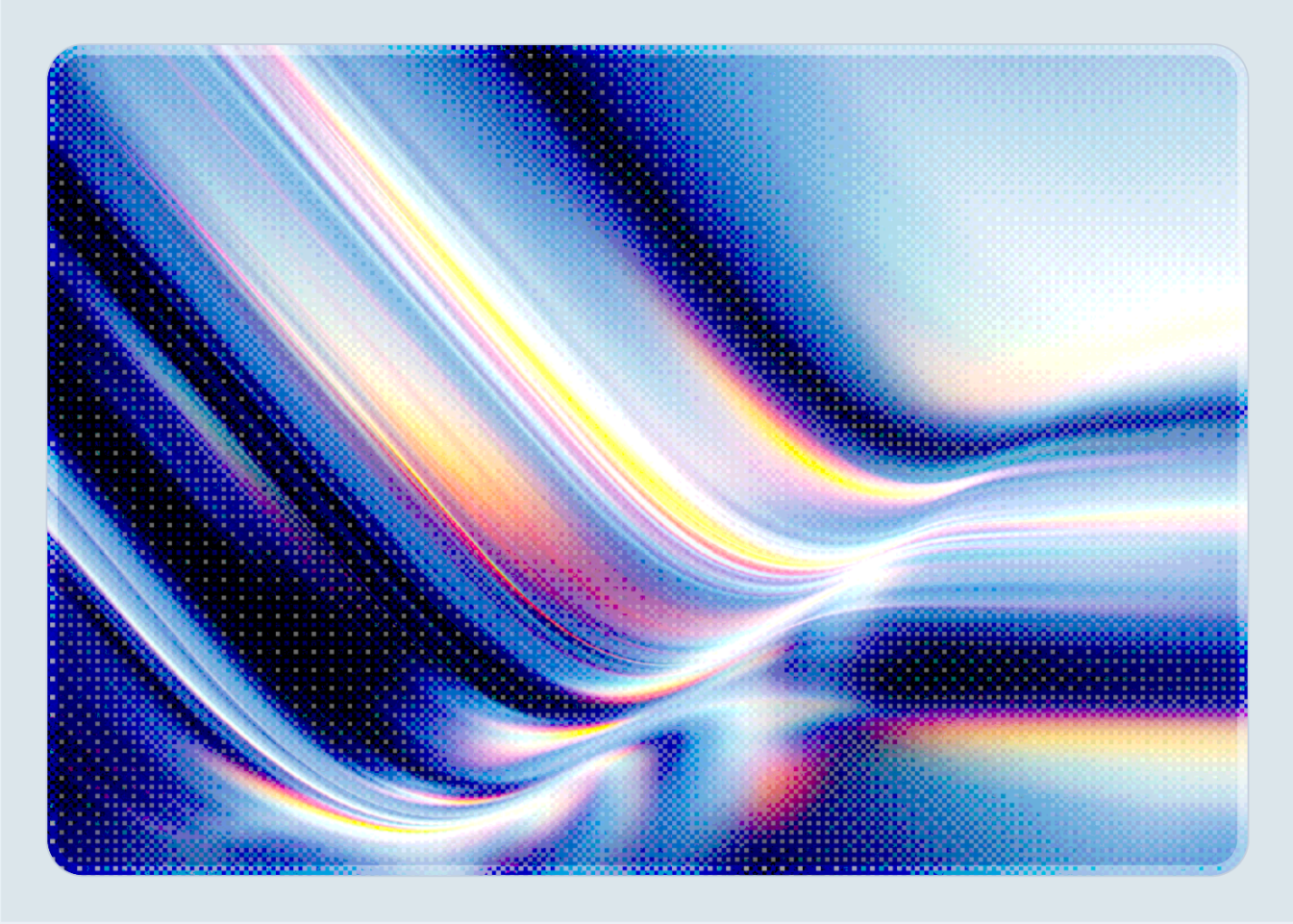
CSAI Foundation | Cloud Security Alliance

