ADA: Automated Moving Target Defense for AI Workloads via Ephemeral Infrastructure-Native Rotation in Kubernetes

Akram Sheriff
isheriff@cisco.com
Cisco Systems

Ken Huang
ken.huang@distributedapps.ai
distributedapps.ai

Zsolt Nemeth
zsolt@r6security.com
R6 Security

Madjid Nakhjiri
nakhjiri@yahoo.com
Independent Security Researcher

May 25, 2025

Abstract

This paper introduces the Adaptive Defense Agent (ADA), an innovative Automated Moving Target Defense (AMTD) system designed to fundamentally enhance the security posture of AI workloads. ADA operates by continuously and automatically rotating these workloads at the infrastructure level, leveraging the inherent ephemerality of Kubernetes pods. This constant, managed churn systematically invalidates attacker assumptions and disrupts potential kill chains by regularly destroying and respawning AI service instances. This methodology, applying principles of chaos engineering as a continuous, proactive defense, offers a paradigm shift from traditional static defenses that rely on complex and expensive confidential or trusted computing solutions to secure the underlying compute platforms, while at the same time agnostically supporting the latest advancements in agentic and non-agentic AI ecosystems and solutions such as agent-to-agent (A2A) communication frameworks or model context protocols (MCP). This AI native infrastructure design, relying on the widely proliferated cloud native Kubernetes technologies, facilitates easier deployment, simplifies maintenance through an inherent zero-trust posture achieved by rotation, and promotes faster adoption. We posit that ADA's novel approach to AMTD provides a more robust, agile, and operationally efficient zero-trust access technique for Agentic AI systems, achieving security through proactive container runtime environmental manipulation rather than reactive patching.

Keywords: AI Infra, Machine Learning, AMTD, Kubernetes, Container runtime security, Cloud Native AI security, Zero trust access (ZTA), Agent-to-Agent (A2A), Model Context Protocol (MCP), NVIDIA NIM, Agentic AI, Multi-Agent Systems, MAESTRO framework.

1 Introduction

The integration of Artificial Intelligence (AI) and Machine Learning (ML) is revolutionizing industries, with AI workloads increasingly deployed as microservices, such as NVIDIA NIM (NVIDIA Inference Microservices) [3]. These services are predominantly hosted on container orchestration platforms like Kubernetes, which offer scalability and resilience, but also introduce new security challenges [6].

Current security paradigms, often reliant on static defenses, struggle to adequately protect these fluid environments. One can go to great lengths in server and device design to achieve confidential or trusted computing for container images. Traditionally, sensitive applications such as those in financial, critical infrastructure, and defense have relied on expensive, complicated hardware-based secure execution environments such as ARM TrustZone, TCG TPM or similar. These solutions require not only special hardware considerations but also very careful design and implementation of hardware-firmware interactions to keep the workload execution within a trusted boundary and its interactions with the outside world secure.

The rise of AI applications, in particular those deploying multiple agents, and leveraging Gen AI models with intensive interactions with outside through prompts or tool interactions would make the design and implementation of a confidential computing environment even more complex. However, microservices systems are always one vulnerable misconfiguration or container image/firmware image vulnerability away from an exploit even in hardware-secured environments. As we will see in the MAESTRO threat modeling section of this paper, an AI system has many more components than a traditional application system that can be exploited. Models can be modified to include malware that perform privilege escalations and gain access to critical system components, input prompts into the system can include malware, and access to various AI system artifacts can be compromised. Attackers can exploit the predictability of static configurations to establish persistence and execute advanced attacks. Although NVIDIA provides tools and guidance to secure NIMs [3], the security of the underlying infrastructure remains a critical concern. This paper argues for a transformative approach to securing these AI workloads by making the infrastructure itself an active defense mechanism.

We introduce Adaptive Defense Agent (ADA), a system rooted in the principles of Automated Moving Target Defense (AMTD) [1]. ADA innovatively applies AMTD by continuously rotating AI workload instances (Kubernetes pods) at the infrastructure level. This automated "destroy and respawn" cycle makes the AI service instances ephemeral, directly invalidating attacker assumptions about the target's stability and accessibility. This concept draws parallels with chaos engineering, where controlled disruption is used not just for resilience testing, but as a continuous, proactive security strategy [2].

The key innovation of ADA lies in its infrastructure-native design, which leverages many Kubernetes components and features, which contrasts sharply with the complexities often introduced by agent-to-agent (A2A) security models. A2A solutions can require new security controls, complex identity management, and substantial maintenance efforts. ADA bypasses these by leveraging existing Kubernetes capabilities, offering:

- **Simplified Deployment:** Integrating as a Kubernetes controller with minimal changes to existing deployments.
- **Streamlined Maintenance:** Achieving a zero-trust posture through rotation, reducing the need for complex instance-level trust management.
- **Accelerated Adoption:** Aligning with the operational practices of the DevOps, LLMops, and MLOps teams familiar with Kubernetes [6].

ADA proposes a zero trust model "by design"—where the fleeting nature of an instance minimizes its implicit trust—rather than "by patching." This paper details the architecture of ADA, its operational advantages, and its potential to significantly enhance the security of modern AI deployments by providing a proactive, agile, and resilient defense.

