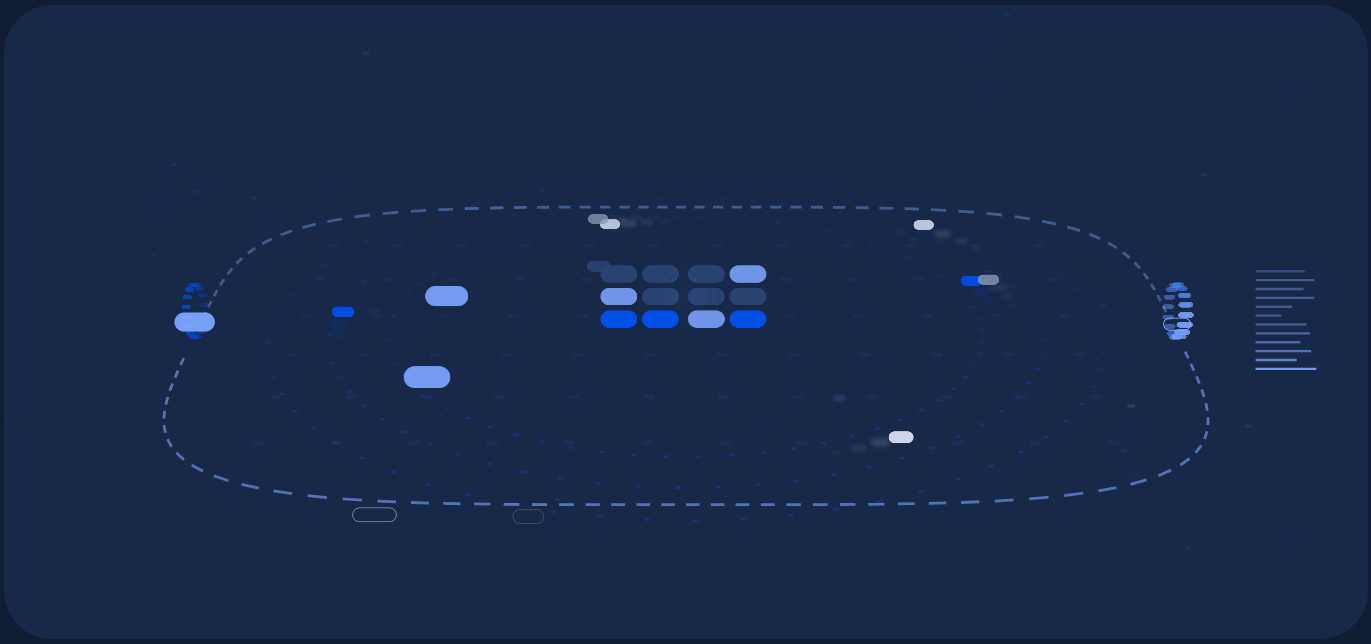
Rillor

EXPLAINER

Security risks in self-hosted LLM deployments

Self-hosted LLM security in six documented failure modes: exposed inference servers, serving software flaws, model files that run code, shared KV caches, poisoned documents and GPU container escapes.



12 November 2026 Private SI cloud rillor.com/insights/self-hosted-llm-security-risks rillor.com

Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

Running an LLM on infrastructure you control brings the inference server, the model files, the prompt cache, the retrieval index and the GPU container layer inside your own boundary, and each of them has documented failures and published controls [1, 7, 10, 17, 21, 27].

- Exposed inference servers are common enough to count. Cisco Talos reported 1,139 exposed Ollama servers on 1 September 2025, 214 of them responding with live models, and Censys reported about 10,600 exposed Ollama instances on 24 September 2025 [1, 2].
- Serving software carries critical flaws, such as a path traversal in Ollama's model pull that could lead to remote code execution, and unsafe loading of Python pickle data over unauthenticated ZeroMQ network sockets, code that Oligo Security found copied between inference frameworks [5, 6, 8, 9].
- Model files in Python's pickle format can run code when loaded. JFrog found around 100 models with harmful payloads on the Hugging Face Hub in February 2024, and in 2025 two more passed the Hub's scanner unflagged [11, 12].
- Sharing the key-value (KV) cache of processed prompt tokens across tenants can leak prompts. An NDSS 2025 attack fully or partially reversed prompt inputs with a 99% average success rate in its tests, and vLLM's defense, a per-tenant salt so only requests with the same salt share cached prefixes, is opt-in [7, 17].
- Five planted texts per target question gave the PoisonedRAG attack a 90% success rate against a knowledge database of millions of texts, and a 2023 method recovered 92% of short text inputs exactly from their embeddings [21, 22].
- Two NVIDIA Container Toolkit flaws that NVIDIA rated critical, CVE-2024-0132 and CVE-2025-23266, let a crafted container image reach its host. The first spared the Container Device Interface (CDI) mode and the second did not, and both were fixed in toolkit releases [27, 30, 31].
- Each failure maps to the OWASP Top 10 for LLM Applications 2025 and to NIST AI 100-2 E2025, and the sources name controls: authenticated endpoints, isolated node networks, safetensors files or weights-only loading, per-tenant cache salts, permission-aware vector stores and patched toolkits [1, 7, 10, 13, 25, 26, 33].

