SoK: A taxonomy of LLM threats

Ioannis Stylianou[a,b,i], Panagiotis Bountakas[a], Apostolis Zarras[a], Aristeidis Farao[a,b], Vaios Bolgouras[a] and Christos Xenakis[a,b]

[a] *University of Piraeus, 80 Karaoli & Dimitriou, Piraeus, 18534, Attica, Greece*
[b] *InQbit Innovations SRL, Bucharest, Romania*

ARTICLE INFO

Keywords:
Adversarial attacks
Large language models
AI security
Cyber threats

ABSTRACT

Securing large language models (LLMs) is critical as deployments scale, yet existing guidance is fragmented and phase-specific. This Systematization of Knowledge paper presents a unified, lifecycle-centric taxonomy of 20 attack classes spanning training, inference, and system integration. We synthesize academic and industry work into a dependency-aware threat schema that addresses causal and associative interdependencies, grounded in real-world case studies and visual mappings. Building on this, we develop a defense-in-depth strategy for coverage-oriented risk assessment, mitigation prioritization, security posture measurement, and open research directions in adversarial ML and secure systems engineering.

1. Introduction

The rapid proliferation of Large Language Models (LLMs) and the emergence of autonomous, Generative Artificial Intelligence (GenAI) systems have unlocked unprecedented capabilities in natural language understanding, generation, and decision-making. However, a diverse and evolving landscape of cyber-threats has emerged alongside these advances, ranging from data and model poisoning during training to prompt-injection and supply-chain compromises at inference and system levels. Critical flaws and vulnerabilities have been discovered not only in models with limited usage, but also in well-known and well-used Large Language Model (LLM) Agents, like Google's Gemini CLI, which could execute arbitrary malicious code on users' machines [1].

Several studies have proposed methods to compromise LLMs, with prompt injection emerging as one of the most prominent threats [2]. In response, recent efforts by the Open Worldwide Application Security Project (OWASP) have released a top-ten list of LLM-specific threats, providing a structured foundation for researchers and security professionals to investigate and address each category through targeted attacks and defenses [3]. While surveys on Artificial Intelligence (AI) system vulnerabilities have significantly contributed to enhancing the security of Machine Learning (ML) and AI applications [4], similar systematic reviews that focus explicitly on LLMs remain nascent [5].

At the same time, despite numerous industry and academic efforts to categorize isolated attacks on LLMs [3, 6–10], no unified, lifecycle-centric framework exists that *.i/* spans training, inference, and system-level threats, *.ii/* synthesizes interdependencies among adversarial techniques, and *.iii/* guides both researchers and practitioners toward prioritized defenses.

Existing AI security taxonomies often isolate a single life cycle stage or threat vector and rarely state explicit inclusion criteria or map the end-to-end attack surface for LLMs and agentic systems. As these capabilities move into critical workflows, there is a need for a holistic taxonomy that: *.i/* identifies relevant attacks across all stages (e.g., training, inference, system), *.ii/* traces causal and associative links among threat classes, *.iii/* recommends mitigations grounded in incident case studies and prior taxonomies, and *.iv/* aligns with established industry standards to support adoption.

In this Systematization of Knowledge (SoK) and taxonomy paper, we address this gap by introducing a taxonomy of LLM and GenAI threats that offers a coherent reference for security teams to assess vulnerabilities rapidly, prioritize defenses based on inter-threat dependencies, and benchmark their posture against best practices from technical reports (e.g., CSA, Cisco, OWASP, MITRE, FS-ISAC) and academic literature. Grounded in real-world case studies, our framework maps theoretical attack classes to observed compromises, translating abstract threat categories into actionable insights.

In summary, we make the following main contributions:

- We propose a unified taxonomy comprising 20 distinct threat classes spanning the LLM lifecycle, categorized by training-time, inference-time, and system-level threats.

[i] Corresponding author at: University of Piraeus, 80 Karaoli & Dimitriou, Piraeus, 18534, Attica, Greece.
Email addresses: stylianou@unipi.gr (I. Stylianou); bountakas@unipi.gr (P. Bountakas); zarras@unipi.gr (A. Zarras); arisfarao@unipi.gr (A. Farao); vbolgouras@unipi.gr (V. Bolgouras); xenakis@unipi.gr (C. Xenakis)
ORCID(s): 0009-0008-6490-6721 (I. Stylianou)

- We analyze causal and associative relationships among threat classes based on literature and case-studies, and present them using intuitive, color-coded visualizations.
- We demonstrate how these threats manifest in real-world and experimental contexts.
- We outline technical and organizational countermeasures for threats, informed by existing literature and technical reports.