2 Related Work

ADA builds on established concepts in MTD, chaos engineering, and Kubernetes security, synthesizing them into a novel defense strategy for AI workloads.

2.1 Moving Target Defense (MTD)

MTD aims to create dynamic, unpredictable systems to increase attacker costs and reduce their window of opportunity [1]. The techniques span the network, platform, application, and data layers. AMTD focuses on automating these defenses, crucial for cloud environments. ADA is an AMTD system that focuses on infrastructure-level rotation (re-puffing) for AI workloads. Academic research often highlights strategies such as spatial, temporal, and configuration randomization to thwart attackers.

2.2 Chaos Engineering for Security

Chaos engineering involves controlled experiments to build confidence in the resilience of a system [2]. Applying this to security involves proactive disruption to uncover weaknesses or make the environment unpredictable for attackers. ADA operationalizes this as a continuous defensive measure, where the "chaos" of pod rotation is the intended security mechanism.

2.3 Kubernetes Security

Kubernetes provides security features such as RBAC, pod security standards, and network policies [6]. Zero-trust principles are often implemented via service meshes. ADA complements these by addressing the temporal aspect of instance security, assuming a breach is possible, and aiming to limit its lifespan through rotation. Secure deployment on Kubernetes platforms such as Amazon EKS also involves adherence to the best practices outlined by cloud providers [10].

2.4 AI Workload Security (NVIDIA NIMs Focus)

Securing AI models, such as NIM, involves protecting weights, data, and the inference process [3]. NVIDIA offers guidance and tools for this. ADA adds a crucial infrastructure security layer by making NIM instances ephemeral, hindering attackers trying to establish persistence on these inference servers.

2.5 Ephemeral Infrastructure

The security benefits of ephemeral (short-lived) and immutable infrastructure are well recognized. Replacing instances rather than patching them reduces configuration drift and eliminates persistent threats. ADA embodies this for AI workloads as a proactive AMTD strategy.

2.6 Context-Aware Resource Mutation

Kubernetes manifests for agent deployments are dynamically mutated using CRD-defined policies. Mutation parameters are derived from telemetry feeds (e.g., Kube-state-metrics, OPA Gatekeeper alerts) and include GPU access changes, updated runtime environments, and new container images.

2.6.1 ADA Mutation Engine or Adaptive Mutator Layer

Beyond schedule-based rotation, ADA can be extended with context-sensitive Kubernetes manifest mutation, enabling deeper environmental adaptation. Mutation logic is defined through CRD-based policies and triggered by real-time telemetry sources —-such as Kube-state metrics, Prometheus alerts, or OPA Gatekeeper violations. These inputs drive selective updates to pod specifications, including:

- Switching container images based on risk level or freshness
- Adjusting access to specialized resources (e.g., GPUs or ephemeral volumes)
- Modifying runtime environments (e.g., switching from Python to CUDA contexts)

This approach introduces an additional control layer that changes workload posture based on observed conditions, enhancing ADA's resilience, and aligning with the principles of self-healing, adaptive infrastructure.

Listing 1: Context Mutation Policy Example

```
apiVersion: ADA.security.r6.dev/v1
kind: ContextMutationPolicy
metadata:
  name: nim-risk-based-mutation
spec:
  selector:
    matchLabels:
      app: nim-inference
  triggers:
    telemetrySources:
      - type: PrometheusAlert
        name: high_gpu_usage
      - type: GatekeeperViolation
        constraint: disallowed-hostpath
  mutations:
    - type: ContainerImageUpdate
      containerName: nim
      newImage: "nvcr.io/nim/secure-nim:latest"
    - type: ResourceAdjustment
      containerName: nim
      resources:
        limits:
          nvidia.com/gpu: "0"
        requests:
          cpu: "500m"
          memory: "256Mi"
    - type: EnvPatch
      containerName: nim
      env:
        - name: RUNTIME_MODE
          value: "SECURE"
```

2.7 MCP Security

Model Context Protocol (MCP) as a mechanism to provide agentic AI solutions to a user running an AI application that can deploy Gen AI models and a host of software tools through the services provided by a MCP server. MCP protocol has been subject to some scrutiny by the cybersecurity community. Many vulnerabilities have been identified, ranging from tool poisoning, MCP server poisoning, MCP client-MCP server communication vulnerabilities, etc. All components within the MCP ecosystem, ranging from the AI application to the MCP server and the underlying tools, can benefit from platform security hardening. ADA can be seen as very suitable in any deployment where any of the MCP components are running as container images.

2.8 A2A Security

Researchers in [14] examine how to securely build AI applications using Google's Agent-to-Agent (A2A) protocol, which allows autonomous agents to communicate and collaborate. The A2A protocol uses machine-readable AgentCards for discovery and secure JSON-RPC requests for task execution, with real-time updates via server-sent events.

The authors identify security risks such as prompt injection, data poisoning, replay attacks, and network threats. To address these, they recommend digitally signing AgentCards, strict input validation, robust authentication, and fine-grained access controls.

