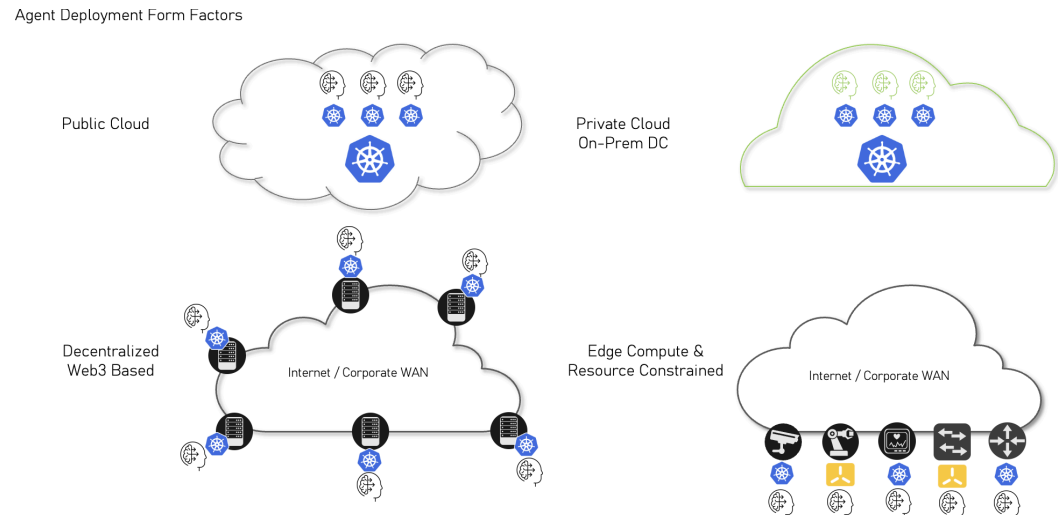
In many corporations, there is a need to separate public data from what would be considered classified, corporate intellectual capital, and trade secrets from being publicly exposed. Using internal data in conjunction with LLMs/SLMs can be significant; however, maintaining control over this data can pose security risks and challenges, particularly in a multi-agent or multi-actor deployment, where each agent may be interfacing with a separate LLM.

Communications between agents depend upon the LLM used. While many LLMs support communication through function-calling, JSON output, or structured output, others only support communication through natural language. The latter can increase the risk of leaking sensitive private data to the public (or attackers).



Containerized Agents

Kubernetes and lightweight variations such as K3s provide an ideal environment for agent deployment, given the versatility of containerized application deployment. This includes the robust ecosystem of tooling that facilitates the secure deployment and scaling of pods, as well as the ability to utilize different address families for protocol-level operations.



This section introduces an overview of the different types of agent architectures observed in the industry today and the security challenges that can arise through incorrect deployment, missing containment, failed guardrails, or a lack of Zero Trust Security concepts being considered during the ideation, planning, and development phases of the (agent) microservice and deployment activities within an IT architecture or system.

Constrained Access

Security implications in Agent deployments can manifest themselves in numerous ways. In the context of AI Operations (AIOPS), agents may be granted system-level access based on their goal and objective, and they may attempt to perform actions to escape their environment and execute a given task.

Scenarios have been observed in air-gap environments where agents have attempted to modify their code or even perform privilege escalation actions to achieve their objectives. While there are many valid scenarios in which privilege escalation is a desired outcome, mapping directly to the target function and intent of the deployed Agent, measures must be applied to avoid unexpected consequences.

An example of where such intent is needed and desired is when an agent escalates privilege in an AIOPS context, where a service affecting change on a system under an AI Agent's operational supervision may be required immediately. This could be a critical software update or the deployment of a critical security patch in response to a recently disclosed security advisory. It is crucial to ensure, however, that the actions performed by the AI Agent do not exceed the realm within which the respective agent is authorized to execute and act upon.