The remainder of this manuscript is organized as follows. Section 2 details our methodology for constructing the taxonomy and selecting incident case studies. Section 3 reviews prior work and situates our taxonomy within existing frameworks. Section 4 details the threat classification and inter-threat relationships. Section 5 presents real-world case studies. Section 6 describes our defense-in-depth recommendations. Section 7 describes the causal and associative relationships between threats. Section 8 discusses limitations and outlines future research directions. Finally, Section 9 offers concluding remarks.

2. Methodology

We derive our taxonomy and dependency map using a mixed-methods design that combines a structured review of academic and industry sources with an analysis of real-world incidents. Our methodology comprises four stages: (i) identifying and screening relevant literature and threat taxonomies, (ii) consolidating categories into a unified schema, (iii) analyzing representative incident case studies, and (iv) mapping threats to mitigations and inter-threat relationships. This process is iterative, with insights from later stages feeding back to refine earlier ones.

2.1. Literature and taxonomy search

We conducted a structured search of major bibliographic databases (Google Scholar, ACM Digital Library, arXiv, IEEE Xplore, SpringerLink, Elsevier) and gray-literature repositories (OWASP, CSA, MITRE, FS-ISAC, vendor security whitepapers) over 2015–2025. Searches combined model-related terms (e.g., "large language model", "LLM", "agentic AI") with security concepts ("threats", "vulnerabilities", "risks", "attacks", "incidents", "taxonomy"). To mitigate terminology drift across academia and industry and to accommodate differences in database query syntax, we used an iteratively refined controlled vocabulary and source-specific query templates that enumerate synonyms and aliases across model, deployment/integration, threat/attack, and framework/review concepts, and expanded this vocabulary with additional aliases observed during screening (Appendix A). We retained sources that primarily addressed security, privacy, or safety of LLMs or closely related foundation models, or that proposed a structured taxonomy, survey, or framework for AI/ML threats with an explicit mapping to LLM-based systems, and excluded purely marketing-oriented material or work that treated AI security only tangentially. Although the focus of this SoK is LLM threats, we intentionally included broader AI/ML-security and agentic-system terms because (i) many relevant frameworks and surveys index LLM-related risks under broader AI/ML terminology, and (ii) modern deployments increasingly expose LLMs through tool-using agents, introducing distinct system-level risks (e.g., excessive agency and insecure tool use). This yielded a curated corpus of peer-reviewed studies and influential industry frameworks [3, 6–10], which forms the evidence base for our taxonomy. After converging on our threat classes, we ran targeted follow-up searches using each class name as a keyword to capture additional specialized work.

2.2. Taxonomy construction and harmonization

From each selected source we extracted the threat, risk, or vulnerability categories used to describe attacks against LLMs and related AI systems, along with their definitions and example scenarios. We then applied open coding to group conceptually similar categories into canonical labels, normalizing synonymous or overlapping terms (e.g., "sensitive data disclosure", "privacy leakage", and "training data memorization" under a unified data-leakage threat).

Annotator setup and coding protocol. Coding was performed by two authors who independently mapped extracted categories to candidate codes using a shared coding sheet and an evolving codebook. We first co-coded an initial calibration subset to refine the codebook and harmonize interpretation. The codebook defined, for each threat class, (i) a canonical label, (ii) lifecycle-phase assignment, (iii) inclusion/exclusion criteria, and (iv) representative indicators and examples. We then coded the remaining corpus in iterative batches; when new sources introduced a genuinely distinct concept (new attacker objective and/or new attack surface), we updated the codebook and revisited earlier-coded items when necessary to maintain internal consistency.

Disagreement handling. Disagreements were logged at three levels: (a) label mapping (which code best fits a source category), (b) scope/boundaries (whether two codes should be merged or split), and (c) lifecycle assignment (training-time vs inference-time vs system-level). Resolution followed a structured reconciliation process: we re-checked the original source definitions and examples and applied decision rules prioritizing (i) the *attacker's primary objective* (e.g., behavioral control vs information disclosure), (ii) required attacker capability and access (e.g., oracle access vs artifact tampering vs runtime integration access), and (iii) whether the *mitigation surface* differs

materially (i.e., whether a practitioner would defend it differently). When consensus was not immediate, the case was escalated to the broader author group for adjudication.