Case studies include secure collaborative document editing and privacy-preserving data analysis. The paper concludes that while A2A is powerful in building secure agentic AI systems, its effectiveness relies on strong cryptographic safeguards, continuous threat assessment, and zero-trust security practices. Although [14] proposes valid controls for A2A security, this paper presents a complementary approach that can serve as a final mitigation strategy when other security measures have been exhausted or compromised.

3 The ADA System: An Innovative Defense

ADA introduces a novel AMTD strategy by treating AI workload instances as inherently ephemeral components, continuously refreshed to maintain a secure baseline.

3.1 Core Principles: A New Defensive Posture

- **AMTD via Continuous Rotation:** The cornerstone of ADA is the automated, continuous cycle of destroying and respawning AI workload instances (pods), driven by either a fixed schedule or anomaly detection. This dual-trigger approach ensures proactive disruption of attacker persistence, including threats that may evade detection.

- **Infrastructure-Native Simplicity:** ADA integrates directly with Kubernetes, minimizing new architectural components and aligning with cloud-native operations [6].

- **Zero-Trust through Ephemerality:** ADA actualizes a zero trust position at the instance level. An instance is never trusted for long; trust is vested in the orchestration of its constant, clean refresh.

- **Chaos as Proactive Defense:** The "chaotic" but controlled rotation is the defense itself, fostering an environment inherently hostile to attackers [2].

3.2 Architecture: Lean and Integrated

Key components:

- **ADA Controller:** Orchestrates the rotation based on defined policies.

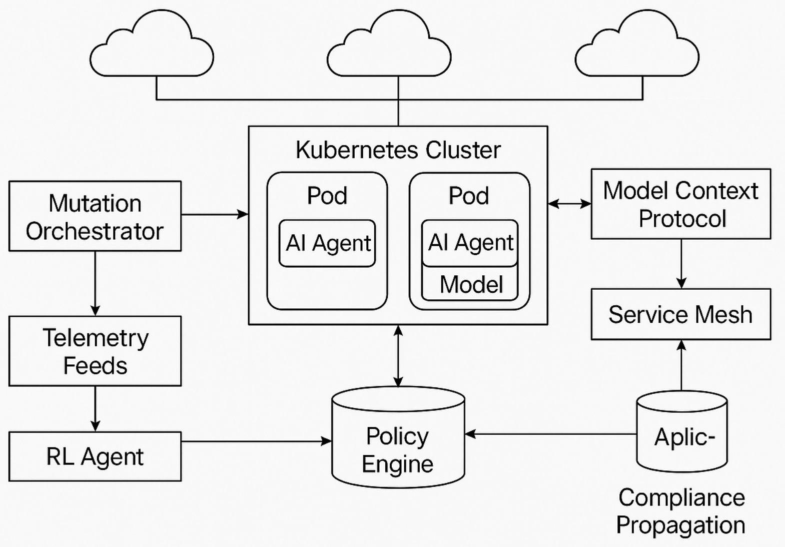


Fig. 1. Adaptive Moving Target Defense for AI Agents in Multi-Cloud Kubernetes Environments

Figure 1: ADA Architecture Diagram

- **Rotation Policies (CRDs/Annotations):** Define frequency (rotationInterval), strategy (e.g., "RollingUpdate"), and conditions for rotation.
- **Kubernetes API Integration:** All actions leverage the Kubernetes API, ensuring compliance with native security constructs [6].

This architecture is non-intrusive to the AI application (e.g., NIM [3]) itself.

3.3 Workflow: Automated and Continuous Refresh

ADA performs rotation by deleting and recreating pods on a fixed schedule, defined by a rotationInterval. This interval is configurable per workload via Kubernetes annotations or CRDs. By default, each pod is treated as stateless and ephemeral–only the container runtime, memory state, and temporary storage are recycled. Persistent resources (e.g., volumes, config maps) remain unchanged unless explicitly mutated.

1. **Onboarding:** AI workload (for example, NIM deployment [3]) is annotated or linked to an ADA policy.
2. **Monitoring & Policy Evaluation:** ADA Controller tracks instance age against rotationInterval.
3. **Rotation Execution:** The controller triggers a Kubernetes rolling update (or similar strategy) to replace old pods with new clean instances from the original image. This is seamless for stateless services like most NIMs [3, 6].
4. **Invalidation of Attacker Assumptions:** Each rotation aims to:
    - Change network presence (new IP of the pod).

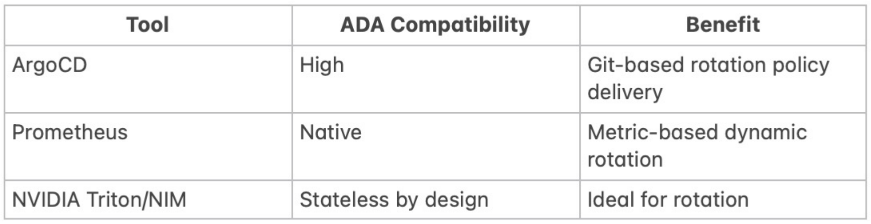
- Wipe in-memory attacker artifacts.
- Remove the exploited state specific to the instance.
- Nullify compromised credentials within the pod.
- Increase the cost of persistence of the attacker significantly [1].