AI Inference Servers Are the New Attack Surface

LMDeploy CVE-2026-33626 and the Pattern Across vLLM, Triton, SGLang, and Ollama

2026-04-26

Unofficial AI-assisted Research



CSAI

CSA cloud security alliance®

Key Takeaways

- The exploitation of LMDeploy CVE-2026-33626 within 12 hours and 31 minutes (rounded in some reporting to 13 hours) of advisory publication on April 21–22, 2026 is the most recent data point in a year-long pattern of AI inference servers being weaponized within hours of disclosure, regardless of their install base [1][2].
- Across LMDeploy, vLLM, NVIDIA Triton, SGLang, Modular Max Server, Microsoft Sarathi-Serve, Meta Llama Stack, and Ollama, five architectural patterns account for the majority of recently disclosed vulnerabilities, with individual frameworks affected by varying subsets: external-fetch endpoints with no URL validation (SSRF), unsafe deserialization over inter-process communication, hardcoded trust boundaries that override operator security settings, missing default authentication on management interfaces, and memory-unsafe parsing of attacker-controlled multimodal inputs (see Security Analysis for per-framework citations).
- The "ShadowMQ" research published by Oligo Security in November 2025 documented more than a dozen named RCE-class CVEs propagated by copy-paste code reuse of a single unsafe ZeroMQ–pickle deserialization pattern across major AI inference frameworks, demonstrating that vulnerability classes in this ecosystem are systemic rather than per-product [3][4].
- Internet exposure is at scale: SentinelOne SentinelLABS and Censys identified approximately 175,000 publicly accessible Ollama hosts across 130 countries in early 2026 [5], while a Cisco Talos September 2025 case study identified roughly 1,139 publicly reachable Ollama instances at that earlier point in time [6][7] — figures that, while methodologically distinct, both point to substantial and growing internet exposure of inference infrastructure.
- Defending this attack surface requires treating inference servers as a distinct, internet-edge asset class subject to emergency patching SLAs, infrastructure-layer SSRF and egress controls, mandatory authentication on management APIs, and software composition analysis coverage equivalent to that applied to web application stacks.

Background

AI inference servers — the runtime systems that load model weights, accept prompt or multimodal inputs, and return generated outputs — have become a core layer of enterprise AI infrastructure over the past two years. Open-source frameworks now dominate this layer, with vLLM, NVIDIA Triton, Hugging Face Text Generation Inference, SGLang, Ray Serve, LMDeploy, Ollama, Microsoft Sarathi-Serve, Modular Max Server, and Meta's Llama Stack each occupying a portion of the deployment landscape. Many enterprises with mature generative AI workloads run more than one of these frameworks in production.

The category emerged rapidly. vLLM, originally a UC Berkeley research project, became a widely used reference inference engine for OpenAI-compatible APIs and is the engine behind a substantial share of internal and commercial deployments. Ollama positioned itself as the developer-friendly path to running open-weight models locally and on small servers, achieving wide adoption among individual developers, research teams, and enterprise pilots. NVIDIA Triton serves the high-performance, multi-framework segment in production GPU clusters. LMDeploy, developed by Shanghai AI Laboratory's InternLM project, focuses on multimodal and Chinese-origin frontier models. SGLang, derived in part from vLLM, targets structured generation and reasoning workloads.

This rapid emergence has had two consequences directly relevant to security. The vulnerability record indicates that capability development — multimodal inputs, distributed serving, agent and tool-calling integrations, and management APIs — outpaced security hardening in many of these frameworks, with adversarial-input threat modeling appearing to have been retrofitted rather than designed in from the outset. In parallel, the frameworks share substantial code and design patterns — both deliberately, through forks and adaptations, and accidentally, through copy-paste reuse of common idioms — which has caused vulnerability classes discovered in one framework to recur in many others.

CVE-2026-33626 in LMDeploy, exploited within hours of public disclosure on April 21, 2026, is best understood as an instance of this broader pattern rather than as a one-off bug. Cloud Security Alliance's prior research note on the LMDeploy incident documented the vulnerability and exploitation timeline in detail [1]. This note examines what the LMDeploy incident, taken together with the year of CVEs that preceded it, reveals about the AI inference server attack surface as a class.