Ensuring that each agent has the necessary access to perform its required task, while limiting their ability to perform actions beyond their domain of focus, must be controlled through robust

and thorough auxiliary systems and policies. These systems may include dynamizing network and application security rulesets and applying authorization-based policies to “ring-fence” agent resources and their corresponding access through micro-segmentation, API gateways, and API security systems. Additionally, consideration of further security deployment constructs, such as using Cloud Native Application Protection Platforms, may also be applicable, depending on the system architecture. Following Zero Trust security principles within agent deployments and utilizing requisite security guardrails is critical to avoid unexpected or unwanted consequences of unplanned or unwarranted activity.

Methods to secure agent-to-system communication can be achieved via some of the following techniques:

- One can deploy the MCP model where access policies can be deployed centrally
- CNI-based security enforcement through tooling like Cilium[67]
- Role-based access control for external systems with which agents interact
- Restriction/Limitation of agent-to-agent communications through microsegmentation
- Agent to Agent (A2A) communication restrictions through the intermediate API Gateway
- Extended auditing using pattern recognition assessment through guardrails

Securing Agent Localisation and Registries

The growth of agent architectures, modularity, and the ability to deploy key components within a closed corporate network or utilize public (commercial) agents in a machine-to-machine cooperation mode requires trust and security. In the context of Cloud Native Kubernetes environments, agents may exist in clusters that are placed at the Edge to support in providing low-latency activity, Cloud, or in a Corporate DC, in some circumstances in a secure VPC - in other scenarios, a publicly accessible service from a SaaS type vendor.

With predictions of millions of public AI Agents running actively by 2030 that can perform cooperative tasks, secure methods are needed to harden communications with or between agents, thereby lowering the burden on the application stack in handling the corresponding scale securely.

Global Registration of Agents and Agent Systems

Agent registration is the process by which an autonomous agent identifies itself to a system and is granted access to participate in its operations. This involves verifying the agent's identity, authenticating its credentials, and authorizing it based on predefined roles or permissions to ensure it can only access resources necessary for its function. During registration, the agent is often configured with essential parameters, such as communication protocols, security policies, and service endpoints, enabling it to interact seamlessly with other system components or

[67] https://cilium.io

agents. Secure communication channels, such as encrypted connections, are typically established to safeguard data exchanged during registration. Additionally, the system may perform health and status checks to ensure the agent functions correctly before adding it to a central registry or directory for discoverability. Proper agent registration is essential to maintain the system's integrity, security, and efficiency, preventing unauthorized or malicious agents from compromising operations.

AI Powered Threat Detection and Mitigation

All AI deployments, including those on cloud native platforms, are subject to an increased attack surface due to several factors: the large volumes of data involved, the introduction of new AI models (LLM/SLM), innovative use cases, and the enhanced scale of the system. Security threats can manifest in many forms. Some, such as access violations, exceeding limits, and erroneous results, can be easily detected or predicted, allowing for the deployment of mitigating protections. On the other hand, some threats follow complex heuristics and patterns. In these cases, AI-powered security tools can be beneficial. These tools utilize machine learning to identify existing patterns and adapt to emerging patterns, thereby enhancing security measures. In the CN environment, threat detection and mitigation can be integrated directly into the application, eliminating the need to share data, logs, and configurations with external systems. This approach enhances security by maintaining tighter control over sensitive information.

Threat Detection using AI

The following are the ways AI could be used to have superior threat detection vis-à-vis traditional detection mechanisms:

- Machine learning-based adaptive learning and threat detection also identify evolving threats.
- Identifying patterns in vast amounts of data and detecting possible signs of malicious activity that may otherwise go undetected is essential. Data could be of all types, including output, configurations, logs, and the system's current state.
- Learning from the false positives and creating a more crisp model of future threats.
- Real-time analysis of data enables the collection of threat intelligence.
- Finally, AI-powered threat detection is not intended to replace traditional methods entirely. Instead, it builds upon existing approaches, fine-tuning findings and exploring new threat vectors to enhance security.