4 ADA vs. A2A Security: A Leap in Simplicity and Agility

ADA's innovative posture is clear when contrasted with traditional or complex Agent-to-Agent (A2A) security controls, which often layer on new protocols and identity management for interservice trust.

- **Deployment Simplicity:** ADA uses existing Kubernetes knowledge [6], unlike A2A models that might require the adoption of new SDKs or protocols by application developers.
- **Maintenance Efficiency:** Zero-trust via rotation sidesteps complex agent identity lifecycles or A2A schema management. Security is an emerging property of the behavior of the infrastructure.
- **Rapid Adoption:** Familiarity for Kubernetes teams accelerates uptake.
- **Pragmatic Zero-Trust:** ADA offers zero trust "by design" for the instance itself by making it transient, reducing the reliance on intricate, potentially fallible cryptographic verification between every instance of an ephemeral agent. The security of the Kubernetes control plane itself remains paramount [6].

While A2A security is vital for certain interaction patterns, ADA argues that for securing the AI workload instance itself, continuous rotation offers a more robust and operationally simpler path within Kubernetes. This approach reduces the temporal attack surface in any single instance [1].



Tool	ADA Compatibility	Benefit
ArgoCD	High	Git-based rotation policy delivery
Prometheus	Native	Metric-based dynamic rotation
NVIDIA Triton/NIM	Stateless by design	Ideal for rotation

Figure 2: ADA vs A2A Security Comparison Matrix

5 Security Considerations

ADA may be implemented as a multi-agent system. This section outlines its security considerations using the MAESTRO framework for structured threat modeling.MAESTRO (Multi-Agent Environment, Security, Threat, Risk, and Outcome) is a layered and architectural framework for threat modeling multi-agent systems. Structures the analysis across seven distinct layers, from foundation models to the agent ecosystem, enabling identification of layer-specific and cross-layer vulnerabilities. MAESTRO emphasizes the dynamic and interconnected nature of agentic systems, providing a comprehensive approach to security that considers AI-specific risks, traditional security principles, and the overall system architecture.

5.1 Threat Model: ADA and MAESTRO

ADA's core mission is to thwart attackers seeking persistence on AI workload instances (e.g., NIM pods [3]) for data/model theft, lateral movement, or resource abuse. We can more rigorously model this threat landscape using the MAESTRO framework to better cover a range of potential vulnerabilities in our environment.

5.1.1 MAESTRO-Guided Threat Analysis

The following analysis maps ADA components and mitigations in the seven architectural layers of MAESTRO to provide a comprehensive threat model. Note that many of the cross-layer threats are the same as the core threats in a secure MAESTRO environment.

Layer 1: Foundation Models

- **Threat:** Model Poisoning (Indirect) - While ADA does not directly influence model training, a compromised container (Layer 4) could exfiltrate data used for future fine-tuning.
- **Mitigation:** Robust image scanning and CI/CD policies (Layer 4) to ensure trusted base images.

Layer 2: Data operations

- **Threat:** Data Exfiltration (Indirect) - Attackers could exfiltrate model weights or other data before rotation occurs if proper access controls are not in place (Layer 6).
- **Mitigation:** Fine-grained access controls (RBAC, Network Policies - Layer 6) restrict data access. Secure credential management (Layer 6) prevents credential theft.

Layer 3: Agent framework

- **Threat:** Privilege Escalation via CRDs - Maliciously crafted ADA CRDs or vulnerabilities in the ADA controller code itself could lead to privilege escalation within the Kubernetes cluster (Layer 4).
- **Mitigation:** Rigorous validation of CRDs, minimal permissions for the ADA controller service account, and code reviews (Layer 6). Protect admission controllers.
- **Threat:** Tool Misuse - An attacker gains knowledge of internal APIs and calls in ADA to manipulate the system.
- **Mitigation:** Apply least privilege principles and RBAC.

Layer 4: Deployment & Infrastructure

- **Threat:** Compromised Container Images - Malicious code could be injected into the container images used by ADA, compromising workload instances.
- **Mitigation:** Image signing, enforced scanning, and policy setting in CI / CD pipelines. Use short-lived credentials and external secret managers.
- **Threat:** Orchestration Attacks - Vulnerabilities in Kubernetes itself or misconfigurations could be exploited for unauthorized access/control.

- **Mitigation:** Regular security audits, adherence to Kubernetes best practices, and use of tools such as OPA Gatekeeper or Kyverno for policy enforcement.
- **Threat:** Lack of network policies - Open network policies can expose lateral movement opportunities.
- **Mitigation:** Apply K8s Network Policies using a service mesh.

Layer 5: Evaluation & Observability

- **Threat:** Insufficient Logging/Visibility - If ADA rotations aren't properly logged and monitored, detecting persistent threats becomes difficult.
- **Mitigation:** Integrate ADA with observability tools (Prometheus, Grafana) to track rotation frequency, failures, and other key metrics. Ensure proper audit trails. Limit automated changes and keep audit trails to monitor the state.
- **Threat:** Over-rotation covering persistent threats: If ADA's rotation interval is too fast, then a persistent threat could be covered up.
- **Mitigation:** Integrate observability tools and security auditing to ensure that this is not the case.