Practical merge/split heuristics and representative boundary cases. We merged categories when they were terminological variants describing the same mechanism and mitigation surface, and split categories when differences in objective, access, or mitigations would change how the threat should be evaluated or defended. Most reconciliation discussions concerned conceptually adjacent threats already distinguished in our threat descriptions, including: (i) Prompt Injection (Threat 03) vs Meta Prompt Extraction (Threat 04) (behavior override vs confidential prompt disclosure); (ii) Data Leakage (Threat 05) vs Model Inversion and Reconstruction (Threat 09) vs Model Theft (Threat 10) (passive memorization/exposure vs active analytical reconstruction vs functional replication/IP theft); (iii) Supply Chain Compromise (Threat 01) vs Third-Party Integration Risks (Threat 18) (upstream artifact tampering during development/training vs runtime plugin/API exposure); (iv) Unbounded Consumption (Threat 14) vs Resource Consumption (Threat 16) (adversarially induced DoS/cost amplification vs baseline operational demands); and (v) Bias and Fairness (Threat 11) vs Ethical and Compliance Risks (Threat 15) vs Regulatory Compliance (Threat 17) (discriminatory outputs vs broader governance/policy harms vs formal legal obligations). Appendix B summarizes these boundary decisions and the rationale used in the final taxonomy.

The resulting set of candidate threats was organized according to the primary stage of the AI lifecycle in which they manifest (training-time, inference-time, or system-level). Comparing this canonical set against each source taxonomy revealed areas of convergence, divergence, and missing coverage, informing the final set of 20 threat classes presented in the threat landscape section.

2.3. Incident case study selection

To ground the taxonomy in practice, we compiled real-world incidents and high-fidelity research demonstrations involving LLMs or agentic systems. Candidate cases were identified via targeted searches of security advisories, incident reports, technical blogs, and references within the literature corpus. We prioritized incidents that (i) clearly mapped to at least one of our threat classes, (ii) had documented impact on confidentiality, integrity, availability, ethical or regulatory compliance, or operational cost, (iii) contributed to diversity in sectors, deployment settings (research prototypes, commercial services, open-source ecosystems), and attack vectors, and (iv) were recent, with the majority drawn from 2020–2025 due to the emergence of widely deployed LLM services, while also including seminal earlier incidents and high-fidelity demonstrations (2015–2019) when they best illustrated a threat class. For each case, we summarized the system context, attack mechanics, exploited weaknesses, and observed consequences, and used these analyses to validate and, where necessary, refine threat definitions and lifecycle assignments.

2.4. Inter-threat relationships and mitigation mapping

We analyzed both the literature and the incident corpus for explicit or implicit statements about dependencies between threats (e.g., data poisoning enabling downstream prompt injection). We encoded these observations in a graph where nodes represent the 20 threat classes and edges capture either causal relationships (one threat enabling or increasing the likelihood of another) or associative relationships (threats that are semantically close, consistently co-occur, or are addressed by overlapping controls). To anchor the dependency map in observable failures, we only assign a *causal* edge when at least one incident or controlled demonstration in the case-study corpus illustrates a plausible enabling mechanism; surveyed research and industry guidance are used to corroborate and clarify the mechanism. In parallel, we systematically extracted defensive measures recommended by the surveyed frameworks and post-incident analyses, and mapped each control to the lifecycle phase(s) and threat class(es) it primarily mitigates. Overlapping mitigations were consolidated into higher-level technical and organizational countermeasures, which underpin the defense-in-depth framework and relationship visualizations presented in Section 7.

3. Related work

To the best of our knowledge, comprehensive studies that systematically examine and categorize threats and vulnerabilities associated with LLMs remain limited. Existing efforts are primarily driven by community-based or commercial initiatives (e.g., CSA [6], Cisco AI Defense [7], OWASP [3], MITRE ATLAS [9], FS-ISAC [10]), while academic research in this area is still in its early stages [5]. Similarly, the literature on threats posed by AI agents is nascent. We have synthesized threats identified across multiple taxonomies, categorized by training-time, inference-time, and system-level risks as outlined in Section 4, and present them in Table 1.

The LAKERA *AI Risk Map* offers a practitioner-oriented overview of GenAI risks and lifecycle coverage and explicitly cross-links OWASP Top 10 (2025) and MITRE ATLAS with selected references to NIST AI RMF and the EU AI Act. Because the mapping focuses largely on OWASP/ATLAS rather than a broader

Table 1
Comparative coverage of threat categories in prominent LLM/AI threat taxonomies. Check marks (✓) indicate that the taxonomy explicitly covers the attack type. Parenthetical notes provide alternative naming or clarifications. Columns represent different threat taxonomies, and rows represent attack types.

