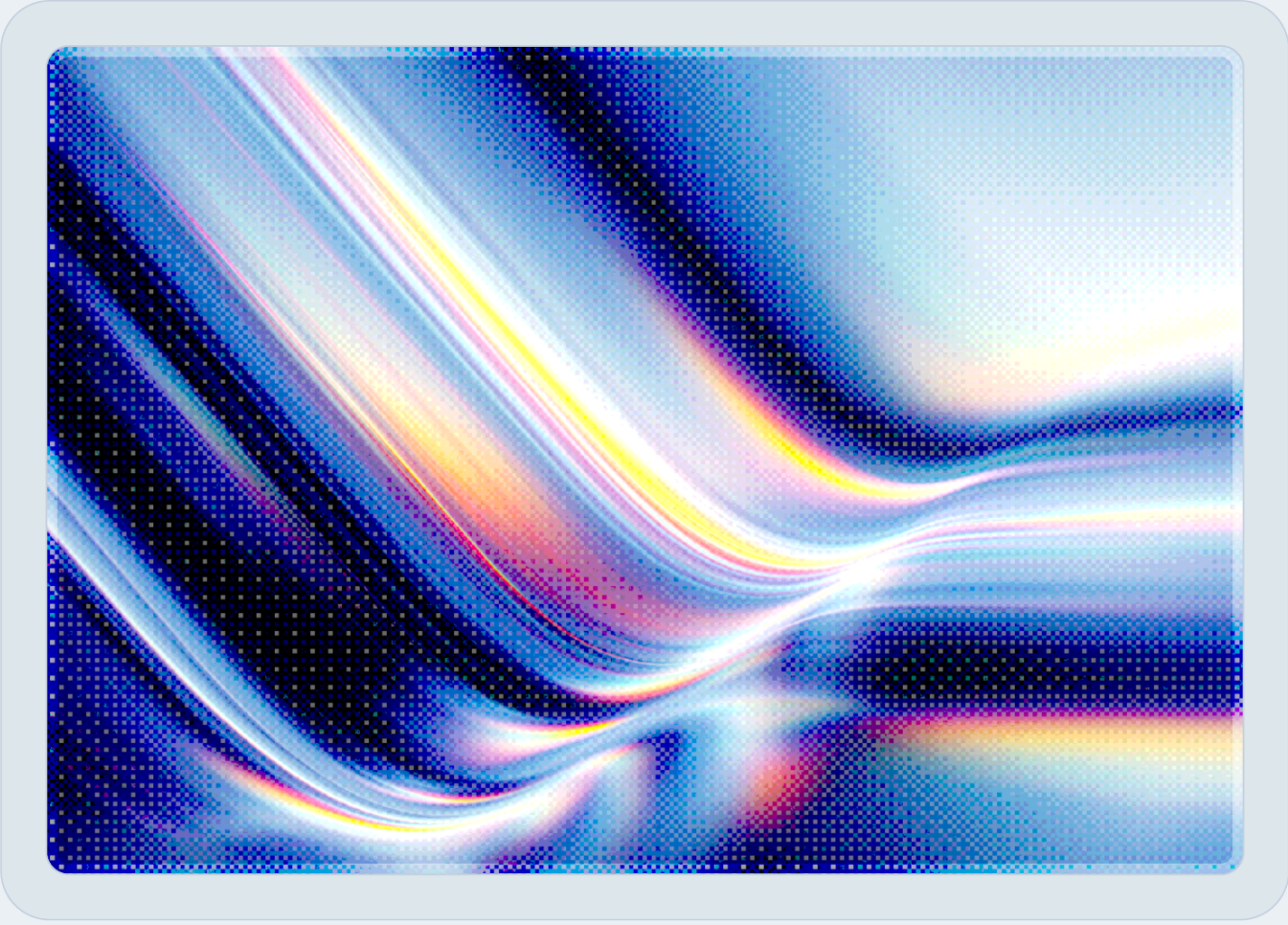
CSAI Foundation | Cloud Security Alliance