Background

What changes when you run the model yourself

A self-hosted deployment runs every part of the serving stack itself: the HTTP server that accepts prompts, the engine and the traffic between its nodes, the model files and any code shipped with them, the KV cache, the retrieval index and the container runtime that gives workloads their GPUs. NIST's 2025 taxonomy of adversarial machine learning notes that such systems inherit many weaknesses of the traditional software supply chain, such as reliance on third-party dependencies, and advises addressing security "by considering the full attack surface, including data and model supply chains, software, and network and storage systems" [26].

Several components document defaults that suit one trusted machine or network. Ollama binds to 127.0.0.1 on port 11434 by default [4], and in 2024 Wiz noted that it "does not support authentication out-of-the-box" [5]. vLLM states that all communication between the nodes of a multi-node deployment is "insecure by default and must be protected by placing the nodes on an isolated network" [7]. Its prefix cache is shared across all requests unless a cache salt is passed [7].

The vocabulary used here

The OWASP Top 10 for LLM Applications 2025 names ten risks: LLM01 Prompt Injection, LLM02 Sensitive Information Disclosure, LLM03 Supply Chain, LLM04 Data and Model Poisoning, LLM05 Improper Output Handling, LLM06 Excessive Agency, LLM07 System Prompt Leakage, LLM08 Vector and Embedding Weaknesses, LLM09 Misinformation and LLM10 Unbounded Consumption [23]. NIST AI 100-2 E2025, published in March 2025, sorts attacks on generative models by the attacker's goal: availability breakdown (NISTAML.01), integrity violation (NISTAML.02), privacy compromise (NISTAML.03) and misuse enablement (NISTAML.04) [26]. Its index also lists supply chain attacks (NISTAML.05), model poisoning (NISTAML.051), indirect prompt injection (NISTAML.015) and prompt injection (NISTAML.018) [26]. Finding 7 places each failure mode in both schemes.

This report is for platform engineers, ML engineers and security leads who run models on infrastructure they control, or plan to. It leaves out tool permissions for agents, which OWASP files under LLM06 Excessive Agency, and methods for testing agents against prompt injection.

Method

What was examined. Six failure modes of self-hosted LLM stacks: exposed inference endpoints, vulnerabilities in serving software, model files that execute code, KV caches shared between tenants, poisoned documents in retrieval indexes, and the GPU container layer. For each, the documented cases, the conditions they need and the controls the sources give.

Sources. Internet scans and honeypot data, vulnerability write-ups and advisories, NVD records, project and vendor documentation, attack papers and the OWASP and NIST taxonomies, 36 in all.

How. CVE descriptions, publication dates and scores come from NVD records read through the NVD API, with the scoring organization named, because one CVE can carry different scores from NVD and from the organization that reported it. NVIDIA's security bulletin pages refused automated access, so NVIDIA's position comes from its GitHub advisory and its release notes [33, 36]. Scan counts are given with their dates and methods and never added together. Attack mechanics are described only as far as each source publishes them. Model families are not named. Every page was viewed on 8 November 2026, and [7] again on 11 November after its update. The mappings in Findings 7 and 8 are this report's own.

Findings

1. Exposed inference endpoints

Scans. Cisco Talos used Shodan to find Ollama servers reachable from the internet and sent each a test prompt. A reply with HTTP 200 and the correct answer "indicates that the server is accepting and executing prompts without requiring authentication" [1]. On 1 September 2025 Talos reported 1,139 vulnerable Ollama instances [1]. Of these, 214 were responding with live models, about 18.8% [1]. The remaining servers, about 80%, were reachable through unauthenticated interfaces with no model loaded, and Talos describes them as still "susceptible to exploitation via unauthorized model uploads or configuration manipulation" [1]. Of the exposed servers, 88.89% followed a common chat-completions route structure [1]. Talos notes that its method targets default and common ports such as 11434, which biases it toward standard configurations [1].

In a post dated 24 September 2025, Censys reported 10.6K high-confidence Ollama instances exposed to the internet, about 10,600 services across 1,229 autonomous systems [2]. Over 25% ran on a port other than the default, some on ports close to it such as 11435 and 11439, which Censys suspects is "an attempt at security through obscurity" [2]. Of the 10.6K hosts, 1.5K answered at least one of two test prompts and 1,122 answered both [2]. The Talos and Censys figures come from separate scans with different methods and coverage, and are not added together here.