Threat Type	CSA [6]	Cisco [7]	Cui et al. [8]	OWASP TOP 10 [3]	MITRE ATLAS [9]	FS-ISAC [10]
Training-Time Attacks						
Data Poisoning	✓	✓ (*Training Data Poisoning*)	✓ (*Data poisoning attacks*)	✓ (*LLM03: Training Data Poisoning*)	✓ (*Poison Training Data*)	✓ (*Poisoning Attack*)
Backdoor Attacks	✗ (*implicit in Data Poisoning*)	✓ (*Targeted Poisoning/backdoors*)	✓ (*Backdoor attacks via poisoning*)	✓ (*Compromised models/backdoors*)	✗ (*not explicitly covered*)	✓ (*Backdoor Poisoning*)
Inference-Time Attacks						
Prompt Injection	✓ (*via Model Manipulation*)	✓	✓ (*Prompt injection attacks*)	✓ (*LLM01: Prompt Injection*)	✓ (*LLM Prompt Injection technique*)	✓ (*Backdoor Poisoning*)
Adversarial Inputs	✓ (*adversarial inputs under Model Manipulation*)	✓ (*Indirect Prompt Injection*)	✓ (*Adversarial prompts incl. jailbreaks*)	✗ (*not covered*)	✓ (*Adversarial inputs causing misclassification*)	✗ (*not explicitly covered*)
Model Extraction	✓ (*Model Theft*)	✓ (*Model Theft as IP theft*)	✓ (*Model extraction/stealing*)	✓ (*LLM10: Model Theft*)	✓ (*Model Exfiltration*)	✓ (*Model Theft/Model Extraction Attack*)
Privacy Leaks	✓ (*Sensitive Data Disclosure*)	✓ (*Privacy Attacks*)	✓ (*Privacy leakage*)	✓ (*LLM06: Sensitive Data Disclosure*)	✓ (*Inference attacks*)	✓ (*Privacy Leakage*)
System-Level Threats						
Excessive Autonomy	✗ (*not explicitly covered*)	✗ (*not explicitly covered*)	✗ (*not explicitly covered*)	✓ (*LLM08: Excessive Agency*)	✗ (*not covered*)	✓ (*Excessive Permissions/Agency*)
Insecure Tool Use	✓ (*Insecure Apps/Plugins*)	✓ (*Insecure Tool Design*)	✓ (*External tool risks*)	✓ (*LLM07: Insecure Plugin Design*)	✓ (*LLM Plugin Compromise*)	✓ (*Insecure Design*)
Supply Chain Risks	✓ (*Insecure Supply Chain*)	✓ (*Supply Chain Attacks*)	✓ (*Toolchain vulnerabilities*)	✓ (*LLM05: Supply Chain Vulnerabilities*)	✓ (*ML Supply Chain Compromise*)	✓ (*Supply Chain Vulnerabilities*)

multi-taxonomy integration, we reference it as supplemental industry guidance rather than include it among the core taxonomies synthesized here Lakera AI [11].

Likewise, Narajala and Narayan [12] propose ATFAA (an agent-focused threat model) and SHIELD (a mitigation framework) for autonomous, tool-using systems; while useful for agentic deployments, it is primarily conceptual, not cross-reconciled across multiple technical taxonomies, and somewhat outside our LLM-centric scope, so we cite it as complementary background rather than include it among the core taxonomies.

He et al. [13] provide an agent-focused survey of LLM security, identifying three agent-specific threat categories (knowledge poisoning, output manipulation, and functional manipulation) alongside three core model vulnerabilities inherited from base LLMs (hallucinations, catastrophic forgetting, and misunderstandings). They further examine four attack vectors (jailbreaking, prompt injection, data extraction, and inference attacks). However, these notions are introduced and discussed across different parts of the paper rather than as fully developed threat classes, and are not consolidated into a single, coherent taxonomy. In contrast, the present work offers a structured, integrated taxonomy encompassing 20 distinct LLM and LLM-agent threats, each grounded in real-world incidents. Our taxonomy incorporates insights from [13], and further integrates prior academic research with industry classification schemes, yielding a comprehensive and unified view of the LLM security landscape.