Security Analysis

LMDeploy as Pattern Indicator

The LMDeploy SSRF flaw is simple in mechanism: the `load_image()` function in `lmdeploy/vl/utils.py` accepted any URL beginning with `http`, fetched it from the inference server, and returned the result, without any validation of whether the URL pointed to an internal service, a cloud metadata endpoint, or a link-local address [1][2]. The advisory was published on GitHub at 15:04 UTC on April 21; the first observed exploitation in Sysdig's honeypot fleet occurred at 03:35 UTC on April 22, 12 hours and 31 minutes later, with no public proof-of-concept code in circulation at the time [1][2]. The attacker's ten-request reconnaissance session targeted AWS Instance Metadata Service credentials, internal Redis and MySQL services, distributed inference control endpoints, and out-of-band DNS exfiltration.

What makes this vulnerability significant beyond its specific impact on LMDeploy users is how cleanly it instantiates each of the design patterns that recur across the inference server category. The vulnerable function fetches attacker-controlled URLs as part of normal multimodal request handling. The vulnerable parameter is structurally indistinguishable from legitimate input, defeating signature-based detection at the application layer. The vulnerable deployment is bound to `0.0.0.0` by default. The exploitable secondary surface — an unauthenticated `/distserve/p2p_drop_connect` endpoint — reflects the practice of exposing internal coordination interfaces without authentication, on the assumption that they will only ever be reached from within a trusted cluster network. Each of these patterns has parallels in other inference frameworks, and each has produced its own CVE. As the table below indicates, real vulnerabilities frequently combine more than one pattern: the LMDeploy `/distserve` endpoint, for instance, fits both the missing-authentication and the uncontrolled-internal-fetch categories.

Five Recurring Vulnerability Patterns

The vulnerabilities disclosed in AI inference servers over the last twelve months cluster into five architectural patterns. Understanding the patterns is more useful than tracking individual CVEs because the same patterns are likely to continue producing new vulnerabilities as new frameworks are written and as existing frameworks add new features.

The first pattern is **external-fetch endpoints without URL validation**, the SSRF class that LMDeploy CVE-2026-33626 exemplifies. Multimodal models accept image, video, and audio inputs, and the prevailing convention — originating with the OpenAI multimodal API — is to pass these inputs as URL

references that the inference server resolves at request time. Unless the framework explicitly validates resolved IP addresses against private and link-local ranges before connecting, this design coerces the inference server into acting as an HTTP proxy for the requester. The pattern recurs across frameworks: URL-based multimodal input transmission is convenient and widely adopted, while the URL validation logic that would prevent SSRF is easy to overlook in code reviews focused on model performance and inference latency.

The second pattern is **unsafe deserialization in inter-process and inter-node communication**. Production inference servers frequently use ZeroMQ, gRPC, or similar transports to coordinate between API frontends, scheduler processes, GPU worker processes, and distributed cluster nodes. When these channels carry serialized Python objects deserialized via `pickle` — directly or via wrappers like ZMQ's `recv_pyobj()` — any attacker who can write to that channel achieves arbitrary code execution. Oligo Security's "ShadowMQ" research, published in November 2025, identified more than a dozen named RCE-class CVEs matching this pattern across vLLM (CVE-2025-30165), NVIDIA TensorRT-LLM (CVE-2025-23254), Modular Max Server (CVE-2025-60455), Meta Llama Stack (CVE-2024-50050), Microsoft Sarathi-Serve, and SGLang, with the unsafe pattern often propagated by literal copy-paste — one vulnerable file in SGLang reportedly began with the comment "Adapted from vLLM" [3][4]. Researchers identified thousands of ZMQ sockets exposed unencrypted to the public internet, some clearly belonging to production inference servers [3].

The third pattern is **hardcoded trust boundaries that bypass operator security settings**. Several inference frameworks accept the `trust_remote_code` parameter from Hugging Face Transformers, which when enabled allows downloaded model repositories to execute arbitrary Python code at load time. This is by design when an operator enables it deliberately. However, vulnerabilities have been discovered in which the framework hardcodes `trust_remote_code=True` in specific code paths, silently overriding an operator's `--trust-remote-code=False` configuration. vLLM CVE-2026-27893, recorded in NVD with a CVSS score of 8.8, documented this in two model implementation files [9], and InstructLab CVE-2026-6859, disclosed on April 22, 2026 with a CVSS score of 8.8, documented the same flaw in the `linux_train.py` script [10]. The pattern is dangerous because it converts a security control the operator believes is in effect into a no-op, with no error or warning visible at runtime.

