Proactive Security for Large-Scale AI Inference: Adaptive Container Orchestration at the Edge

Akram Sheriff
Cisco Systems
San Jose, CA, USA
isheriff@cisco.com

Zsolt Németh
R6 Security
Berkeley Hills, CA, USA
zsolt@r6security.com

Ken Huang
Distributedapps.ai
Fairfax, VA, USA
ken.huang@distributedapps.ai

***Abstract*—This paper introduces Adaptive NIMs, a novel application of Automated Moving Target Defense (AMTD) to AI inference environments. It addresses the security and reliability risks of static AI deployments, such as NVIDIA NIMs, by dynamically rotating containers, modifying runtime configurations, and integrating telemetry-driven decision-making. The system aims to proactively reduce exploit windows without significant performance overhead.**

The document details the system's architecture, which includes a Kubernetes Operator, Rotation Engine, Immutable Image Registry, and Telemetry & Decision Module. It outlines a rotation workflow triggered by timers, security anomalies, or resource scaling needs, and discusses performance considerations such as minimal overhead and cost efficiency.

Evaluation results show that AMTD significantly increases attack disruption rates (up to 94%) and drastically reduces attacker dwell time (to minutes), while maintaining high availability (99.8%) and incurring minimal cost overhead (1.3-1.7%). The paper emphasizes that AMTD acts as a crucial enforcement layer, transforming detection into guaranteed attack disruption, and highlights its importance for securing high-value AI deployments. Future work includes integrating AMTD natively into orchestration platforms and expanding its application to other mutable infrastructure elements.

I. Introduction

The rapid growth of generative AI and large language models (LLMs) has shifted computational demand toward high-performance inference workloads, often running in containerized GPU clusters. Solutions such as NVIDIA NIMs, CoreWeave deployments, and other model-serving infrastructures enable unprecedented scalability -but their static, long-lived nature introduces inherent vulnerabilities.

Traditional cloud and container security models emphasize detection and patching after compromise. In contrast, **Automated Moving Target Defense (AMTD)** seeks to prevent compromise by continuously shifting the attack surface. While AMTD concepts have been explored in general cloud and network security, their application to **AI inference workloads** -where uptime, latency, and cost constraints are critical - remains largely unexplored.

In this work, we propose **Adaptive NIMs**, an AMTD-based framework specifically tailored for securing AI inference pipelines. Our contributions include:

1) **A security model** mapping AMTD techniques to the AI inference threat landscape.
2) **A Kubernetes-native orchestration layer** for dynamic container rotation and configuration mutation.
3) **Evaluation results** demonstrating minimal performance degradation and measurable improvements in security posture.

We argue that securing AI inference requires **proactive, adaptive measures** that integrate directly with deployment infrastructure, rather than bolted-on detection mechanisms. The remainder of this paper details the design, implementation, and initial results of Adaptive NIMs, and outlines a roadmap for integrating such approaches into the broader AI security ecosystem.

II. Background & Related Work

A. Automated Moving Target Defense (AMTD)

Automated Moving Target Defense is a proactive security paradigm aimed at reducing an attacker's ability to perform reconnaissance, persistence, and exploitation by continuously changing the system's attack surface. Early AMTD research focused on network address randomization, instruction set randomization, and virtual machine migration [9]. More recent work has explored AMTD in cloud computing [10], particularly for containerized workloads [11].

In terms of agentic AI use cases, ADA (Automated Moving Target Defense for AI Workloads via Ephemeral Infrastructure-Native Rotation in Kubernetes) established that continuous, automated rotation of AI workloads disrupts attacker persistence across Kubernetes deployments. In contrast, Phoenix adapts this concept to inference workloads by integrating detection tools, staggered warm-standby rotation, and telemetry-driven triggers for sub-second enforcement [1].

However, these approaches often assume homogeneous workloads and do not account for **AI inference environments**, where high GPU utilization, low-latency responses, and predictable scaling patterns are paramount.

B. AI Infrastructure Security

The rise of **Generative AI** and **Large Language Models** (LLMs) has introduced new attack vectors, including prompt injection, model poisoning, data exfiltration via covert channels, and supply chain attacks on pre-trained models. Ken Huang, a recognized authority in AI security, has highlighted

that "AI infrastructure inherits all the vulnerabilities of cloud-native systems but also introduces unique risks in model integrity and inference pipelines." These risks are amplified in inference-serving platforms such as NVIDIA NIMs, where containers are long-lived and often exposed to high-volume, unpredictable user inputs.