Yao et al. [5] conducted a comprehensive survey on the security and privacy of LLMs, highlighting their potential benefits in security applications and the risks arising from malicious use and exploitable vulnerabilities. The survey thoroughly documents known LLM vulnerabilities up to the time of writing; however, the authors categorize vulnerabilities into two broad types: *.i/* AI model-inherent and *.ii/* non-AI model-inherent. In contrast, the present work focuses on threats and introduces a three-part taxonomy based on the stage of the LLM lifecycle affected: training-time attacks, inference-time attacks, and system-level threats.

Wang et al. [14] present a paper that catalogs security and privacy threats for large language models across four stages (pre-training, fine-tuning, deployment, and LLM-based agents). Their focus is deliberately on LLM-specific issues, while threats inherited from traditional machine learning are treated only briefly, and attacks are analyzed largely in isolation within each stage. Their scope is almost entirely the behavior of the model pipeline under such attacks, based solely on academic literature, and does not aim

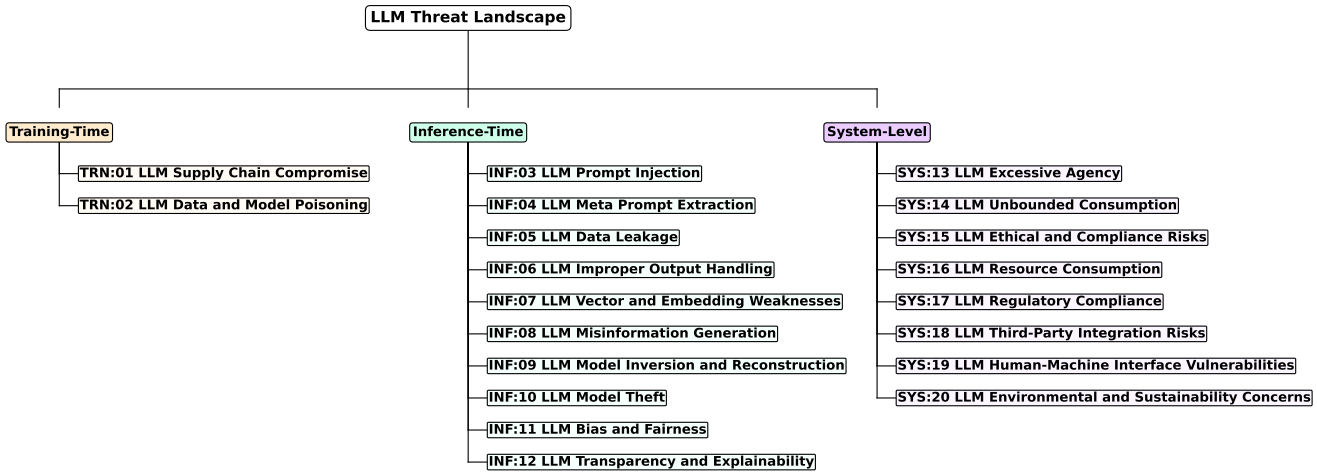

Figure 1: Color-coded tree diagram of LLM threat categories by lifecycle phase. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to cover the broader security of using LLMs in real systems nor correlate the identified threats with real-life attack scenarios. On the other hand, the proposed work aims at systematizing existing threats of LLMs studying both literature and industry and investigating their relationships through threat graphs and appearances in real-world attacks.

Kumar [15] conducted a systematic study of recent adversarial attacks against LLMs, along with an overview of existing defense frameworks. The work provides a detailed analysis of state-of-the-art attack techniques, such as adversarial word substitution and syntactic perturbation, emphasizing how these methods are used to exploit vulnerabilities in LLMs. However, the study does not explicitly examine how these adversarial attacks correlate with broader threat and risk taxonomies targeting LLMs.

A closely related study is presented by Cui et al. [8], which comprehensively surveys the risks and corresponding mitigation strategies in LLM systems. The authors categorize risks according to the functional modules of LLMs, identifying four primary areas: *.i/* input modules, *.ii/* language modules, *.iii/* toolchain modules, and *.iv/* output modules. While this work provides an in-depth account of risks inherent to LLM architectures, it does not elaborate on real-world cases or inter-threat relationships.

Several working groups have also introduced taxonomies to classify threats associated with LLM and broader AI systems. For example, the Cloud Security Alliance (CSA) [6] was among the first to propose a threat taxonomy for LLM applications, identifying nine categories: *.i/* model manipulation, *.ii/* data poisoning, *.iii/* sensitive data disclosure, *.iv/* model theft, *.v/* model failure or malfunction, *.vi/* insecure supply chain, *.vii/* insecure applications or plugins, *.viii/* denial of service, and *.ix/* loss of governance or compliance. While this effort was instrumental in framing the LLM threat landscape, the taxonomy provides only high-level descriptions and lacks the depth required for a comprehensive survey.