Layer 6: Security and compliance

- **Threat:** Poor RBAC Settings - Overly permissive RBAC settings for the ADA controller service account could lead to privilege escalation.
- **Mitigation:** Enforce least privilege for the ADA controller. Implement data plane or workload-level data auditing (Falco).
- **Threat:** Evasion - Attackers use adversarial techniques to bypass the system.
- **Mitigation:** Data Poisoning AI driven input validation; rerandomization of agent containers.

Layer 7: Agent ecosystem

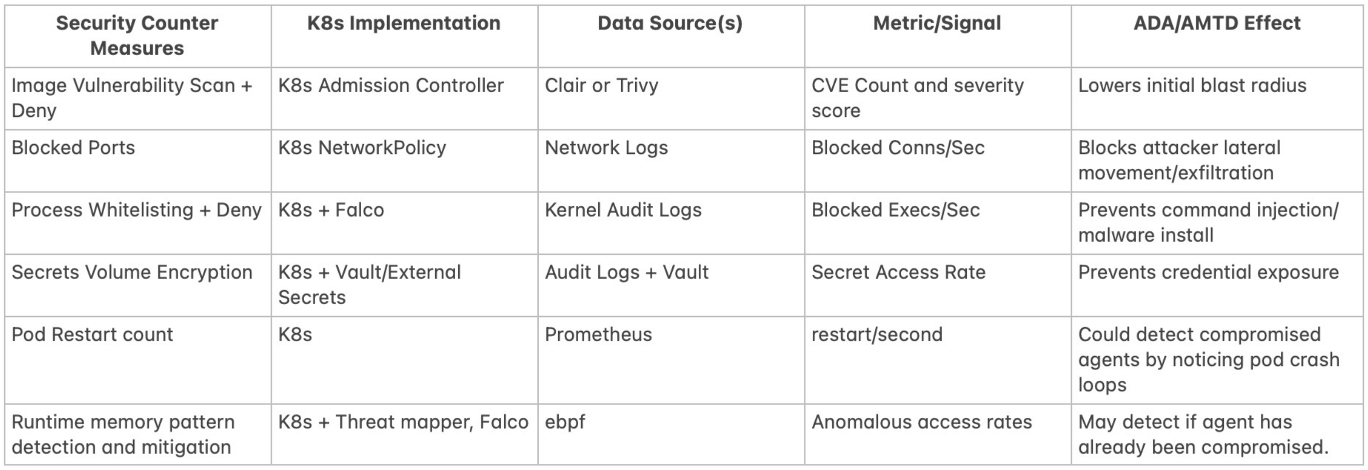
- **Threat:** Overly permissive deployment policies: enforce minimum privilege.
- **Mitigation:** Validate deployments policies using K8s Network Policies.

5.1.2 Cross-Layer Threats

- **Privilege Escalation & Lateral Movement (Layers 4, 6, 7):** An attacker gaining initial access to a container could exploit overly permissive RBAC (Layer 6) and network policies (Layer 7) to move laterally within the cluster and eventually compromise the Kubernetes control plane (Layer 4).
- **Supply Chain Compromise (Layers 1, 3, 4):** A tainted base image (Layer 1) could be used in the container image (Layer 4), which in turn affects the ADA controller and thus affects everything.

- **Over-Rotation Obscuring Threats (Layers 4, 5):** Overfrequent rotation without proper monitoring (layer 5) could mask persistent threats. An attacker might gain a foothold, but the rapid rotation cycle simply wipes them out before they can be detected, allowing them to reinfect a fresh instance shortly after.

By systematically analyzing each layer and the interactions between layers, we work towards developing a list of technologies and tools complementary to the ADA deployment. a more comprehensive understanding of the potential threats facing ADA. This informs the selection of effective mitigation strategies that could be added to Kubernetes-based ADA deployment.



Security Counter Measures	K8s Implementation	Data Source(s)	Metric/Signal	ADA/AMTD Effect
Image Vulnerability Scan + Deny	K8s Admission Controller	Clair or Trivy	CVE Count and severity score	Lowers initial blast radius
Blocked Ports	K8s NetworkPolicy	Network Logs	Blocked Conns/Sec	Blocks attacker lateral movement/exfiltration
Process Whitelisting + Deny	K8s + Falco	Kernel Audit Logs	Blocked Execs/Sec	Prevents command injection/ malware install
Secrets Volume Encryption	K8s + Vault/External Secrets	Audit Logs + Vault	Secret Access Rate	Prevents credential exposure
Pod Restart count	K8s	Prometheus	restart/second	Could detect compromised agents by noticing pod crash loops
Runtime memory pattern detection and mitigation	K8s + Threat mapper, Falco	ebpf	Anomalous access rates	May detect if agent has already been compromised.

Figure 3: ADA Performance Metrics and Thresholds

5.1.3 Threat Mitigation Coverage

Figure 4 outlines the effectiveness of AMTD against high value AI agent attack vectors.