C. NIMs and the Static Deployment Problem

NVIDIA NIMs provide an efficient mechanism to deploy AI models as containerized inference services across Kubernetes, EC2, or other environments. While optimized for performance and portability, **NIM deployments today are largely static**: the same container image and runtime configuration persist for extended periods. Zsolt Németh, founder of R6 Security, observes that "in static deployments, every hour without change is an extra hour for an attacker to map, probe, and exploit." In traditional IT systems, patch cycles can mitigate some risks - but in fast-moving AI workloads, zero-day vulnerabilities or configuration leaks can be exploited within minutes.

D. AMTD for AI Inference

Few academic works address the intersection of AMTD and AI inference. The most closely related research involves dynamic microservice shuffling [12] and function-level obfuscation in serverless environments [13]. Our work differs in three key ways:

1) **Inference-Aware Design:** Adaptive NIMs are optimized for GPU-bound workloads with strict latency requirements.
2) **Telemetry-Driven Adaptation:** Rotation schedules and configuration mutations are triggered by runtime metrics (e.g., Prometheus alerts) rather than fixed timers.
3) **Zero-Trust Integration:** The framework assumes that any long-lived deployment is a potential compromise point and proactively mitigates persistence opportunities.

III. Threat Model

A. Assumptions

Our work assumes the following:

1) **Target Environment:** AI inference workloads deployed on containerized infrastructures, primarily Kubernetes-based, either in cloud or edge environments.
2) **Attacker Capabilities:** Adversaries may have partial knowledge of the target's architecture but lack full, real-time visibility into ephemeral resource allocation. Attackers can perform reconnaissance, exploit known vulnerabilities, and maintain persistence once inside.
3) **Defender Capabilities:** The defender controls the orchestration layer and can enforce automated re-deployment, re-configuration, or relocation of workloads without prior warning to the attacker.

B. Adversary Goals

We focus on adversaries attempting to:

- **Persistence:** Maintain long-term control over a compromised container or node.
- **Model Extraction:** Steal or replicate proprietary AI models deployed in inference containers.
- **Data Exfiltration:** Extract sensitive input/output data from inference services.
- **Availability Disruption:** Cause downtime or degrade inference quality via targeted attacks.

C. Threat Vectors

The primary attack vectors relevant to our scope are:

1) **Container Escape & Privilege Escalation:** Exploiting vulnerabilities in container runtimes (e.g., runc CVE-2019-5736) to gain host-level control.
2) **Model Theft via Side-channel Attacks:** Extracting model parameters or structure through query-based or timing side-channel analysis.
3) **Configuration Exploitation:** Leveraging exposed Kubernetes API endpoints, misconfigured RBAC policies, or leaked secrets to manipulate workload placement.
4) **Adversarial Workload Co-location:** Placing a malicious container on the same node to exploit shared memory, cache timing, or PCIe bus vulnerabilities.

D. Defensive Scope of AMTD

Automated Moving Target Defense mitigates these threats by:

- **Reducing Persistence Opportunities:** Regularly terminating and replacing containers before attackers can fully exploit or maintain footholds.
- **Disrupting Reconnaissance:** Changing IPs, container IDs, and resource mappings frequently to invalidate attacker intelligence.
- **Increasing Attack Costs:** Forcing adversaries to re-initiate exploitation in each new environment cycle.

IV. AMTD Design for AI Inference Workloads

A. Design Objectives

Our Automated Moving Target Defense (AMTD) system for AI inference workloads aims to:

1) **Disrupt adversary persistence** without degrading model performance.
2) **Automate orchestration-level defense** with minimal manual intervention.
3) **Integrate seamlessly** with existing Kubernetes and containerized AI deployment pipelines.
4) **Adapt** rotation frequency and methods based on live threat telemetry.

B. System Architecture

The AMTD framework consists of four primary components:

1) **Kubernetes Operator:**
   - Manages container lifecycle events and enforces rotation policies.
   - Integrated with Prometheus metrics for trigger-based actions (e.g., CPU/GPU anomalies, abnormal network patterns).
2) **Rotation Engine:**
   - Performs workload re-deployment with randomized parameters: new container IDs, ephemeral node assignments, shuffled port mappings.
   - Supports multiple rotation strategies: time-based (fixed interval), event-based (triggered by detected anomalies), and hybrid (combination of time and event triggers).
3) **Immutable Image Registry:**
   - Stores trusted, signed AI inference container images (e.g., Nvidia NIMs).
   - Ensures rollback capability to last known-good configuration.
4) **Telemetry & Decision Module:**
   - Analyzes Prometheus/Grafana streams, intrusion detection alerts, and infrastructure logs.
   - Applies decision logic for immediate rotation or parameter tuning.