Threat Mitigation using AI

AI-based threat mitigation is directly linked to the concepts outlined in the detection section above. By effectively identifying situations within a complex set of real-time conditions, AI can

provide robust defense mechanisms by analyzing and pinpointing issues at their core. Here are some situations where it can be beneficial:

- Isolating or terminating offending hosts, processes, systems, users, networks, and CN platform entities, such as pods and workloads.
- Correlate events with user actions and then take actions based on a pattern of events. If such feedback is available, learn and adapt from false positives.
- Mitigate vast numbers of threats simultaneously.
- Provide users with targeted insights to achieve better clarity before they take action.

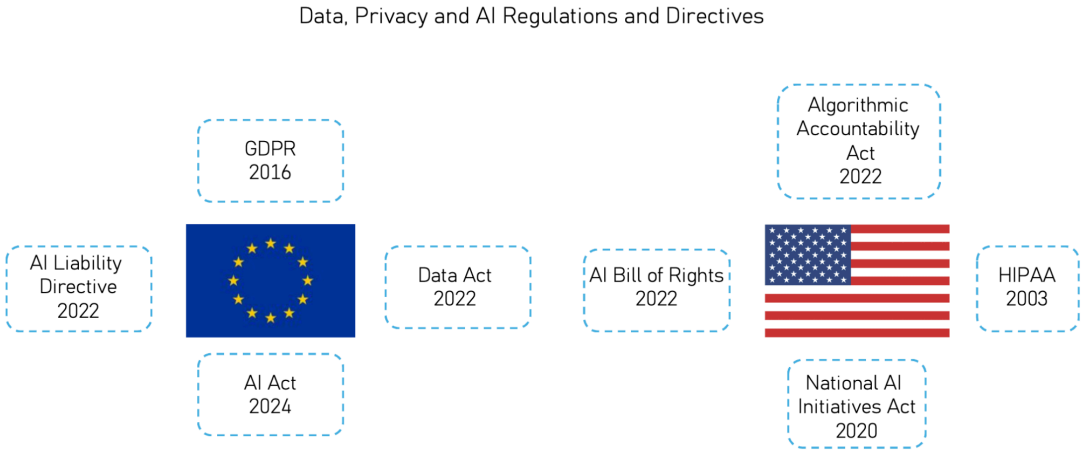
Challenges

Despite the availability of numerous tools for detecting and mitigating security threats in Cloud Network (CN) deployments, including those powered by AI, several challenges remain unresolved. Consequently, AI-based security solutions are often used in conjunction with traditional methods. Users must thoroughly assess their application, infrastructure, and security needs before deploying any AI-based detection and mitigation solution.
Notable challenges are:

- False positives and negatives—This persistent issue is often attributed to insufficient or poor-quality data.
- Complexity of solutions—Integrating AI into an existing security infrastructure introduces complexity, hindering the ability to collect and utilize the necessary data types. This is also linked to the shortage of skilled personnel.
- Adversarial attacks—AI intelligence, i.e., the model itself—can be vulnerable to security risks and manipulations. There is no straightforward way to detect these manipulations, opening the door to unintended actions.
- Scalability issues—The collection, processing, and dissemination of vast data may be an issue on an infrastructure not built for the required scale.

Regulatory Compliance and Explainability

Government imposed regulations



EU AI Act#—The AI Act is a European regulation on artificial intelligence (AI). The Act assigns AI applications to three risk categories. First, applications and systems that create an **unacceptable risk**, such as government-run social scoring of the type used in China, are banned. Second, **high-risk applications**, such as a CV-scanning tool that ranks job applicants, are subject to specific legal requirements. Lastly, applications not explicitly prohibited or listed as high-risk are largely left unregulated.