How exposure happens. Ollama binds to 127.0.0.1 on port 11434 by default [4]. Its documentation shows how to listen on all interfaces with the OLLAMA_HOST variable, and gives a reverse proxy example [4]. Wiz wrote that the default Linux installation binds to localhost, while Docker installations ran the server with root privileges and listened on 0.0.0.0 by default [5].

vLLM documents gaps of its own. Its --api-key flag protects only endpoints under four path prefixes, and "many other sensitive endpoints are exposed on the same HTTP server without any authentication enforcement" [7]. The examples include /invocations, which "routes to the same inference functions as /v1 endpoints", /pause, which "causes denial of service", /update_weights, which "can alter model behavior", and /metrics [7]. The documentation says "Do not rely exclusively on --api-key for securing access to vLLM" [7]. It warns never to enable development mode in production, says dynamic LoRA adapter loading should not be enabled where untrusted clients reach the server, and calls its gRPC interface "insecure by default" [7].

Who finds them. Pillar Security ran honeypots between December 2025 and January 2026 and captured 35,000 attack sessions, an average of 972 a day [3]. The targets included "Ollama running on port 11434 without authentication" and chat-completions-compatible APIs on port 8000 exposed to the internet [3]. Pillar reports that "once an endpoint appears in Shodan or Censys scan results, exploitation attempts typically begin within hours" [3].

Controls. Talos writes that Ollama instances and LLM servers in general "should never be publicly exposed without requiring secure API key-based or token-based authentication", and adds changing default ports, disabling verbose banners and watching for endpoints that become publicly reachable [1]. Pillar asks

operators to verify that Ollama, vLLM and similar services "require valid credentials for all requests", with rate limiting [3]. vLLM recommends a reverse proxy that "explicitly allowlists only the endpoints you want to expose to end users", blocks all other endpoints and adds authentication, rate limiting and logging [7].

2. Vulnerabilities in serving software

Probllama. CVE-2024-37032 affected Ollama before version 0.1.34, which "does not validate the format of the digest (sha256 with 64 hex digits) when getting the model path" [6]. NVD lists it as CWE-22, path traversal, with a CVSS 3.1 base score of 8.8 from CISA-ADP. MITRE, the numbering authority, gave no score, and NVD has not scored it [6]. Wiz, which found it, explains that a malicious registry manifest with a path traversal payload in its digest field lets an attacker overwrite files on the server, which "can be further exploited into a full Remote Code Execution" [5]. Wiz reported the flaw on 5 May 2024, a patched version was released on 8 May and the write-up followed on 24 June [5]. Wiz called the issue "extremely severe in Docker installations" [5].

ShadowMQ. In November 2025 Oligo Security traced a family of flaws to "the overlooked unsafe use of ZeroMQ (ZMQ) and Python's pickle deserialization" [9]. Oligo found that "code files were copied between projects (sometimes line-for-line)", so that vLLM, SGLang and two other inference servers "all contained nearly identical unsafe patterns: pickle deserialization over unauthenticated ZMQ TCP sockets" [9]. In SGLang the file in question begins "Adapted from vLLM" [9]. Oligo also reports "thousands of exposed ZMQ sockets communicating unencrypted over the public internet" [9].

A separate vLLM flaw with the same pattern, reported outside Oligo's research, is CVE-2025-32444. Versions from 0.6.5 to before 0.8.5 with the mooncake integration "are vulnerable to remote code execution due to using pickle based serialization over unsecured ZeroMQ sockets", and those sockets listened on all network interfaces [8]. Deployments without mooncake were not affected, and 0.8.5 carries the fix. GitHub, as the numbering authority, scores it 10.0 and NVD 9.8 [8].

Traffic between nodes. The insecure default in vLLM covers PyTorch Distributed, KV cache transfer and parallel communication, and vLLM restates PyTorch's warning that "Connections are accepted from anywhere without checks" [7]. Its firewall advice is to "Block all incoming connections except to the TCP port the API server is listening on" [7].

Controls. Oligo's advice is "Don't use pickle or recv_pyobj() with untrusted data" [9]. OWASP's supply chain entry asks for a patching policy "to mitigate vulnerable or outdated components" and an up-to-date, accurate and signed inventory of components [24]. The vLLM guidance adds the isolated network and the firewall rule above [7]. Both Probllama and the mooncake flaw were closed by upgrading to a named release [5, 8].

3. Model files that execute code

Pickle. Hugging Face's Hub documentation warns that "there are dangerous arbitrary code execution attacks that can be perpetrated when you load a pickle file" [10]. The Hub scans uploads with ClamAV and with pickle import scans, notes that ClamAV "doesn't cover pickle exploits", and adds: "this is not 100% foolproof. It is your responsibility as a user to check if something is safe or not" [10].

Cases. In February 2024 JFrog reported around 100 models on the Hub "housing real, harmful payloads" [11]. In one PyTorch model the payload was injected through the pickle module's **reduce** method and opened a reverse shell to a real IP address when loaded [11]. PyTorch models led the count, followed closely by TensorFlow Keras models, and the Hub marked scanned models unsafe without blocking their download [11].

In February 2025 ReversingLabs reported two models with malicious code "that were not flagged as 'unsafe'" by the Hub's scanning [12]. They were compressed with 7z rather than PyTorch's default ZIP format, which stops torch.load() from opening them, and the scanner missed them [12]. Their pickle streams were also broken, and Picklescan failed to detect dangerous functions that run before the break, because "Pickle opcodes are executed as they are encountered" [12]. Hugging Face removed the models in less than 24 hours and changed the Picklescan tool to catch threats in broken pickle files [12].