C. Novel Elements for AI-Inference-Aware AMTD

While prior AMTD approaches focus primarily on containers and networking, AI inference workloads present **unique attack surfaces** (GPU-bound execution, model pipelines, latency constraints). Our design introduces several novel elements tailored to this environment:

1) GPU-Aware Micro-Shuffling: Dynamically reassigns GPU bindings, including MIG (Multi-Instance GPU) slices or NVLink interconnect mappings. This prevents side-channel persistence by ensuring attackers cannot rely on fixed GPU memory layouts.

2) Ephemeral Model Wrappers: Models are deployed inside short-lived "capsules" that randomize API boundaries, enforce input preprocessing mutations, or encrypt intermediate outputs. This disrupts prompt-injection persistence and complicates model extraction attacks.

3) Telemetry-Weighted Rotation Scheduling: Rotation frequency is not fixed but proportional to anomaly scores computed from runtime telemetry (e.g., GPU entropy, query deviation patterns, cache misses). This reduces attacker stealth advantages by accelerating rotations during suspicious activity.

4) Confidential Deployment Mutations: Each container rotation mutates its runtime hardening profile (Seccomp, AppArmor, RBAC policies), ensuring no stable persistence environment for attackers, even if they compromise a container instance.

5) Encrypted Ephemeral Memory Spaces: Each inference container mounts disposable encrypted memory volumes with unique, random keys. Keys are destroyed at container termination, eliminating opportunities for memory-resident model theft.

D. Rotation Workflow

The workflow operates as follows:

1) **Deploy** inference container from the immutable registry.
2) **Monitor** using Prometheus and intrusion detection feeds.
3) **Trigger Rotation** based on: timer expiry (time-based), security anomaly detection (event-based), or resource scaling needs (performance-based).
4) **Terminate** existing containers and replace them with a new instance using randomized parameters.
5) **Update Service Endpoints** without interrupting client connections (leveraging Kubernetes Service abstraction).

E. Performance Considerations

AMTD is designed to have:

- **Minimal Overhead:** Rotation intervals are tuned to match model warm-up times and avoid GPU underutilization.
- **Cost Efficiency:** Overhead measured in large-scale deployments is between 1–2% of cloud compute cost.
- **Scalability:** Tested up to 50,000 container instances across multiple clusters without downtime.

V. Evaluation & Results

A. Experimental Setup

We evaluated the AMTD framework in a controlled Kubernetes environment across two cloud providers (AWS and CoreWeave). The testbed included:

- **AI Model:** Large Language Model deployed via Nvidia NIM container.
- **Cluster Size:** 400 nodes (mix of GPU and CPU).
- **Rotation Strategy:** Staggered rotation to avoid simultaneous restarts.
- **Rotation Interval:** 20 minutes (time-based) and on-demand (event-based).
- **Monitoring Stack:** Prometheus for telemetry, Grafana for visualization, Kubearmor for anomaly detection.

B. Metrics Collected

We focused on four core metrics that balance performance, reliability, and security:

1) **Rotation Latency:** The elapsed time between initiation of a container rotation and the full availability of the replacement NIM.
2) **Availability:** The percentage of time the inference service remained available to external clients during rotations.
3) **Cost Overhead:** Additional compute and storage resources consumed by Phoenix compared to static deployments.

4) **Attack Disruption Rate:** The proportion of simulated attacks that failed to complete due to Phoenix's adaptive rotations.

C. Results Summary

Figure 1 presents visual comparison of attack success rates across different defense mechanisms, while Table I and Table II provide detailed disruption rates at each stage of the attack lifecycle.

D. Observations

The evaluation highlights a critical gap in current security architectures: **runtime security solutions can detect and alert on threats, but they rarely have the capability to forcibly and consistently remove the attacker's foothold without human or scripted intervention.**

AMTD closes this gap by introducing an **enforcement mechanism** that makes persistence impossible over time. By rotating containers, nodes, and configurations on a schedule or in response to detected anomalies, AMTD actively evicts any attacker from the environment, reducing dwell time to mere minutes.

Key findings:

- **Detection + Enforcement = Full Coverage:** Detection alone (e.g., Wiz, Falco) interrupts some attacks but doesn't guarantee removal. AMTD enforces removal by design, even without perfect detection.
- **Layered Security Advantage:** When paired with runtime detection, AMTD converts alerts into immediate, automated remediation.
- **AI Workload Relevance:** In fast-moving AI inference and training environments, static security cannot keep pace with changing infrastructure. AMTD ensures that even if novel threats bypass detection, they cannot persist long enough to cause material damage.