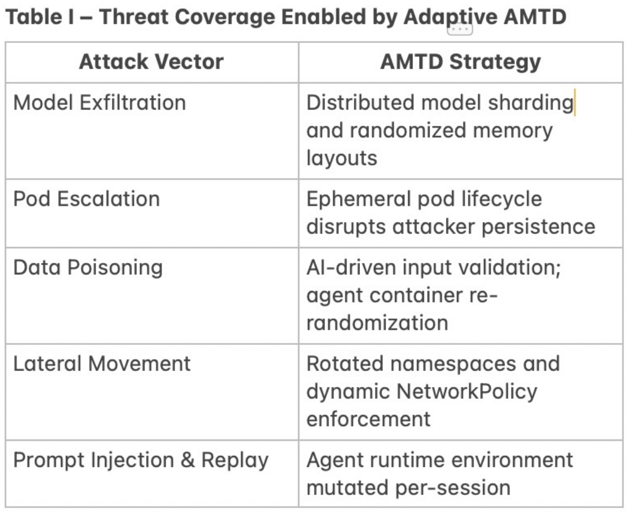


Table I – Threat Coverage Enabled by Adaptive AMTD

Attack Vector	AMTD Strategy
Model Exfiltration	Distributed model sharding and randomized memory layouts
Pod Escalation	Ephemeral pod lifecycle disrupts attacker persistence
Data Poisoning	AI-driven input validation; agent container re-randomization
Lateral Movement	Rotated namespaces and dynamic NetworkPolicy enforcement
Prompt Injection & Replay	Agent runtime environment mutated per-session

Figure 4: Threat Coverage Enabled by Adaptive AMTD

The following attack vectors and their corresponding AMTD strategies demonstrate the comprehensive threat mitigation approach of ADA:

- **Model Exfiltration:** Distributed model sharding and randomized memory layouts
- **Pod Escalation:** Ephemeral pod lifecycle disrupts attacker persistence
- **Data Poisoning:** AI-driven input validation; agent container re-randomization
- **Lateral Movement:** Rotated namespaces and dynamic NetworkPolicy enforcement
- **Prompt Injection & Replay:** Agent runtime environment mutated per-session

Integration with Agentic AI Platforms Agentic workloads based on JSON-RPC or LangGraph routing benefit from AMTD in the following ways:

- **Adaptive Routing Control:** ADA-AMTD dynamically reroutes agent invocations via updated paths in the orchestration DAG upon anomaly detection.
- **Runtime RAG Sandbox:** Retrieval-Augmented Generation (RAG) components run in isolated pods with limited time-bound data exposure.
- **Short-Lived Checkpointing:** Stateful agent memory is written to encrypted ephemeral volumes, which are rotated and re-keyed on every inference cycle.

Operational Considerations To ensure low disruption, AMTD's rotation cadence is adjusted based on service health metrics (latency, throughput) and threat risk scores. Mutation policies are versioned via GitOps pipelines (e.g., ArgoCD), and observability integrations filter out expected noise from benign mutations.

5.2 Metrics

To validate the effectiveness of ADA, we propose monitoring the following key metrics:

- **Time-to-Evict (TTE):** Should align with rotationInterval.
- **Attacker Effort Increase:** Quantified by chain disruption.
- **Service Impact:** Measured by uptime and latency during rotation.
- **Resource Overhead:** Controller and churn impact, measured against established baselines.

5.3 Other Security Considerations

5.3.1 Misconfigurations

Threats: Poor RBAC settings, overly permissive policies, lateral movement opportunities. **Mitigations:** Enforce least privilege, validate CRDs, use Kubernetes network policies.

5.3.2 Accounts & Secrets

Threats: Hardcoded secrets, leaked service accounts, no rotation. **Mitigations:** Use short-lived credentials, external secret managers, rotate secrets automatically.

5.3.3 Ephemeral Infrastructure

Threats: Predictable naming, persistent data, short but exploitable windows. **Mitigations:** Use randomized names, ephemeral volumes, trigger resets on anomalies.

5.3.4 Supply Chain

Threats: Tainted images, compromised build pipelines, unreliable artifacts. **Mitigations:** Sign images, enforce scanning and policy gating in CI/CD.

5.3.5 Tenancy Boundaries

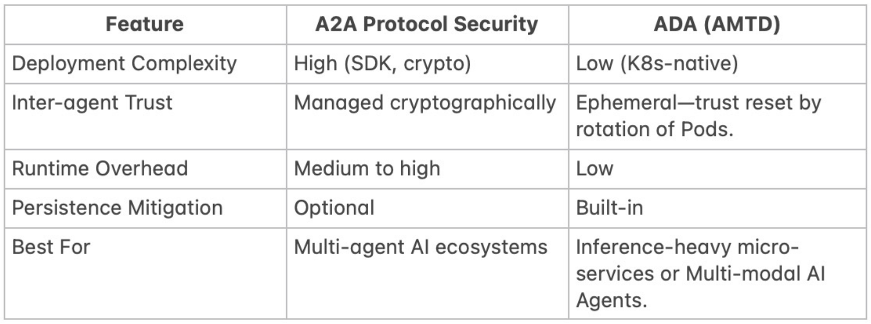
Threats: Namespace collisions, unauthorized access by the tenant. **Mitigations:** Use namespace scoping, enforce strict tenant policies, isolate workloads.