Loaders and formats. PyTorch's documentation states that since version 2.6, torch.load uses weights_only=True when no pickle module is passed [13]. That setting "narrows the surface of remote code execution attacks" but does not guard against denial of service, and the documentation adds that memory corruption "might still be possible" [13].

The restricted path had a flaw of its own. CVE-2025-32434 describes remote command execution in PyTorch 2.5.1 and earlier "when loading a model using torch.load with weights_only=True", patched in 2.6.0, scored 9.3 under CVSS 4.0 by GitHub and 9.8 under CVSS 3.1 by NVD [14]. Hugging Face points to safetensors as the alternative to pickle [10]. A 2023 audit of safetensors by Trail of Bits, commissioned by Hugging Face with two other organizations, found "no critical security flaw leading to arbitrary code execution" [15].

Code can also travel with a model. Hugging Face's Transformers documentation asks for "extra precaution when loading a custom model", which requires trust_remote_code=True, and advises loading custom models from a specific revision "to avoid loading model code that may have changed" [16].

Framing. OWASP's supply chain entry asks teams to "only use models from verifiable sources and use third-party model integrity checks with signing and file hashes" [24]. NIST AI 100-2 notes that attackers can offer "maliciously designed models, such as pre-trained models that enable a BACKDOOR POISONING ATTACK" [26].

4. Shared KV caches between tenants

The mechanism. Serving engines keep a key-value (KV) cache for the tokens they have processed. vLLM's prefix caching "reuses KV cache blocks across requests that share a common prompt prefix", and its documentation calls that reuse a timing side channel in multi-tenant deployments: an attacker "can measure differences in Time to First Token (TTFT) to infer whether a guessed prompt prefix matches another user's cached prompt" [7]. NVD's record for CVE-2025-46570 describes the effect in vLLM before 0.9.0: when a matching prefix chunk is found, prefill speeds up, and "these timing differences caused by matching chunks are significant enough to be recognized and exploited" [19]. GitHub scores it 2.6, low [19].

The research. At NDSS 2025, Wu and colleagues studied KV cache sharing "for identical token sequences among multiple users" in SGLang and vLLM, and showed that it "may lead to new side channel attack vectors, allowing unauthorized reconstruction of user prompts" [17]. Their attack, PROMPTPEEK, reached "an average success rate of 99% in fully or partially reversing the prompt input, 98% in reversing the prompt template, and 95% without additional background knowledge" on a 13-billion-parameter open-weight model on an A100 80G GPU [17].

The attack needs an adversary who acts "as a standard user without special privileges" and can observe whether requests trigger sharing [17]. The authors conclude that "service providers should obscure KV cache activities from clients" [17].

A study of hosted services found the same behavior in production. Gu and colleagues note that "cached prompts are processed faster than non-cached prompts", and detected "global cache sharing across users in seven API providers", with potential leakage about users' prompts [18].

The control. vLLM's defense is a cache salt: "The salt is mixed into the hash of the first KV cache block, so only requests carrying the same salt can share cached prefix blocks" [7]. Without it, "all requests can share cached prefixes", and "cache_salt is opt-in and not passed by default" [7]. For multi-tenant deployments vLLM asks operators to "set cache_salt on every request, using a secret scoped to the tenant boundary you want to enforce", recommends random salts long enough to be unpredictable, for example 43 base64 characters (256 bits), and notes that "salting reduces cache efficiency" [7]. A vLLM server shared by several teams therefore keeps cross-team prefix sharing unless the layer in front of it adds a salt per team.

5. Poisoned documents and vector indexes

Planted instructions. Greshake and colleagues wrote in 2023 that LLM-integrated applications "blur the line between data and instructions", and showed that adversaries can "remotely (without a direct interface) exploit LLM-integrated applications by strategically injecting prompts into data likely to be retrieved" [20]. In 2023 they judged that "effective mitigations of these emerging threats are currently lacking" [20]. NIST AI 100-2 describes the same weakness: because these models "combine the data and instruction channels", attackers can use the data channel to affect system operations, and indirect prompt injection is mounted "by a third party" through resource control [26].

Poisoned knowledge bases. PoisonedRAG showed that "an attacker could inject a few malicious texts into the knowledge database of a RAG system to induce an LLM to generate an attacker-chosen target answer for an attacker-chosen target question" [21]. It reached "a 90% attack success rate when injecting five malicious texts for each target question into a knowledge database with millions of texts", and the authors found the defenses they evaluated "insufficient" [21].

The index itself. Embeddings carry much of their text. Morris and colleagues showed that "a multi-step method that iteratively corrects and re-embeds text is able to recover 92% of 32-token text inputs exactly", and recovered full names from a dataset of clinical notes [22]. OWASP's entry on vector and embedding weaknesses warns that "in multi-tenant environments where multiple classes of users or applications share the same vector database, there's a risk of context leakage between users or queries", and that poisoned data "can originate from insiders, prompts, data seeding, or unverified data providers" [25].