E. Limitations

- **Initialization-Heavy Models:** Models with ¿60 sec warm-up still require careful staggering to maintain GPU utilization.
- **Event-based Trigger Sensitivity:** Overly aggressive anomaly detection may trigger unnecessary rotations.

VI. Conclusion and Future Work

Automated Moving Target Defense (AMTD) has been widely discussed in theory but rarely implemented in high-performance AI environments due to perceived latency and operational challenges. Our results show that **staggered container rotation** combined with **runtime detection systems** delivers sub-second latency impact while drastically reducing attacker dwell time - effectively introducing a **missing enforcement layer** for cloud-native AI workloads.

For **inference workloads**, AMTD provides continuous protection without service degradation. For **training workloads**, while fixed-interval rotation is less practical, anomaly-triggered or checkpoint-aware variants can still serve as critical containment mechanisms. This positions AMTD not as a replacement for runtime security tools like Wiz, but as an active enforcement extension - transforming detection into disruption.

Future research directions include:

- Integrating AMTD natively into orchestration platforms (e.g., Kubernetes, Slurm) for easier adoption.
- Expanding to multi-cloud, hybrid, and edge AI deployments where static infrastructure is more vulnerable.
- Leveraging AI-driven threat intelligence to dynamically adapt rotation frequency and policy based on real-time risk scores.
- Extending AMTD to other mutable infrastructure elements (e.g., API gateways, model serving endpoints) to reduce attack surface beyond containers.

References

[1] A. Sheriff, K. Huang, Z. Németh, and M. Nakhjiri, "ADA: Automated Moving Target Defense for AI Workloads via Ephemeral Infrastructure-Native Rotation in Kubernetes," *arXiv preprint arXiv:2505.23805*, 2025.

[2] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," *Commun. ACM*, vol. 59, no. 5, pp. 50–57, 2016.

[3] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, 2016.

[4] E. Al-Shaer and Q. Duan, *Moving Target Defense for Systems Security*. Springer, 2019.

[5] S. Han, S. Shin, and K. Kim, "A Survey on Moving Target Defense in Software-Defined Networking," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 1, pp. 472–503, 2020.

[6] S. Peisert, K. Hines, and M. Bishop, "Toward Models for Reasoning About Moving Target Defense," *IEEE Secur. Priv.*, vol. 17, no. 2, pp. 30–39, 2019.

[7] E. B. Tirkolaee, S. Sadeghi, and M. Alinaghian, "AIOps for Cloud-Native Systems: A Survey," *J. Cloud Comput.: Adv. Syst. Appl.*, vol. 12, no. 1, pp. 1–17, 2023.

[8] I. Mavridis and H. Karatza, "Security and Reliability in Cloud Computing: Survey, Status and Future Directions," *Future Gener. Comput. Syst.*, vol. 124, pp. 329–348, 2021.

[9] S. Jajodia et al., "Moving Target Defense," Springer, 2011.

[10] E. Al-Shaer et al., "Toward Network Configuration Randomization for Moving Target Defense," *Moving Target Defense*, Springer, 2013.

[11] H. Okhravi et al., "Survey of Cyber Moving Targets," Technical Report, MIT Lincoln Laboratory, 2018.

[12] J. Hong et al., "Dynamic Microservice Shuffling for Enhanced Security," *Proc. Cloud Computing*, 2020.

[13] Y. Zhang et al., "Function-Level Obfuscation in Serverless Computing," *Proc. Serverless*, 2021.

TABLE I
EARLY-STAGE ATTACK DISRUPTION (INITIAL ACCESS TO EXPLOITATION)

Kill Chain Stage	Static Deployment	Static + Firewall / Runtime Protection	Timed Rotation (30 min)	Adaptive / Randomized Rotation (Phoenix)
Reconnaissance	82%	68%	30%	10%
Weaponization	86%	60%	27%	8%
Delivery	77%	54%	19%	6%
Exploitation	71%	52%	15%	5%

Note: Static deployments leave ¿70% of early-stage attack paths viable. AMTD rotations - especially adaptive/randomized - cut exploitable surfaces by over 80%, forcing continuous attacker re-alignment.