5.3.6 Runtimes & APIs

Threats: Dangerous CRDs, unvalidated API use, privilege escalation paths. **Mitigations:** Protect admission controllers, validate CRD, enable API auditing.

5.3.7 Operations

Threats: Over-rotation hiding persistent threats, insufficient visibility. **Mitigations:** Integrate observability tools, limit automated changes, keep audit trails.



Feature	A2A Protocol Security	ADA (AMTD)
Deployment Complexity	High (SDK, crypto)	Low (K8s-native)
Inter-agent Trust	Managed cryptographically	Ephemeral—trust reset by rotation of Pods.
Runtime Overhead	Medium to high	Low
Persistence Mitigation	Optional	Built-in
Best For	Multi-agent AI ecosystems	Inference-heavy micro-services or Multi-modal AI Agents.

Figure 5: Security Considerations and Mitigation Strategies

5.4 Example Scenario: Compromised NIM

A compromised NIM instance [3] is automatically replaced by ADA within the rotation interval, thwarting prolonged attacks. Without ADA, the compromise could persist indefinitely.

5.5 Applicability

The proposed approach is ideal for stateless AI inference services (NIMs [3, 14]), high-risk services, and environments with rapid CI/CD, leveraging Kubernetes [6, 10].

5.6 Threat Mitigation Coverage

Table 1 outlines the effectiveness of AMTD against high value AI agent attack vectors.

Table 1: Threat Coverage Enabled by Adaptive AMTD

Attack Vector	**AMTD Strategy**
Model Exfiltration	Distributed model sharding and randomized memory layouts
Pod Escalation	Ephemeral pod lifecycle disrupts attacker persistence
Data Poisoning	AI-driven input validation; agent container re-randomization
Lateral Movement	Rotated namespaces and dynamic NetworkPolicy enforcement
Prompt Injection & Replay	Agent runtime environment mutated per-session

5.6.1 Integration with Agentic AI Platforms

Agentic workloads based on JSON-RPC or LangGraph routing benefit from AMTD in the following ways:

- **Adaptive Routing Control:** ADA-AMTD dynamically reroutes agent invocations via updated paths in the orchestration DAG upon anomaly detection.
- **Runtime RAG Sandbox:** Retrieval-Augmented Generation (RAG) components run in isolated pods with limited time-bound data exposure.
- **Short-Lived Checkpointing:** Stateful agent memory is written to encrypted ephemeral volumes, which are rotated and re-keyed on every inference cycle.

5.6.2 Operational Considerations

To ensure low disruption, AMTD's rotation cadence is adjusted based on service health metrics (latency, throughput) and threat risk scores. Mutation policies are versioned via GitOps pipelines (e.g., ArgoCD), and observability integrations filter out expected noise from benign mutations.

6 Discussion: The ADA Advantage and Future

6.1 Benefits: A proactive and resilient defense

ADA offers proactive security, resilience against instance-specific exploits, a drastically reduced attacker window, and operational simplicity for Kubernetes users [1, 6]. Its innovative strength is making the environment itself the primary defense.

6.2 Limitations

- **Stateful Applications:** The main challenge requiring complex state migration strategies.
- **Ultra-Fast Attacks:** May not prevent the initial "smash and grab" but clean up afterward.
- **Performance:** Churn needs careful management [6].
- **Orchestrator Security:** Relies on a secure Kubernetes control plane [6].

6.3 Future Work: Expanding Innovation

6.3.1 Formal Security Modeling

While ADA demonstrates a strong conceptual foundation, future work should focus on formally quantifying its security impact. This includes modeling attacker dwell time, rotation-induced uncertainty, and disruption of kill chains using techniques such as attack graphs, game theory, or simulation frameworks. Such models will provide measurable assurance of ADA's Moving Target Defense (MTD) benefits.

6.3.2 Rotation-Aware Observability

Frequent instance rotation can produce telemetry churn and false positives in conventional monitoring stacks. ADA should incorporate rotation-aware observability by tagging pod lifecycle events and enriching logs and metrics with rotation context. This allows tools such as Prometheus, Grafana, and SIEM systems to suppress noise and retain clarity during normal churn cycles.

6.3.3 Adaptive Rotation Based on Risk Scores

Rather than relying solely on time-based or static thresholds, ADA can evolve to support rotation policies based on dynamic risk signals. These could be derived from ML-based anomaly detection, policy engines (e.g., OPA), or behavioral scoring. Higher-risk workloads could be rotated more aggressively, aligning defense efforts with situational awareness.

6.3.4 Integration with Confidential Computing

To extend the protection to the data in use, ADA can be adapted to run within confidential computing environments such as Intel SGX or AMD SEV. By rotating pods inside secure enclaves, ADA can combine workload churn with hardware-backed isolation, addressing advanced threat scenarios, including insider attacks or host compromise.