Controls. OWASP asks for "fine-grained access controls and permission-aware vector and embedding stores", "strict logical and access partitioning of datasets in the vector database" and data accepted "only from trusted and verified sources" [25]. The inversion result means embeddings need the same access rules as the documents behind them [22].

6. The GPU container layer: the NVIDIA Container Toolkit CVE history

CVE	Published	What the record describes	Scores	Fixed in	CDI use	Sources
CVE-2024-0132	26 Sep 2024	A crafted image may gain access to the host file system, default configuration	9.0 NVIDIA, 8.3 NVD	Toolkit 1.16.2, GPU Operator 24.6.2	Not affected	[27, 28, 33, 36]
CVE-2025-23359	12 Feb 2025	A crafted image could gain access to the host file system, default configuration	8.3 NVIDIA, 8.1 NVD	Toolkit 1.17.4	Not stated	[32, 33]
CVE-2025-23266	17 Jul 2025	Arbitrary code with elevated permissions through container setup hooks	9.0 NVIDIA, no NVD score	Toolkit 1.17.8, device plugin 0.17.3, GPU Operator 25.3.2	Affected	[29, 30, 31]

Table 1. Three NVIDIA Container Toolkit CVEs that reach the host, from the NVD records, NVIDIA's advisory and release notes, the GitHub advisory and Wiz.

CVE-2024-0132. NVD's record describes a time-of-check time-of-use (TOCTOU) flaw in toolkit 1.16.1 and earlier, in the default configuration, that "does not impact use cases where CDI is used" and may lead to code execution, escalation of privileges and data tampering, among other effects [27]. NVIDIA's own advisory in its GitHub repository, published 8 October 2024, rates it critical at 9.0 and names version 1.16.2 as patched [36].

Wiz, which reported it on 1 September 2024, wrote that it "enables attackers who control a container image executed by the vulnerable toolkit to escape from that container and gain full access to the underlying host system", and listed GPU Operator 24.6.1 and earlier as affected [28]. In shared environments such as Kubernetes, Wiz wrote, an attacker with permission to deploy a container "could escape that container and gain access to data and secrets of other applications running on the same node", or even the same cluster [28].

CVE-2025-23359. In February 2025 NVD published a second TOCTOU flaw in the toolkit's default configuration, through which "a crafted container image could gain access to the host file system" [32]. NVIDIA scores it 8.3 and NVD 8.1, and the toolkit's 1.17.4 release notes list it [32, 33].

CVE-2025-23266 (NVIDIAScape). NVD's record says the toolkit "for all platforms contains a vulnerability in some hooks used to initialize the container, where an attacker could execute arbitrary code with elevated permissions", classed as CWE-426, untrusted search path [29]. NVIDIA scores it 9.0, and NVD lists no score of its own [29]. Wiz reported it on 17 May 2025 at Pwn2Own Berlin and published on 17 July 2025, two days after NVIDIA's bulletin [31].

Wiz writes that it "allows a malicious container to bypass isolation measures and gain full root access to the host machine", through a misconfiguration in how the toolkit handles OCI hooks, whose createContainer hooks "inherit environment variables from the container image unless explicitly configured not to" [31]. Wiz lists toolkit versions up to and including 1.17.7 as affected, with only CDI mode affected before 1.17.5, and the GPU Operator up to and including 25.3.1 [31]. The GitHub advisory lists the patched releases of the toolkit (1.17.8), the device plugin (0.17.3) and the GPU Operator (25.3.2) [30]. As an interim step Wiz describes opting out of the enable-cuda-compat hook [31], which the 1.17.5 release notes say was re-added for CUDA forward compatibility and can be disabled with a feature flag [33].

A run of fixes. Version 1.16.2 "provides critical security updates and is recommended for all users" and lists CVE-2024-0132 and CVE-2024-0133 [33]. Version 1.17.0 fixes CVE-2024-0134, in which symlinks created as a container starts could escape the container root. Version 1.17.3 lists CVE-2024-0135, CVE-2024-0136 and CVE-2024-0137, and 1.17.4 lists CVE-2025-23359 [33]. Counted from these notes, releases 1.16.2 to 1.17.4 carried fixes for at least seven CVEs [33].

CDI. CDI "is an open specification for container runtimes that abstracts what access to a device, such as an NVIDIA GPU, means, and standardizes access across container runtimes" [34]. The toolkit can generate CDI specifications from version 1.12.0 [34]. In 1.18.0 the runtime's default mode moved to a just-in-time CDI specification, and the legacy mode is deprecated [33]. CVE-2024-0132 did not affect CDI use [27, 28], while Wiz lists CDI mode as affected by CVE-2025-23266 [31]. Moving to CDI changed which flaws applied, and patched releases were the fix for both.