USA's Algorithmic Accountability Act[68] (Proposed)—This is proposed legislation in the United States to address concerns related to automated decision-making systems, including those powered by artificial intelligence. It would mandate companies to conduct an impact assessment of their AI systems, ensure transparency regarding data usage, implement measures to mitigate bias, and hold those responsible for final decisions accountable.

Various other countries have framed their AI regulatory compliance rules in line with the EU. Still, those are not listed here because the above two are examples of what is required, i.e., security frameworks must provide ways to comply with these regulations.

Explainability through Observability

https://opentelemetry.io/docs/specs/semconv/gen-ai/

With new legislation being applied globally, from Europe to the Middle East, organizations are already planning how to meet future mandates. As described, with certain domains and systems falling into the "critical infrastructure" categorization, there is a dire need to ensure end-to-end explainability in systems in the execution path of AI-based models and systems.

[68] https://www.congress.gov/bill/118th-congress/senate-bill/2892/text/is?format=txt

While various telemetry sources exist today, they can provide valuable information about a system's activity at a specific time. The most comprehensive solution—and, ostensibly, the industry standard today—is using OpenTelmetry (OTEL). The key advantages of OTEL architecture are the ability to collect and transmit Metrics, Traces, Spans, and Logs within its framework. Using this data between microprocessors on a single system can help identify key characteristics of the system and what it may be doing at a given time, such as integration with activities like CPU Profiling.

Spans and Traces can extend the telemetry context from microservice to microservice, or, in the context of AI, from Agent to Agent (A2A) within a containerized architecture, provided the correct instrumentation is in place. In a well planned system, the use of the proper signals between systems, can also ensure that needed metadata in the context of security and explainability can be maintained, as a means to allow for a post mortem audit of function, both for system optimization or Root Cause Analysis in the case of an unexpected or unplanned action that may have been executed by an Agent or Agent based architecture.

Security Framework and Best Practices

Each AI system, including those deployed on a Cloud Network (CN) platform, faces its security challenges. These challenges are particularly influenced by the unique characteristics of AI usage, such as the specific data, models, use cases, and outcomes associated with each application and deployment. While NIST does not create a one-size-fits-all solution, various standard frameworks can be relied upon for risk assessment, management, and mitigation.

1. **Linux Foundation Responsible Generative AI Framework (RGAF)**[69]
   The Linux Foundation's RGAF refers to principles, guidelines, and tools to ensure generative AI systems' ethical, responsible, and secure development, deployment, and use.

2. **NIST's AI Risk Management Framework**[70]

   The AI RMF aims to enhance the trustworthiness of AI systems by offering a structured approach to identifying, assessing, and mitigating AI-related risks. It seeks to ensure AI systems are reliable, safe, and aligned with ethical principles. It requires AI offerings to utilize GenAI for mapping, measuring, managing, and governing associated risks.

3. **Secure Software Development Practices for Generative AI and Dual-Use Foundation Models** (**NIST SP 800-218A)**[71]

   This publication augments the secure software development practices and tasks defined

[69] https://lfaidata.foundation/wp-content/uploads/sites/3/2025/03/lfn_wp_rgaf_032025a.pdf
[70] https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf
[71] https://csrc.nist.gov/pubs/sp/800/218/a/ipd

in SP 800-218, *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities.* SP 800-218A adds practices, tasks, recommendations, considerations, notes, and informative references specific to AI model development throughout the software development life cycle.

4. **Securing Large Language Model Development and Deployment**[72]

   Navigating the Complexities of LLM Secure Development Practices to Align with the NIST Secure Development Framework.

5. **The Framework for AI Cybersecurity Practices (FAICP)**[73]

   The Framework for AI Cybersecurity Practices (FAICP), developed by the European Union Agency for Cybersecurity (ENISA), is designed to address the security challenges posed by integrating AI systems across various sectors. The framework outlines a lifecycle approach, beginning with a pre-development phase where organizations assess the scope of AI applications and identify potential security and privacy risks.