6.3.5 Extension to Stateful Workloads

Currently optimized for stateless services, ADA's applicability to stateful or long-running tasks (e.g., fine-tuning or reinforcement learning agents) requires future support for state checkpointing and safe rehydration strategies. Exploring rotation-aware storage, such as volume snapshotting or sidecar-based state offloading, will expand the scope of ADA to a broader class of AI workloads.

6.3.6 Secure Sandboxed Execution for Agentic Code

For multi-agent or GenAI workloads that may execute untrusted or self-modifying code, lightweight sandboxing tools like Pyodide (Python in WebAssembly) and Deno (secure JavaScript/TypeScript runtime) can be integrated. These frameworks provide run-time confinement and synchronous restrictions at the application layer. While not part of ADA's core, integrating sandboxing at the pod level complements rotation by adding execution isolation within each short-lived instance.

7 Conclusion

ADA presents a forward-thinking AMTD system that leverages the inherent dynamism of Kubernetes to provide robust security for AI workloads such as NIMs [1, 3, 6]. By making instances ephemeral through continuous rotation, it offers a simpler, more agile approach than complex A2A security frameworks for instance hardening. This innovative use of infrastructure as an active defense mechanism marks a significant step towards building intrinsically secure AI systems. ADA champions a shift from static fortifications to dynamic and resilient defenses, making the operational environment a core component of its own protection.

8 Disclaimer

The views and opinions expressed in this paper are those of the authors and do not necessarily reflect the official policy or position of their respective organizations. The research presented here is for Open Source, academic, research, and educational purposes only.There is no IP or copyright violation by doing this open source AI Research work for creating and publishing this paper.

References

[1] Cai, G., Wang, B., Liu, Y., Zhang, W., Li, B., Li, H., & Xia, C. (2020). Moving target defense: state of the art and characteristics. *Frontiers of Information Technology & Electronic Engineering*, 21(10), 1429-1457.

[2] Akitra. (2025, April 7). Chaos Engineering in Cybersecurity: Stress-Testing Systems to Build Resilience. *Akitra Blog.*

[3] NVIDIA Developer. (2025). NVIDIA NIM. Retrieved May 13, 2025.

[4] NVIDIA Press Release. (2024, March 18). NVIDIA Launches Generative AI Microservices for Enterprises to Create and Deploy Custom Applications on Their Platforms.

[5] Basiri, A., Ghasemzadeh, N., Khorsandroo, S., & Adhikari, A. (2016). *Chaos Engineering.* O'Reilly Media.

[6] Kubernetes Authors. (n.d.). Security. *Kubernetes Documentation.* Retrieved May 13, 2025.

[7] Cho, J. H., Sharma, D. P., Alavizadeh, H., Yoon, S., Ben-Asher, N., Moore, T. J., ... & Kim, D. S. (2020). Toward proactive, adaptive defense: A survey on moving target defense. *IEEE Communications Surveys & Tutorials*, 22(1), 709-745.

[8] Evans, D., Nguyen-Tuong, A., & Knight, J. (2011). ROTE: A Runtime System for Moving Target Defense. Technical Report CS-2011-02, University of Virginia.

[9] Styra. (2023, October 30). Best Practices for Kubernetes Security. *Styra Blog.*

[10] Amazon Web Services. (n.d.). What is Amazon EKS? Retrieved May 13, 2025.

[11] Okenyi, P. O., & Nwakanma, I. C. (2021). A survey on moving target defense strategies: A game-theoretic approach. *Journal of Cybersecurity and Privacy*, 1(3), 487-508.

[12] Al-Shaer, E., Duan, Q., & Jajodia, S. (Eds.). (2021). *Moving Target Defense: Concepts, Metrics, and Engineering.* Springer.

[13] Jajodia, S., Ghosh, A. K., Swarup, V., Wang, C., & Wang, X. S. (Eds.). (2011). *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats.* Springer.

[14] Idan Habler, Ken Huang, Prashant Kulkarni, and Vineeth Sai Narajala. “Building A Secure Agentic AI Application Leveraging A2A Protocol.” *arXiv preprint arXiv:2504.16902*, May 2025.

[15] Akram Sheriff, et al. “Runtime security analytics for serverless workloads.” 2023.

[16] Jajodia, Sushil, et al. *Moving target defense: Creating asymmetric uncertainty for cyber threats.* Springer Science & Business Media, 2011.

[17] Akram Sheriff, et al. “’EVENT MESH’ TRIGGERED METHOD FOR HYBRID CLOUD CHAINING VIA TUNNELING.” Technical Disclosure Commons, 2021. `https://www.tdcommons.org/dpubs_series/4505/`

[18] Akram Sheriff, et al. “Dynamic proxy response from application container.” U.S. Patent US11689505B2, 2023. `https://patents.google.com/patent/US11689505B2/en`

[19] Burns, Brendan, et al. “Design and implementation of the Kubernetes container orchestrator.” 2016.

[20] MITRE ATT&CK Framework. `https://attack.mitre.org`

[21] Akram Sheriff, et al. “DYNAMIC RESOURCE PROFILE-BASED CONTAINER AND SERVERLESS CONSTRUCT FOR COMPOSABLE OBSERVABILITY TRACING.” Technical Disclosure Commons, 2022. `https://www.tdcommons.org/dpubs_series/5814/`