Isolation beyond the container. Wiz writes that "containers are not a strong security barrier and should not be relied upon as the sole means of isolation", and advises at least one strong barrier such as virtualization [31]. The Kubernetes documentation on multi-tenancy says containers "are processes running on a shared kernel" that mount /sys and /proc from the host, which makes them "less secure than an application that runs on a virtual machine which has its own kernel" [35]. It names virtual machines and userspace kernels as two approaches to sandboxing that insulate the host from container escapes [35]. Both Wiz write-ups advise prioritizing hosts that run images from untrusted or public sources [28, 31].

7. Mapping to OWASP 2025 and NIST AI 100-2

Failure mode	OWASP Top 10 for LLM Applications 2025	NIST AI 100-2 E2025
Exposed inference endpoints	LLM10 Unbounded Consumption	Availability breakdown (NISTAML.01), and the network part of the attack surface
Serving software flaws	LLM03 Supply Chain	The software part of the attack surface and its third-party dependencies
Model files that run code	LLM03 Supply Chain, LLM04 Data and Model Poisoning	Supply chain attacks (NISTAML.05), model poisoning (NISTAML.051)
Shared KV caches	LLM02 Sensitive Information Disclosure, LLM07 System Prompt Leakage	Privacy compromise (NISTAML.03)
Poisoned documents and indexes	LLM01 Prompt Injection, LLM04 Data and Model Poisoning, LLM08 Vector and Embedding Weaknesses	Indirect prompt injection (NISTAML.015), integrity violation (NISTAML.02)
GPU container escapes	LLM03 Supply Chain	The software, network and storage parts of the attack surface

Table 2. Each failure mode placed in the OWASP 2025 list and NIST AI 100-2 E2025, a mapping made for this report from the entry names in [23] and the categories in [26].

OWASP's list has no entry for an unauthenticated server as such, so exposed endpoints map only in part, through unmetered use of the model (LLM10) [23]. The KV cache leak maps to LLM07 as well as LLM02 because PROMPTPEEK also reversed prompt templates [17]. LLM05, LLM06 and LLM09 concern what a model or agent does with its output, and fall outside this report.

8. Controls for each failure mode

Failure mode	Controls the sources give	Sources
Exposed inference endpoints	Authentication on every request, a reverse proxy that allows listed endpoints only, rate limits and logging, watching for endpoints that become public	[1, 3, 7]
Serving software flaws	A patching policy, isolated networks between nodes, inbound traffic only to the API port, no pickle over untrusted sockets	[7, 9, 24]
Model files that run code	Safetensors or weights-only loading on PyTorch 2.6.0 or later, models only from verifiable sources with signatures and hashes, custom code pinned to a revision	[13, 14, 15, 16, 24]
Shared KV caches	A secret cache salt on every request, scoped to the tenant boundary, and cache activity hidden from clients	[7, 17]
Poisoned documents and indexes	Permission-aware vector stores, logical and access partitioning of datasets, data only from trusted and verified sources, embeddings treated like their text	[22, 25]
GPU container escapes	Patched toolkit, device plugin and GPU Operator releases, priority for hosts that run untrusted images, a stronger isolation barrier such as virtualization	[28, 30, 31, 35]

Table 3. Controls drawn from each source's own mitigation text.

Implications

Defaults set the starting exposure. Ollama ships bound to localhost [4], so an exposed Ollama server comes from a changed bind address or an installation that listened on all interfaces by default [4, 5]. vLLM shares its prefix cache across all requests and leaves traffic between nodes unauthenticated unless the operator acts [7]. A review of a self-hosted deployment can start from two lists: every port each component listens on, and every cache or index that more than one tenant reads.

Patching is part of the design. Each CVE in this report was closed by a release: Ollama 0.1.34 [5, 6], vLLM 0.8.5 and 0.9.0 [8, 19], PyTorch 2.6.0 [14] and toolkit releases 1.16.2 to 1.17.8 [30, 33]. The ShadowMQ pattern was not closed everywhere: in November 2025 Oligo reported that SGLang had implemented incomplete fixes [9]. The Probllama fix came three days after the report [5], and CVE-2024-0132 was fixed 25 days after Wiz reported it [28].

Scanners and safer formats leave gaps. Two malicious models passed the Hub's scanner in 2025 [12], and PyTorch's restricted loader carried a flaw scored 9.8 by NVD until 2.6.0 [14].

Sharing changes which controls apply. One team on one server needs authentication and patching. Several teams on one server, one vector database or one GPU node also need a cache salt per tenant [7], partitioned indexes [25] and an isolation barrier stronger than the container [31, 35].

Research results are not incident counts. The KV cache and poisoning findings are controlled experiments [17, 21]. Their conditions are what a design review checks: whether an untrusted user shares a cache with others, and who can write to the index.