The fourth pattern is **missing or default-disabled authentication on management interfaces**. Ollama exposes its full model management API — including the ability to pull models, delete models, and submit inference requests — on port 11434 without authentication by default [5][7]. CVE-2025-51471 in Ollama allowed cross-domain token exposure via a malicious `WWW-Authenticate` realm header, and CVE-2025-63389 documented missing authentication on model management endpoints in versions through 0.13.5 [11][12]. The pattern is not unique to Ollama. LMDeploy's `/distserve` endpoints are

unauthenticated by default. NVIDIA Triton's HTTP and gRPC endpoints are unauthenticated by default and rely on operators to deploy network-layer access control. The result is a category of inference servers in which the entire administrative surface is reachable by anyone who can route to the listening port.

The fifth pattern is **memory-unsafe parsing of attacker-controlled multimodal inputs**. vLLM CVE-2026-22778, with a CVSS score of 9.8, illustrated this category through a chained exploit: an information disclosure flaw in vLLM's error handling leaked a heap address that reduced ASLR effectiveness from billions of possibilities to roughly eight, and a heap overflow in the JPEG2000 decoder bundled with OpenCV's FFmpeg dependency allowed remote code execution when triggered by a constructed video URL submitted to the multimodal endpoint [13][14][15]. The vulnerability affected vLLM versions 0.8.3 through 0.14.0 and was fixed in 0.14.1 [14]. NVIDIA Triton's August 2025 vulnerability chain (CVE-2025-23319, CVE-2025-23320, CVE-2025-23334) followed similar logic in shared-memory-region handling within the Python backend, allowing unauthenticated remote attackers to chain an information leak with memory access primitives to achieve full remote code execution [16] [17]. Because inference servers are increasingly designed to accept arbitrary uploaded media, and because they delegate parsing to large native dependencies (FFmpeg, OpenCV, libpng, JPEG2000 decoders) that are themselves complex and historically vulnerability-prone, this pattern is likely to persist as long as inference frameworks rely on these native dependencies for binary parsing, regardless of how thoroughly individual framework code is reviewed.

The table below consolidates the five patterns alongside representative CVEs and the affected frameworks. Real vulnerabilities frequently combine more than one pattern, so a per-framework checklist should treat the categories as overlapping rather than disjoint.

#	Pattern	Representative CVEs	Frameworks with Disclosed Instances
1	External-fetch endpoints without URL validation (SSRF)	CVE-2026-33626	LMDeploy [1][2]
2	Unsafe deserialization over IPC / inter-node channels	CVE-2025-30165, CVE-2025-23254, CVE-2025-60455, CVE-2024-50050	vLLM, TensorRT-LLM, Modular Max, Llama Stack, Sarathi-Serve, SGLang [3] [4]
3	Hardcoded trust boundaries overriding	CVE-2026-27893, CVE-2026-6859	vLLM, InstructLab [9][10]

#	Pattern	Representative CVEs	Frameworks with Disclosed Instances
	operator settings		
4	Missing or default-off authentication on management APIs	CVE-2025-51471, CVE-2025-63389	Ollama; LMDeploy `/distserve`; Triton (default) [5][7][11][12]
5	Memory-unsafe parsing of multimodal input	CVE-2026-22778, CVE-2025-23319/23320/23334	vLLM, NVIDIA Triton [13][14][15][16][17]

The Common Architectural Drivers

Three architectural choices, common to most AI inference servers, drive these patterns and explain why fixes in one framework do not automatically protect users of another. The first is the assumption of a trusted internal network. Most inference servers were designed to run inside a security perimeter — a Kubernetes cluster, a VPC, a research lab — and their authentication, authorization, and network-binding defaults reflect that assumption. When operators expose them directly to the public internet, as the Ollama exposure data shows is happening at scale, the defaults become an active source of risk to the operator.

The second is the use of Python and dynamic deserialization in the data plane. Python's `pickle` is convenient but unsafe; ZeroMQ's `recv_pyobj()` is convenient but a remote-code-execution wrapper for `pickle`. The inference frameworks adopted these patterns because they make distributed serving easy to implement; the result is that any compromise of, or unauthenticated access to, inter-process channels yields RCE rather than mere data exposure.