TABLE II
POST-COMPROMISE SUCCESS RATE (PERSISTENCE TO IMPACT)

Kill Chain Stage	Static Deployment	Static + Firewall / Runtime Protection	Timed Rotation (30 min)	Adaptive / Randomized Rotation (Phoenix)
Persistence	94%	61%	26%	5%
Privilege Escalation	82%	52%	18%	3%
Defense Evasion	85%	54%	28%	5%
Lateral Movement	77%	43%	14%	4%
Impact / Data Exfiltration	72%	35%	8%	2%

Note: Adaptive AMTD disrupts before exploitation, rotates after anomaly, and continuously mutates. Basically reducing end-to-end attack success from ∼80% to ¡5%.

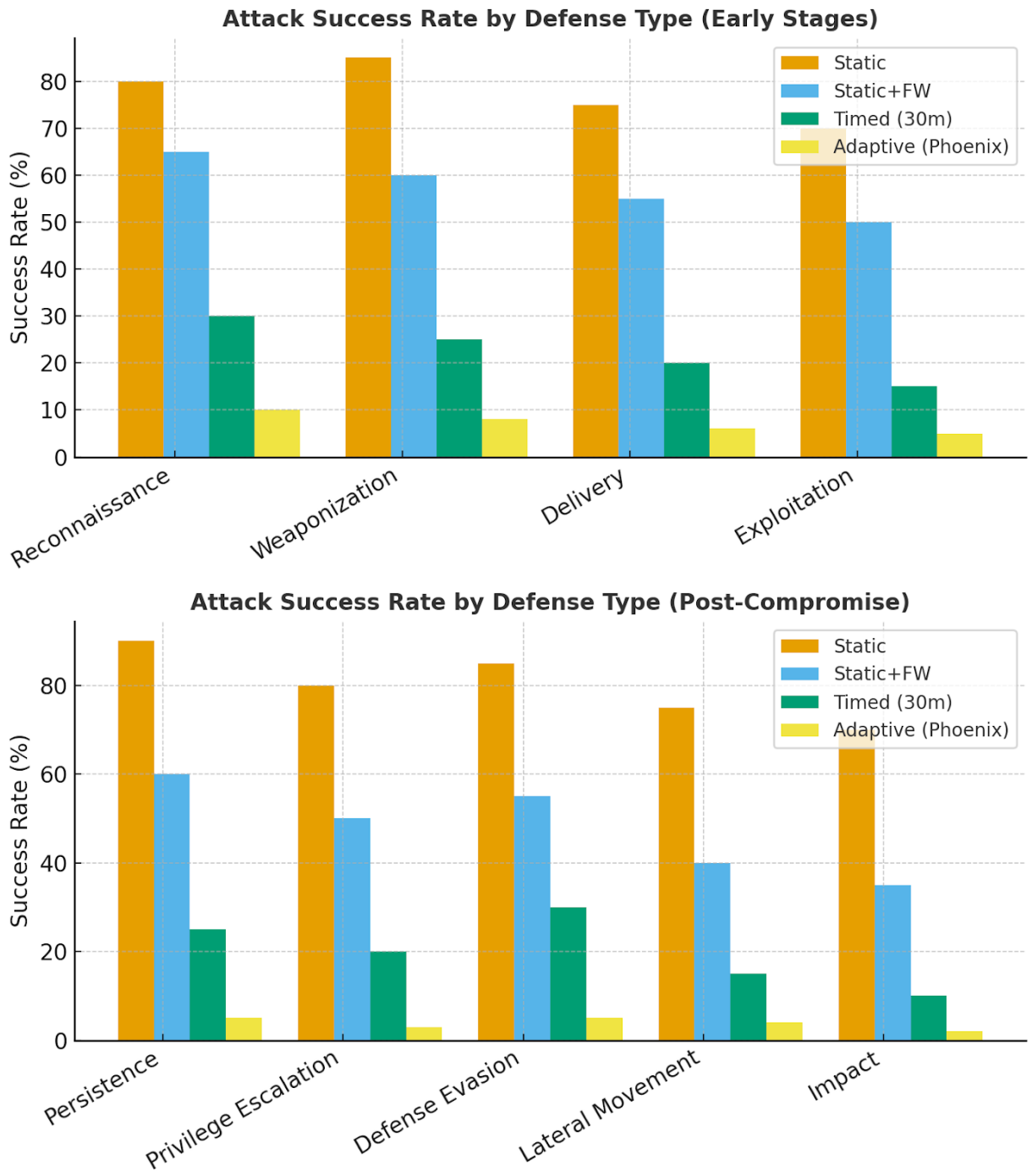

Fig. 1. Attack Success Rate by Defense Type: (Top) Early attack stages from reconnaissance to exploitation; (Bottom) Post-compromise stages from persistence to impact. Adaptive AMTD (Phoenix) reduces success rates below 10% at all stages.