Limits of this analysis

- The exposure counts cover Ollama only and come from two scans published in September 2025 with different methods [1, 2]. Talos notes a bias toward default ports [1]. No comparable count for vLLM or other servers was found.
- The KV cache findings are research results and documented defaults [7, 17, 18, 19]. No source reviewed counts attacks through a KV cache outside research, or how many self-hosted deployments share caches across tenants.
- The poisoning results are demonstrations [20, 21]. No source reviewed measures how often poisoned documents appear in real retrieval indexes.
- JFrog's count of around 100 malicious models dates from February 2024 [11]. No later count was reviewed.
- NVIDIA's security bulletin pages refused automated access. NVIDIA's position comes from its GitHub advisory and release notes [33, 36], and the bulletins' own wording is not quoted.
- Versions, defaults and flags are as documented on each source's viewed date.
- The mappings in Findings 7 and 8 are this report's own. OWASP and NIST do not place these cases themselves.
- Tool permissions for agents, prompt injection test methods and theft of model weights are outside this report.

Sources

1. Detecting Exposed LLM Servers: A Shodan Case Study on Ollama. Cisco Talos, Cisco Blogs, 1 September 2025, viewed 8 November 2026. https://blogs.cisco.com/security/detecting-exposed-llm-servers-shodan-case-study-on-ollama

2. Ollama Drama: Investigating the Prevalence of Ollama Open Instances with Censys. Censys ARC Research Team, Censys, 24 September 2025, viewed 8 November 2026. https://censys.com/blog/ollama-drama-investigating-the-prevalence-of-ollama-open-instances-with-censys/

3. Operation Bizarre Bazaar: First Attributed LLMjacking Campaign with Commercial Marketplace Monetization. Pillar Security, 28 January 2026, viewed 8 November 2026. https://www.pillar.security/blog/operation-bizarre-bazaar-first-attributed-llmjacking-campaign-with-commercial-marketplace-monetization

4. FAQ. Ollama documentation, Ollama, undated, viewed 8 November 2026. https://docs.ollama.com/faq

5. Probllama: Ollama Remote Code Execution Vulnerability (CVE-2024-37032), Overview and Mitigations. Wiz Research, Wiz, 24 June 2024, viewed 8 November 2026. https://www.wiz.io/blog/probllama-ollama-vulnerability-cve-2024-37032

6. CVE-2024-37032. National Vulnerability Database, National Institute of Standards and Technology, published 31 May 2024, viewed 8 November 2026. https://nvd.nist.gov/vuln/detail/CVE-2024-37032

7. Security. vLLM documentation, vLLM project, page dated 10 November 2026, viewed 11 November 2026. https://docs.vllm.ai/en/latest/usage/security/

8. CVE-2025-32444. National Vulnerability Database, National Institute of Standards and Technology, published 30 April 2025, viewed 8 November 2026. https://nvd.nist.gov/vuln/detail/CVE-2025-32444

9. ShadowMQ: How Code Reuse Spread Critical Vulnerabilities Across the AI Ecosystem. Oligo Security, 13 November 2025, viewed 8 November 2026. https://www.oligo.security/blog/shadowmq-how-code-reuse-spread-critical-vulnerabilities-across-the-ai-ecosystem

10. Pickle Scanning. Hugging Face Hub documentation, Hugging Face, undated, viewed 8 November 2026. https://huggingface.co/docs/hub/security-pickle

11. Data Scientists Targeted by Malicious Hugging Face ML Models with Silent Backdoor. JFrog Security Research, JFrog, 27 February 2024, viewed 8 November 2026. https://jfrog.com/blog/data-scientists-targeted-by-malicious-hugging-face-ml-models-with-silent-backdoor/

12. Malicious ML models discovered on Hugging Face platform. ReversingLabs, 6 February 2025, viewed 8 November 2026. https://www.reversinglabs.com/blog/rl-identifies-malware-ml-model-hosted-on-hugging-face

13. Serialization semantics. PyTorch documentation, version 2.15, PyTorch, undated, viewed 8 November 2026. https://docs.pytorch.org/docs/2.15/notes/serialization.html

14. CVE-2025-32434. National Vulnerability Database, National Institute of Standards and Technology, published 18 April 2025, viewed 8 November 2026. https://nvd.nist.gov/vuln/detail/CVE-2025-32434

15. Audit shows that safetensors is safe and ready to become the default. Hugging Face blog, Hugging Face, 23 May 2023, viewed 8 November 2026. https://huggingface.co/blog/safetensors-security-audit

16. Loading models. Transformers documentation, Hugging Face, undated, viewed 8 November 2026. https://huggingface.co/docs/transformers/en/models

17. I Know What You Asked: Prompt Leakage via KV-Cache Sharing in Multi-Tenant LLM Serving. Guanlong Wu, Zheng Zhang, Yao Zhang, Weili Wang, Jianyu Niu, Ye Wu and Yinqian Zhang, Network and Distributed System Security (NDSS) Symposium 2025, February 2025, DOI 10.14722/ndss.2025.241772, viewed 8 November 2026. https://www.ndss-symposium.org/wp-content/uploads/2025-1772-paper.pdf

18. Auditing Prompt Caching in Language Model APIs. Chenchen Gu, Xiang Lisa Li, Rohith Kuditipudi, Percy Liang and Tatsunori Hashimoto, arXiv:2502.07776, version 2, 13 July 2025 (ICML 2025), viewed 8 November 2026. https://arxiv.org/abs/2502.07776