The third is the consumption of attacker-controlled URLs and binary blobs as a normal part of the request path. Multimodal model inputs are URLs and media files. Tool-calling models fetch external resources. Agent frameworks load external instructions. Each of these features expands the inference server's threat model from "answer questions about provided text" to "execute network operations and parse arbitrary binary formats on behalf of remote, unauthenticated requesters." Frameworks that have not retrofitted comprehensive validation onto these surfaces will continue to produce SSRF, RCE, and information disclosure vulnerabilities.

Exposure Scale and Attacker Economics

The volume of internet-exposed inference infrastructure is large and growing. SentinelOne SentinelLABS and Censys jointly identified approximately 175,000 publicly accessible Ollama hosts across 130 countries in research published in early 2026, with the United States and China accounting for the largest national shares [5]. A separate Cisco Talos case study published in September 2025 identified approximately 1,139 publicly reachable Ollama instances using Shodan-based dorking, of which roughly 20% were actively serving models without authentication [6][7][18]. The two studies use materially different methodologies — the SentinelLABS/Censys count reflects all hosts answering on Ollama's default port across global scanning, while the Cisco Talos study tracks instances meeting a narrower exposure profile — and the proportion of the larger 2026 population actively serving models without authentication has not been independently re-measured at the time of writing. Both studies, taken together, point to an installed base in the tens to hundreds of thousands of unauthenticated, internet-reachable inference servers running known-vulnerable software versions.

This scale changes attacker economics. When tens of thousands of unauthenticated servers running known-vulnerable software versions are addressable from the internet, advisory publication can function as a de facto exploit release for opportunistic adversaries, particularly when the advisory text describes the vulnerable code path in sufficient detail to enable LLM-assisted exploit construction against a large pool of unpatched, unauthenticated targets. The Sysdig honeypot observation that LMDeploy was attacked within 12 hours and 31 minutes of advisory publication, with no public proof-of-concept available, is a predictable outcome in this environment [1][2].

Recommendations

Immediate Actions

Organizations running any AI inference framework should within the next week complete an inventory that identifies every inference server in production, development, and shadow-IT use, with version numbers, network exposure, and authentication configuration recorded for each. Specific patches with publicly known exploitation should be prioritized: LMDeploy v0.12.3 or later for CVE-2026-33626 [1][2], vLLM 0.14.1 or later for CVE-2026-22778 and the related ShadowMQ-class CVEs [13][14], NVIDIA Triton 25.07 or later for the CVE-2025-23319 chain [16][17], current Ollama releases for the CVE-2025-51471 and CVE-2025-63389 authentication issues [11][12], and the latest InstructLab release for CVE-2026-6859 [10].

For inference workloads on AWS, IMDSv2 should be enforced with `httpTokens=required` on every instance hosting inference processes. The token-exchange requirement makes IMDS credentials inaccessible via the SSRF primitives that frameworks like LMDeploy expose, regardless of whether the application has been patched. Equivalent metadata-service hardening should be applied on GCP (disable legacy metadata endpoints) and Azure (instance-level firewall restrictions on IMDS access).

Inference servers exposed to the public internet without authentication should be removed from public exposure or placed behind an authenticating reverse proxy as an emergency measure. The Ollama exposure data indicates that this control is widely missing, and the documented exploitation timelines argue for treating it as a baseline rather than an enhancement.

Short-Term Mitigations

Network egress from inference workloads should be controlled through an allowlist proxy or VPC security group rules that explicitly deny outbound connections from inference processes to 169.254.0.0/16, 127.0.0.0/8, and RFC 1918 ranges except where specific, documented internal service communication is required. This is the most reliable defense against the SSRF class because it operates independently of the application and does not require distinguishing malicious URLs from legitimate ones at the request layer.

Inter-process and inter-node communication channels (ZeroMQ sockets, gRPC channels, distributed-serving control plane endpoints) should be bound to localhost or to private cluster networks only, never to `0.0.0.0`, and should require TLS and mutual authentication for any cross-host communication. Researchers identified thousands of ZMQ sockets exposed unencrypted to the public internet during the ShadowMQ investigation [3]; any production inference deployment should treat closing this exposure as a non-optional control.