Bleeding Llama: Remote Memory Exfiltration in Self-Hosted LLM Infrastructure

2026-05-12

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- CVE-2026-7482, dubbed "Bleeding Llama" by Cyera Research, is a critical (CVSS 9.1) heap out-of-bounds read vulnerability in Ollama that enables unauthenticated remote attackers to exfiltrate arbitrary process memory from any exposed server [1].
- Exploitation requires only three unauthenticated API calls against Ollama's model creation and push endpoints; the low attack complexity—as reflected in the CVSS 9.1 vector—places successful attacks within reach of unsophisticated threat actors [1][2].
- Leaked memory may contain system prompts, user conversation fragments, environment variables, API keys, cloud credentials, and sensitive code—any data resident in the Ollama process heap at the time of exploitation [1][2][3].
- Approximately 300,000 Ollama instances were identified as publicly accessible on the internet at the time of disclosure, and the platform's default configurations provide no authentication layer to limit access [1].
- Organizations must update immediately to Ollama 0.17.1, restrict API access to authenticated networks, and audit all deployments for unintended internet exposure [2][4].
- This vulnerability is consistent with a broader pattern observed across self-hosted AI tooling, in which developer-convenience tools ship without the network isolation and access controls required for production environments handling sensitive data.

Background

Ollama is an open-source platform that enables individuals and organizations to download, run, and manage large language models locally or on private infrastructure. It provides a REST API that applications can use to interact with models—enabling capabilities such as chatbots, coding assistants, and document analysis pipelines—without relying on external cloud providers. As organizations have grown more attentive to data privacy and cost concerns around commercial AI APIs, Ollama and similar self-hosting frameworks have seen substantial adoption across development teams and enterprise environments alike.

A core feature of Ollama is its support for the GGUF file format, a binary container used to distribute quantized LLM weights. GGUF files encode a series of tensors—multi-dimensional arrays of numerical weight data—along with metadata that describes each tensor's shape, data type, and position within the file. When Ollama receives a GGUF model, it parses the tensor metadata and processes the weight data accordingly, including converting between quantization formats such as 16-bit float (F16) and 32-bit float (F32). This quantization pipeline sits at the heart of the vulnerability.

The convenience Ollama affords comes with a security tradeoff that official deployment guides and community tutorials do not prominently surface as a prerequisite for shared-network deployments. By default, Ollama's REST API presents no authentication mechanism, and the common deployment configuration `OLLAMA_HOST=0.0.0.0`—widely documented in community tutorials—instructs the server to bind to all available network interfaces [4]. This architecture may be acceptable in isolated single-developer environments, but it creates serious exposure when adopted in shared, networked, or cloud-hosted infrastructure without additional controls. Security teams evaluating AI tooling should treat the absence of default authentication as a primary risk indicator, not a secondary configuration concern.

CVE-2026-7482 was reported to Ollama on February 2, 2026 by researchers at Cyera [1]. Ollama acknowledged the report and provided a corrective pull request on February 25, 2026. The CVE was formally published on May 1, 2026, with full technical disclosure on May 5, 2026 [1].

Security Analysis

The Vulnerability Mechanism

The root cause of CVE-2026-7482 is a failure to validate that the number of tensor elements declared in a GGUF file's metadata corresponds to the actual number of bytes present in the file. The affected code paths reside in Ollama's `fs/ggml/gguf.go` and `server/quantization.go` modules [3]. Because the GGUF format allows the tensor shape to be specified entirely by the file's author, a malicious actor can set those dimensions to arbitrarily large values. Ollama accepts the declared shape at face value and uses it to drive the quantization conversion loop. As Cyera's researchers noted, "there's no validation that the number of elements we're about to read actually matches the real size" [1].

When processing a GGUF file through the `/api/create` endpoint, Ollama performs an F16-to-F32 conversion for tensor data. If the declared element count is inflated far beyond the actual data in the file, the conversion loop reads past the end of the allocated buffer into adjacent heap memory [1]. This is classified under CWE-125, Out-of-bounds Read, reflecting the fundamental nature of the defect: the server trusts external input to bound a memory operation [3].