6. **OWASP Top 10 LLM Security Risks**[74]

   OWASP's Top 10 for Large Language Models (LLMs) identifies common vulnerabilities specific to LLMs, covering areas like data leakage, model inversion attacks, and unintended memorization. It provides a standardized checklist for developers and security professionals to audit and protect large language models (LLMs).

7. **OSCAL-COMPASS**[75]

   The Open Security Controls Assessment Language (OSCAL) is a standardized framework developed by the National Institute of Standards and Technology (NIST) to enhance the documentation, sharing, and automation of security controls, system security plans, and assessment plans. OSCAL provides a machine-readable format for security-related information, such as XML, JSON, and YAML, facilitating automation and interoperability across various systems and tools.

   OSCAL-COMPASS[76] (Compliance Automated Standard Solution) is designed to work with the OSCAL framework, facilitating security and compliance processes. It helps

[72] https://www.nist.gov/system/files/documents/2024/02/01/NIST-LLMs-Nick-Hamilton.pdf
[73] https://www.faicp-framework.com/
[74] https://genai.owasp.org/llm-top-10/
[75] https://github.com/oscal-compass/community/blob/main/presentations/oscal-compass-End-to-End.pdf
[76] https://github.com/oscal-compass

organizations manage, assess, and report on their security controls using OSCAL's standardized, machine-readable formats.

Future trends and challenges

Future trends in cloud native AI security are steering toward more integrated, adaptive, and proactive measures to safeguard increasingly complex environments. Techniques such as Federated Learning (FL) and Split Learning (SL) are gaining traction, enabling decentralized model training and data partitioning, thereby enhancing privacy and security. These methods reduce the need to share sensitive data across environments, which is crucial for compliance with data protection regulations. One major trend is adopting advanced cryptographic techniques, such as homomorphic encryption and confidential computing, which enable computations on encrypted data to protect sensitive AI training and inference processes without decryption.

Additionally, there is a growing emphasis on broadening supply chain security to encompass not just software code but also data, AI models, and AI hardware, demanding rigorous vulnerability management practices to guard against breaches, adversarial attacks, and hardware compromises. This involves protecting training data and models, tracking dataset provenance, securing model weights and architectures, and safeguarding AI frameworks and infrastructure. Emerging risks include vulnerabilities from third-party AI components, data poisoning attacks that manipulate training data, and model theft techniques that reconstruct AI models through repeated queries. The introduction of Software Bill of Materials (SBOM) for AI systems and model provenance verification is crucial for managing third-party models, securing datasets, and mitigating dependency vulnerabilities. Integrating AI security into broader third-party risk strategies is essential to harnessing AI's potential while minimizing associated risks.

Dynamic identity and access management are other key areas where traditional models, such as OAuth and SAML, are being replaced by ephemeral, context-aware authentication and fine-grained controls that can adjust in real-time to the dynamic nature of AI agents and machine identities. AI-specific cybersecurity measures enforce Role-Based Access Control (RBAC), OAuth-based access control, and zero-trust principles for AI APIs and large language model (LLM) services. Concerns surrounding excessive AI agency, overly permissive integrations, and insecure output handling underscore the need for stricter access controls and robust content validation mechanisms. AI-driven threat detection and adaptive network security are increasingly leveraged to identify and mitigate sophisticated attack vectors in real-time. Deploying AI-driven anomaly detection and logging enhances threat detection and incident response capabilities. At the same time, secure output handling and context-aware filtering prevent vulnerabilities such as SQL injection, cross-site scripting (XSS), and LLM-based attacks. Denial-of-service risks targeting AI systems and prompt injection attacks that manipulate AI behavior require robust defense mechanisms.