19. CVE-2025-46570. National Vulnerability Database, National Institute of Standards and Technology, published 29 May 2025, viewed 8 November 2026. https://nvd.nist.gov/vuln/detail/CVE-2025-46570

20. Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. Kai Greshake and others, arXiv:2302.12173, version 2, 5 May 2023, viewed 8 November 2026. https://arxiv.org/abs/2302.12173

21. PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models. Wei Zou, Runpeng Geng, Binghui Wang and Jinyuan Jia, arXiv:2402.07867, version 3, 13 August 2024, viewed 8 November 2026. https://arxiv.org/abs/2402.07867

22. Text Embeddings Reveal (Almost) As Much As Text. John X. Morris, Volodymyr Kuleshov, Vitaly Shmatikov and Alexander M. Rush, arXiv:2310.06816, 10 October 2023 (EMNLP 2023), viewed 8 November 2026. https://arxiv.org/abs/2310.06816

23. OWASP Top 10 for LLM Applications 2025. OWASP Gen AI Security Project, 2025, viewed 8 November 2026. https://genai.owasp.org/llm-top-10/

24. LLM03:2025 Supply Chain. OWASP Gen AI Security Project, 2025, viewed 8 November 2026. https://genai.owasp.org/llmrisk/llm032025-supply-chain/

25. LLM08:2025 Vector and Embedding Weaknesses. OWASP Gen AI Security Project, 2025, viewed 8 November 2026. https://genai.owasp.org/llmrisk/llm082025-vector-and-embedding-weaknesses/

26. Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations, NIST AI 100-2 E2025. National Institute of Standards and Technology, March 2025, DOI 10.6028/NIST.AI.100-2e2025, viewed 8 November 2026. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2025.pdf

27. CVE-2024-0132. National Vulnerability Database, National Institute of Standards and Technology, published 26 September 2024, viewed 8 November 2026. https://nvd.nist.gov/vuln/detail/CVE-2024-0132

28. Wiz Research Finds Critical NVIDIA AI Vulnerability Affecting Containers Using NVIDIA GPUs. Shir Tamari, Ronen Shustin and Andres Riancho, Wiz Research, 26 September 2024, viewed 8 November 2026. https://www.wiz.io/blog/wiz-research-

critical-nvidia-ai-vulnerability

29. CVE-2025-23266. National Vulnerability Database, National Institute of Standards and Technology, published 17 July 2025, viewed 8 November 2026. https://nvd.nist.gov/vuln/detail/CVE-2025-23266

30. GHSA-vmg3-7v43-9g23: NVIDIA Container Toolkit for all platforms contains an Untrusted Search Path. GitHub Advisory Database, GitHub, 17 July 2025, viewed 8 November 2026. https://github.com/advisories/GHSA-vmg3-7v43-9g23

31. NVIDIAScape: A Three-Line Container Escape in NVIDIA Container Toolkit (CVE-2025-23266). Nir Ohfeld and Shir Tamari, Wiz Research, 17 July 2025, viewed 8 November 2026. https://www.wiz.io/blog/nvidia-ai-vulnerability-cve-2025-23266-nvidiascape

32. CVE-2025-23359. National Vulnerability Database, National Institute of Standards and Technology, published 12 February 2025, viewed 8 November 2026. https://nvd.nist.gov/vuln/detail/CVE-2025-23359

33. NVIDIA Container Toolkit Release Notes. NVIDIA documentation, NVIDIA, undated (latest release listed 1.20.1), viewed 8 November 2026. https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/release-notes.html

34. Support for Container Device Interface. NVIDIA Container Toolkit documentation, NVIDIA, undated, viewed 8 November 2026. https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/cdi-support.html

35. Multi-tenancy. Kubernetes documentation, The Kubernetes Authors, last modified 22 May 2025, viewed 8 November 2026. https://kubernetes.io/docs/concepts/security/multi-tenancy/

36. Security advisory GHSA-mjjw-553x-87pq (CVE-2024-0132). NVIDIA, nvidia-container-toolkit repository on GitHub, 8 October 2024, viewed 8 November 2026. https://github.com/NVIDIA/nvidia-container-toolkit/security/advisories/GHSA-mjjw-553x-87pq

About this report

Published by Rillor on 12 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. Security risks in self-hosted LLM deployments. 12 November 2026. https://rillor.com/insights/self-hosted-llm-security-risks

Notices

Rillor is not a registered investment adviser or commodity trading advisor and does not provide investment or trading advice. Research, forecasts, data and software described on this site are for research and engineering use. Nothing here is an offer or recommendation to buy or sell any security, commodity interest or digital asset. Past or simulated results do not indicate future results.

NVIDIA, AMD, Ceph, Kubernetes and related product names are trademarks of their owners and are used only to identify products. No affiliation or endorsement is implied.

Superintelligence (SI) outputs can be wrong. Important decisions should include human review.

Work with Rillor

Send us the workloads you want to run, the data they use, who uses them and where your hardware sits today. Contact us at rillor.com/contact?type=private-si or contact@rillor.com. More on private SI cloud: rillor.com/compute/private-si