The F16-to-F32 conversion makes this particularly damaging from an intelligence-gathering standpoint. Because the operation preserves the raw bit pattern of each 16-bit float as it widens it to 32 bits, leaked heap memory arrives at the attacker in a form that can be decoded without loss. Whatever occupies the heap adjacent to the model buffer—string data from prior API requests, fragments of deserialized JSON bodies, environment variable contents, or active conversation state—becomes embedded in the processed tensor data and is subsequently available for exfiltration [1].

Exploitation in Practice

Exploitation of CVE-2026-7482 requires three sequential, unauthenticated API calls against a target Ollama instance [1][2]. First, the attacker uploads a malformed GGUF blob to the `/api/blobs/sha256:[hash]` endpoint. The file appears structurally valid but declares tensor dimensions whose element count substantially exceeds the file's actual payload. Second, the attacker issues a model creation request to `/api/create`, referencing the malicious blob. Ollama processes the GGUF file and runs the out-of-bounds quantization pass, embedding adjacent heap memory into the resulting model artifact. Third, the attacker invokes the `/api/push` endpoint to push the newly created model to an external registry under their control. The push operation transmits the model artifact —now containing the leaked heap contents—to the attacker's server, completing the exfiltration.

No authentication token, session cookie, or network credential is required at any stage on a default Ollama installation. Based on the passive nature of the exploit—a standard GGUF ingestion and push sequence rather than a denial-of-service vector—the attack is expected to complete without server disruption or obvious error output, though operational timing varies with file size and network conditions. Organizations relying solely on operational anomaly detection to identify intrusion have limited visibility into this attack chain without specific egress monitoring for unexpected model push operations.

Data Exposure Profile

The categories of data that Bleeding Llama can expose are determined by what the Ollama process holds in heap memory at the time of exploitation. In production and development environments, this typically includes system prompts defining the behavior of deployed models; conversation fragments from users sharing the same instance; environment variables, which in many deployments carry API keys for downstream AI services, database connection strings, and cloud provider credentials; source code or documents flowing through inference pipelines; and personally identifiable information or protected health information processed by the model [1][2][3].

The practical severity of each exposure category depends on what the target organization routes through Ollama. For a development team using Ollama for internal experimentation with non-sensitive data, the blast radius may be limited. For an enterprise deployment where Ollama mediates access to a document analysis or customer-facing AI pipeline, the leaked memory could include proprietary trade secrets, patient records, or credentials granting persistent access to downstream production systems. Organizations in regulated industries—healthcare, finance, legal—face the additional dimension of regulatory exposure when PHI or PII is involved, independent of the direct operational harm.

Scale of Exposure

At the time of public disclosure, researchers identified approximately 300,000 Ollama instances reachable from the public internet [1]. This figure reflects both the scale of Ollama's adoption and the frequency with which operators bind the service to all interfaces in cloud-hosted environments without compensating network controls. Because the platform ships without default authentication, the entire population of internet-exposed instances was potentially exploitable with no prerequisite beyond network reachability—no credential theft, no phishing campaign, no supply-chain compromise required.

Concurrent Vulnerabilities in the Ollama Ecosystem

The Bleeding Llama disclosure occurred alongside other recently identified Ollama security issues that compound the risk profile for Windows deployments. Researchers disclosed two vulnerabilities in Ollama's Windows automatic update mechanism: CVE-2026-42248, a missing signature verification flaw that allows an attacker to substitute an arbitrary binary for the legitimate update payload, and CVE-2026-42249, a path traversal vulnerability arising from unsanitized HTTP response headers used to construct the update staging directory [5]. Both carry a CVSS score of 7.7 [10] and, according to the disclosing researchers, can be chained to achieve persistent remote code execution on affected Windows hosts. The disclosing researchers have stated they will publish full technical details if no vendor patch is issued within 90 days of the May 5, 2026 disclosure [10]. Organizations running Ollama on Windows should treat these vulnerabilities as significantly elevating their aggregate risk posture beyond what the individual CVE scores suggest.