Runtime detection should cover the behaviors that distinguish inference-server exploitation from normal traffic: outbound connections from inference processes to cloud metadata IP ranges, anomalous DNS resolution patterns from inference containers, and process-level execution of `python` or shell binaries as children of inference server processes. Sysdig has published Falco rules covering metadata service access from container workloads on AWS, GCP, and Azure [1]. CSPM and EDR tooling should incorporate equivalent detections.

Software composition analysis must include AI inference frameworks, their Python dependencies, and the native libraries those dependencies bundle (FFmpeg, OpenCV, image and video codecs). The vLLM CVE-2026-22778 chain demonstrated that the actual exploitable flaw lived in a transitively bundled FFmpeg version, not in vLLM source code. SCA configurations that ignore transitive native dependencies will miss this entire class of vulnerability.

Strategic Considerations

The compressed exploitation timelines for AI inference vulnerabilities argue for an emergency patching SLA framework specific to this asset class. Traditional 30-day SLAs for high-severity vulnerabilities are inadequate when published advisories produce in-the-wild exploitation in under 13 hours. Organizations should establish an AI inference vulnerability program with a maximum 24- to 48-hour patching SLA for any CVSS High or Critical disclosure affecting an inference framework in production use, with named owners, after-hours escalation paths, and automation infrastructure capable of meeting that SLA. Meeting that SLA requires on-call infrastructure capacity that not all organizations currently have for AI workloads, and standing up that capacity is itself a near-term initiative.

Selection criteria for AI inference frameworks should incorporate security architecture, not only performance and feature set. Organizations evaluating frameworks should ask whether the framework validates URLs against private IP ranges before fetching, whether it requires authentication on management and distributed-serving endpoints by default, whether it uses memory-safe alternatives to `pickle` for inter-process communication, whether it isolates `trust_remote_code` paths and refuses to override operator settings, and whether its security advisory cadence and quality are consistent with treating the framework as production infrastructure. Frameworks that fail multiple criteria warrant additional compensating controls — or substitution — before deployment in environments where any of those failure modes would matter.

The categorization of inference servers within enterprise security programs needs to change. The vulnerability record suggests that inference servers in many organizations are governed under data science or ML platform team policies rather than the production-infrastructure standards applied to web servers, API gateways, or databases — a categorization that the compressed exploitation timelines documented here render inadequate. Inference servers warrant the same architectural review, threat modeling, runtime monitoring, and patching discipline applied to other internet-edge production infrastructure. Until that categorization is applied broadly, the pattern documented in this note — rapid weaponization of advisory disclosures against an installed base that has not patched — will continue.

Finally, organizations should anticipate that the multimodal-input SSRF pattern will recur as new frameworks ship vision, video, and audio support, as agent frameworks add tool-calling and external-resource fetching, and as distributed serving expands its inter-node communication footprint. Treating this as a recurring class of risk — rather than as a series of unrelated bugs — enables earlier detection and a more durable defensive posture than reacting to each individual CVE.

CSA Resource Alignment

CSA's Multi-Agent Environment, Security, Threat, Risk, and Outcome (MAESTRO) framework provides layered threat modeling for agentic and inference-driven AI systems [19]. The vulnerability patterns in this note operate primarily at MAESTRO's deployment infrastructure and inter-agent communication layers. Threat modeling exercises using MAESTRO should include external-fetch primitives in multimodal inference paths, unsafe deserialization across inter-process channels, default-permissive authentication settings on management interfaces, and untrusted parsing of binary inputs as distinct threat categories with their own controls and detections.

The CSA AI Controls Matrix (AICM), the superset of the Cloud Controls Matrix tailored for AI systems, addresses the relevant control objectives across multiple domains [20]. Network controls cover egress restriction and private-range isolation for inference workloads. Identity and access management controls cover IMDS hardening and least-privilege IAM roles for inference instances. Application security controls cover input validation, including URL validation in multimodal request handling. Software supply chain controls cover transitive dependency analysis, the pathway through which the vLLM JPEG2000 vulnerability entered the inference data path. Organizations should map their inference deployments against AICM rather than treating inference as outside the scope of cloud security control frameworks.