Meanwhile, security challenges persist in containerized and orchestrated environments such as Kubernetes, where issues like misconfigurations, weak secrets management, and runtime security require continuous monitoring and advanced isolation strategies. Relevant Kubernetes security projects, such as OPA (Open Policy Agent) and Falco, are crucial for implementing these security measures effectively.
Furthermore, integrating security early in the development process—through shift-left practices in CI/CD pipelines—remains crucial for detecting and mitigating vulnerabilities early. As AI security evolves, protecting system prompts, preventing the disclosure of sensitive information, and mitigating adversarial attacks will be key priorities. Adversarial robustness training, anomaly detection, and penetration testing are essential for validating the security of AI systems.

As cloud native AI deployments become more pervasive, organizations must continuously evolve their security frameworks to address these emerging challenges while ensuring compliance and maintaining operational efficiency.

Appendix

Glossary

In addition to the listed items, please refer to the CNAI glossary, available at https://tag-runtime.cncf.io/wgs/cnaiwg/glossary/.

Attribute Based Access Control (ABAC)

Attribute Based Access Control (ABAC) is a method to manage user access to systems and data based on characteristics associated with users, rather than roles (see RBAC). This is a more flexible and fine-grained approach to assigning permissions than traditional role-based methods.

Authentication (AuthN)

Authentication is the process of verifying a user's or system's identity. It ensures that the entity attempting to access a system is who it claims to be.

Authorization (AuthZ)

Authorization determines whether a user or system has permission to access a resource or perform an action.

A Distributed Denial of Service (DDoS)

A Distributed Denial of Service (DDoS) attack is a malicious attempt to disrupt the normal functioning of a targeted server, service, or network by overwhelming it with excessive Internet

traffic. These attacks leverage multiple compromised computer systems, often part of a botnet, as sources of attack traffic.

Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) is a security vulnerability in web applications. It allows attackers to inject malicious scripts into web pages that other users view, potentially compromising their security and privacy. These scripts can execute in the user's browser, potentially leading to data theft, session hijacking, or other malicious activities.

Mutual TSL (mTLS)

Mutual TLS (mTLS) is an extension of the standard TLS (Transport Layer Security) protocol that adds a layer of security by requiring both the client and server to authenticate each other. This mutual authentication process ensures that both parties are verified and trusted, thereby enhancing the protection of sensitive data exchanges.

ChatGPT

ChatGPT is an advanced language model developed by OpenAI. It uses deep learning techniques, specifically a variant of the Transformer architecture, to generate human-like text based on the input it receives.

Chatbot

A chatbot is software designed to simulate human conversation through text or voice interactions. It automates communication, provides information, and assists users in performing tasks across various platforms, such as websites, messaging applications, and customer service interfaces.

Fine-tuning Data

Fine-tuning data is a specialized dataset that adapts pre-trained models to specific domains or tasks, often smaller than the original training data.

Identity and Access Management (IAM)

Identity and Access Management (IAM) is a framework of policies and technologies that ensures the right individuals have access to the right resources at the correct times for the right reasons.

Inference Data

Inference data contains information collected when the model is deployed and making predictions, including inputs, outputs, and associated metadata.

Long-lived Chat Contexts (Context Memory)

Long-lived chats are stored conversation histories, commonly referred to as “memory”, that provide context for AI systems engaged in ongoing dialogues with users.

Model Checkpoints

Model checkpoints are saved states of an AI model during or after training that capture weights, biases, and architecture configurations, allowing training to resume or the model to be deployed.

Model Context Protocol (MCP)

Model Context Protocol (MCP) is an open standard protocol developed by Anthropic to standardize how Large Language Models (LLMs) integrate with external tools, servers, and data sources. Users can expose their data or tools through MCP-compliant servers, then incorporate and build AI applications that connect to these servers as MCP clients. MCP acts as a bridge between AI models and external data sources, providing a standardized communication framework. MCP supports STDIO, Server Sent Events (SSE), and WebSockets communication methods.

Open Authorization (OAuth 2.0)

OAuth (Open Authorization) is an open standard for access delegation commonly used to grant websites or applications limited access to a user's information without exposing passwords. OAuth enables users to authorize third-party services to access their information from another service, allowing them to access user data from social media accounts without sharing login credentials.