Systemic Implications for the AI Infrastructure Ecosystem

The conditions enabling Bleeding Llama—no default authentication, no input validation on externally supplied model files, and an API endpoint that initiates outbound connections to attacker-controlled servers—reflect the design philosophy of a tool built to minimize friction for individual developers. This

philosophy is appropriate for its original use case but becomes a structural liability when organizations adopt developer tooling for production AI workloads without rearchitecting the security posture.

This cycle has precedents in enterprise IT history: MongoDB's default no-authentication configuration contributed to mass data exposures beginning in 2017, and Redis and Elasticsearch followed similar patterns, with large numbers of internet-exposed instances running without access controls until community-driven hardening guidance and default-configuration changes were introduced. In each case, tools optimized for developer ease required explicit security hardening before production deployment. The LLM serving ecosystem is now traversing this cycle at speed, with Ollama representing one of the higher-profile examples. Security teams should treat Bleeding Llama as a leading indicator of similar vulnerabilities across other self-hosted model serving frameworks and develop proactive assessment processes accordingly.

Recommendations

Immediate Actions

Organizations running any version of Ollama prior to 0.17.1 should apply the patch as their highest-priority response. The upgrade addresses the GGUF tensor validation failure at the core of the vulnerability and should be treated with the urgency appropriate to a CVSS 9.1 flaw with a proven, low-complexity exploit chain [1][3].

Simultaneously, operators should audit all Ollama deployments for internet exposure. Any instance reachable from outside a trusted network boundary should be isolated pending investigation and treated as potentially compromised. Network asset discovery tools, including runZero's software inventory query for Ollama deployments, can accelerate identification of exposed instances within managed infrastructure [4]. For instances that cannot be immediately patched, blocking external access to the `/api/create` and `/api/push` endpoints at the network layer eliminates the specific attack chain while remediation proceeds.

Short-Term Mitigations

No Ollama instance handling sensitive data should be directly accessible from an untrusted network without an authentication layer. Organizations should deploy a reverse proxy or API gateway—such as NGINX, Traefik, or a cloud-provider API management service—with strong authentication (API key or

mutual TLS) enforced before traffic reaches Ollama's listener. This control would have closed the unauthenticated remote attack vector for CVE-2026-7482 and provides broad protection against future vulnerabilities in the same class.

Network segmentation should be applied so that Ollama hosts receive connections only from known, authorized application servers. The principle of least privilege applies to network topology as directly as it does to system accounts: Ollama instances should communicate only with the specific services required for their function. Egress policies should restrict Ollama hosts from initiating outbound model push operations to arbitrary external registries; legitimate workflows can be accommodated through explicit allowlisting, while unexpected push traffic to unknown endpoints becomes a detectable anomaly.

For Windows deployments, organizations should disable or sandbox Ollama's automatic update mechanism and manually verify binary integrity until CVE-2026-42248 and CVE-2026-42249 receive patches [5]. Applying application allowlisting controls around the Ollama installation directory reduces the risk from an unsigned update payload being substituted and executed.

Strategic Considerations

Bleeding Llama should serve as a catalyst for organizations to establish formal inventory and lifecycle management processes for all LLM serving components, applying the same rigor used for application servers and databases. This includes version tracking, patch cadence requirements aligned to the criticality of hosted data, network exposure auditing on a recurring basis, and data classification reviews to determine what categories of information are permissible inputs to self-hosted inference systems.

Organizations deploying Ollama in multi-tenant or shared-infrastructure environments face heightened risk because conversation data from concurrent users may intermingle in the process heap. Where data isolation between users or customers is a compliance requirement—such as HIPAA physical and technical safeguard obligations, GDPR data minimization principles, or contractual data separation clauses—a shared Ollama instance may violate that requirement independent of the current CVE. Architectural patterns that run per-tenant inference processes, use process-isolated serving components, or impose strict input sanitization and output filtering at the API gateway layer should be evaluated as compliant alternatives.