The CSA STAR for AI program provides a structured assurance mechanism for organizations to assess and communicate AI infrastructure security posture [21]. The recurring pattern of inference-server vulnerabilities argues for STAR for AI assessments to include explicit verification of inference framework hardening, network architecture, authentication configuration, and patching SLAs as core attestation elements rather than optional considerations. Self-hosted inference deployments warrant the same level of assurance scrutiny as the model providers and cloud service providers that surround them.

CSA's Zero Trust guidance applies directly to inference server architecture. Assume-breach principles, mutual authentication for all inter-service communication, least-privilege network access, and continuous verification of identity and posture at each connection are the controls that contain inference server compromise even when individual frameworks contain vulnerabilities. An inference node whose egress is allowlist-controlled cannot exfiltrate IMDS credentials. A distributed-serving channel that requires mutual TLS cannot be exploited by an attacker who reaches the network port. Zero Trust network architecture for AI inference is an immediately applicable control set with documented effectiveness against the specific attack techniques observed across the LMDeploy, vLLM, Triton, and ShadowMQ incidents.

References

[1] Sysdig Threat Research Team. "CVE-2026-33626: How Attackers Exploited LMDeploy LLM Inference Engines in 12 Hours." Sysdig, April 2026.

[2] The Hacker News. "LMDeploy CVE-2026-33626 Flaw Exploited Within 13 Hours of Disclosure." The Hacker News, April 24, 2026.

[3] Oligo Security. "ShadowMQ: How Code Reuse Spread Critical Vulnerabilities Across the AI Ecosystem." Oligo Security, November 2025.

[4] The Hacker News. "Researchers Find Serious AI Bugs Exposing Meta, Nvidia, and Microsoft Inference Frameworks." The Hacker News, November 2025.

[5] The Hacker News. "Researchers Find 175,000 Publicly Exposed Ollama AI Servers Across 130 Countries." The Hacker News, January 2026.

[6] BankInfoSecurity. "Exposed LLM Servers Expose Ollama Risks." BankInfoSecurity, 2026.

[7] Cisco Talos / Cisco Blogs. "Detecting Exposed LLM Servers: A Shodan Case Study on Ollama." Cisco Blogs, September 2025.

[8] vLLM Project. "vLLM: Easy, Fast, and Cheap LLM Serving with PagedAttention." GitHub, 2026.

[9] National Vulnerability Database. "CVE-2026-27893: vLLM Hardcoded `trust_remote_code=True` in Model Implementation Files." NIST NVD, 2026.

[10] Tenable. "CVE-2026-6859: InstructLab `linux_train.py` Hardcoded `trust_remote_code` Enables RCE via Malicious Hugging Face Models." Tenable, April 2026.

[11] GitHub Advisory Database. "Ollama Vulnerable to Cross-Domain Token Exposure – CVE-2025-51471." GitHub, 2025.

[12] GitHub Advisory Database. "Ollama Platform Has Missing Authentication Enabling Attackers to Perform Model Management Operations – CVE-2025-63389." GitHub, 2025.

[13] OX Security. "Millions of AI Servers at Risk: Critical vLLM RCE Lets Attackers Take Over via Video Link (CVE-2026-22778)." OX Security, 2026.

[14] Orca Security. "Critical RCE in vLLM Allows Server Takeover via Malicious Video URL (CVE-2026-22778)." Orca Security, 2026.

[15] Kodem Security. "CVE-2026-22778: Critical Remote Code Execution in vLLM Multimodal Inference." Kodem Security, 2026.

[16] Wiz. "Breaking NVIDIA Triton: CVE-2025-23319 – A Vulnerability Chain Leading to AI Server Takeover." Wiz Blog, August 2025.

[17] The Hacker News. "NVIDIA Triton Bugs Let Unauthenticated Attackers Execute Code and Hijack AI Servers." The Hacker News, August 2025.

[18] Cyberpress. "Over 1,100 Ollama AI Servers Exposed Online, 20% Vulnerable." Cyberpress, September 2025.

[19] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA, February 2025.

[20] Cloud Security Alliance. "AI Controls Matrix (AICM)." CSA, 2025.

[21] Cloud Security Alliance. "CSA STAR for AI." CSA, 2026.