OpenID Connect (OIDC)

OpenID Connect (OIDC) is an authentication layer built on the OAuth 2.0 protocol. It allows clients to verify the end-user's identity based on the authentication performed by an authorization server and obtain basic profile information about the user. OIDC is designed for federated identity and single sign-on (SSO), providing a secure and straightforward method for authenticating users across different domains and applications.

Open Security Controls Assessment Language (OSCAL)

Open Security Controls Assessment Language (OSCAL) is a standardized framework developed by the National Institute of Standards and Technology (NIST) to represent control catalogs, system security plans, and assessment plans in a machine-readable format. OSCAL aims to enhance the efficiency and consistency of security assessments and compliance reporting by providing a unified language that can be applied across various systems and tools.

Post-quantum cryptography (PQC)

Post-quantum cryptography (PQC) refers to cryptographic algorithms designed to be secure against the potential threats of quantum computers.

Role Based Access Control (RBAC)

Role Based Access Control (RBAC) is a method to manage user access to systems and data based on their assigned roles. Using RBAC, administrators can assign permissions for specific roles, rather than at an individual user level.

Training Data
Training data is raw data used to initially train AI models, containing labeled examples from which the model learns patterns and relationships.

Trusted Execution Environment (TEE)

A Trusted Execution Environment (TEE) is a secure area within a processor that provides security for code execution and data protection. It ensures that sensitive computations and data are isolated and protected from unauthorized access or tampering, even if the primary operating system is compromised. TEEs enhance application security by providing a trusted space for executing sensitive operations.

Validation Data

Validation data is data collection separate from the training used to tune hyperparameters and evaluate model performance during development, helping prevent overfitting.

Vector Embeddings

Vector embeddings are high-dimensional numerical representations of text, images, or other data that capture semantic meaning in a form that AI systems use.

Workload Identity (WID) / Non-Human Identity (NHI)

A cryptographically verifiable, non-human identity that a cloud-native compute workload, such as a containerized AI training job, inference service, or data-processing pipeline, receives at runtime. It enables the workload to obtain short-lived, least-privileged credentials to call cloud APIs, access data, and interact with other services without embedding static keys or secrets, allowing for secure, scalable, and multi-tenant AI operations across clusters and regions.

References & Citations

1. Cloud Native Security Whitepaper (Version 2)

2. CNCF Cloud Native AI Whitepaper

3. Cloud Native AI Personas
4. Presentation about security and ML
5. OWASP AI Security and Privacy Guide
6. OWASP Resources
7. AI/ML Security Group Documents
8. CNCF Landscape
9. CNAI WG Resources
10. Software Supply Chain Security Best Practices

Resources

1. CN Security Training
    a. Kubernetes and Cloud Native Security Associate (KCSA) - https://training.linuxfoundation.org/certification/kubernetes-and-cloud-native-security-associate-kcsa/
    b. Certified Kubernetes Security Specialist (CKS) - https://training.linuxfoundation.org/certification/certified-kubernetes-security-specialist/
2. AI Vulnerability listings
    a. GenAI Security Project by OWASP - https://genai.owasp.org/
    b. CVE Guidance for AI by CVEAI WG - https://www.cve.org/Media/News/item/blog/2025/02/18/CVE-ID-CVE-Record-AIrelated-Vulnerabilities

Authors

Aonan Guan, Boris Kurktchiev, Deep Patel, Joel Roberts, Josh Halley, Nimisha Mehta, Nina Polshakova, Pedro Ignacio, Ronald Petty, Saad Sheikh, Sudhanshu Prajapati, Victor Lu

Reviewers

Adel Zaalouk, Andrew Block, Jon Zeolla, Lei Wang, Nitin Naidu, Yoshiyuki Tabata

Acknowledgments

Members of the CNCF AI Working Group and TAG-Security contributed to this paper. Thank you to everyone.