Procurement and evaluation criteria for AI infrastructure tooling should incorporate explicit security assessments: Does the tool have a defined vulnerability disclosure process and a published patch cadence? Does it enforce authentication by default? Are model file inputs validated against declared metadata before processing? Applying these questions at adoption time is substantially less expensive than retrofitting security controls after a tool has become embedded in production pipelines.

CSA Resource Alignment

The Bleeding Llama vulnerability maps directly to concerns addressed across several CSA frameworks and guidance documents, providing practitioners with a structured lens for response and longer-term architectural improvement.

CSA's MAESTRO framework for agentic AI threat modeling decomposes AI system risk across a seven-layer architecture, including Foundation Models, Data Operations, Agent Frameworks, Deployment and Infrastructure, Evaluation and Observability, Security and Compliance, and the Agent Ecosystem [6]. Under the MAESTRO framework, the out-of-bounds heap read in Ollama's quantization pipeline would be classified as a Layer 4 (Deployment and Infrastructure) threat: a flaw in the model serving stack that enables data extraction without compromising the model itself. Organizations applying MAESTRO should ensure that infrastructure-layer threat analysis explicitly covers model ingestion APIs and their input validation properties, not only prompt-layer and model-layer risks.

CSA's AI Controls Matrix (AICM) provides 243 control objectives across 18 domains governing AI system security [7]. Controls in the Model Security and AI Supply Chain domains are directly applicable: the AICM requires that organizations govern third-party model file ingestion, enforce API endpoint authentication requirements, and maintain egress monitoring for AI serving infrastructure. Organizations seeking a structured gap assessment should evaluate their AICM implementation against these domains in light of the Ollama exposure pattern.

CSA's guidance on Using Zero Trust to Secure Enterprise Information in LLM Environments provides an applicable architectural blueprint for reducing the attack surface that Bleeding Llama exploited [8]. The vulnerability is fundamentally a failure of the Zero Trust principle that no request should be trusted implicitly: an unauthenticated remote party's request to ingest an arbitrary model file was processed without verification of the file's integrity or the requester's authorization. Applying Zero Trust network access controls, strong API authentication, and continuous traffic monitoring around Ollama deployments would have materially prevented exploitation.

CSA's Cloud Controls Matrix (CCM) v4.1 addresses infrastructure security controls applicable to AI serving nodes as they would be to any cloud-hosted application workload [9]. CCM domains covering infrastructure vulnerability management, encryption key management, and network security all apply directly to the Ollama deployment pattern. Organizations with CCM-aligned compliance programs should map their self-hosted AI infrastructure into their existing CCM assessment scope rather than treating it as a separate, unreviewed category.

References

[1] Cyera Research. "Bleeding Llama: Critical Unauthenticated Memory Leak in Ollama." Cyera, May 2026.

[2] SecurityWeek. "Critical Bug Could Expose 300,000 Ollama Deployments to Information Theft." SecurityWeek, May 2026.

[3] CVEFeed. "CVE-2026-7482 – Ollama heap out-of-bounds read in GGUF tensor parsing leaks server process memory to unauthenticated remote attackers." CVEFeed, May 2026.

[4] runZero. "Ollama vulnerability CVE-2026-7482: Find impacted assets." runZero, May 2026.

[5] Help Net Security. "Unpatched flaws turn Ollama's auto-updater into a persistent RCE vector, researchers say." Help Net Security, May 2026.

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

[7] Cloud Security Alliance. "AI Controls Matrix." Cloud Security Alliance, 2025.

[8] Cloud Security Alliance. "Using Zero Trust to Secure Enterprise Information in LLM Environments." Cloud Security Alliance, 2026.

[9] Cloud Security Alliance. "Cloud Controls Matrix v4.1." Cloud Security Alliance, 2026.

[10] CERT Polska. "CVE-2026-42248." CERT Polska, April 2026.