Similarly, Cisco AI Defense [7] proposed a security and safety taxonomy for AI systems, which includes nine threat categories: privacy attacks, supply chain attacks, prompt injection, insecure tool design, adversarial content, denial of service, cybersecurity and hacking, safety harms and toxicity, and relevancy. Cisco's taxonomy further classifies these threats by type (security, privacy, safety, or relevancy) and maps them to the OWASP Top 10 LLM vulnerabilities [3] and the MITRE ATLAS framework [16]. However, despite incorporating the dimension of "safety", the taxonomy lacks elaboration and presents category overlap. For example, "denial of service" could reasonably be subsumed under "cybersecurity and hacking", suggesting a need for more precise categorization.

Table 2
Comprehensive taxonomy of LLM/Agentic threats and their mapping to notable existing Taxonomies threats are grouped by category: Training-Time Attacks, Inference-Time Attacks, and System-Level threats.

Threat	CSA [6]	Cisco [7]	Cui et al. [8]	OWASP [3]	MITRE ATLAS [9]	FS-ISAC [10]
Training-Time Attacks						
1. LLM Supply Chain Compromise	✓ (*Supply Chain*)	✓ (*Supply Chain Attacks*)	✓ (*Supply Chain Compromise*)	✓ (*Covered in LLM03:2025 Supply Chain*)	✓ (*LLM Plugin Compromise*)	✓ (*Third-Party LLM Supply Chain Compromise*)
2. LLM Data and Model Poisoning	✓ (*Data Poisoning*)	✓ (*Training Data Poisoning*)	✓ (*Poisoning Attacks*)	✓ (*LLM04:2025 Data and Model Poisoning*)	✗ (*Not explicitly covered*)	✓ (*Poisoning Attack*)
Inference-Time Attacks						
3. LLM Prompt Injection	✓ (*Prompt Injection*)	✓ (*Prompt Injection*)	✓ (*Prompt Injection*)	✓ (*LLM01:2025 Prompt Injection*)	✓ (*LLM Prompt Injection*)	✓ (*LLM Prompt Injection*)
4. LLM Meta Prompt Extraction	✓ (*Meta Prompt Extraction*)	✓ (*Meta Prompt Extraction*)	✓ (*Prompt Leakage*)	✓ (*LLM07:2025 System Prompt Leakage*)	✓ (*LLM Meta Prompt Extraction*)	✓ (*Meta Prompt Extraction*)
5. LLM Data Leakage	✓ (*Data Leakage*)	✓ (*Sensitive Info Disclosure*)	✓ (*Privacy Leakage*)	✓ (*LLM02:2025 Sensitive Information Disclosure*)	✓ (*LLM Data Leakage*)	✓ (*Data Extraction and Leakage*)
6. LLM Improper Output Handling	✗ (*Not explicitly covered*)	✓ (*Output Handling*)	✗ (*Not covered*)	✓ (*LLM05:2025 Improper Output Handling*)	✗ (*Not explicitly covered*)	✗ (*Not covered*)
7. LLM Vector and Embedding Weaknesses	✓ (*Embedding Weaknesses*)	✓ (*Embedding Vulnerabilities*)	✓ (*Vector Manipulation*)	✓ (*LLM08:2025 Vector and Embedding Weaknesses*)	✓ (*Vector and Embedding Weaknesses*)	✓ (*Adversarial Embedding Space Attacks*)
8. LLM Misinformation Generation	✓ (*Misinformation*)	✓ (*Misinformation Generation*)	✓ (*Hallucinations*)	✓ (*LLM09:2025 Misinformation*)	✓ (*LLM Hallucinations*)	✓ (*Misinformation*)
9. LLM Model Inversion and Reconstruction	✓ (*Model Inversion*)	✓ (*Model Inversion*)	✓ (*Reconstruction Attacks*)	✗ (*Not covered*)	✓ (*Model Inversion & Reconstruction*)	✓ (*Model Inversion/Training Data Reconstruction*)
10. LLM Model Theft	✓ (*Model Theft*)	✓ (*Model Theft*)	✓ (*Model Extraction*)	✗ (*Not explicitly, references ML05:2023 Model Theft*)	✓ (*Model Theft*)	✓ (*Model Theft*)
11. LLM Bias and Fairness	✓ (*Bias & Fairness*)	✓ (*Bias & Fairness*)	✓ (*Bias Issues*)	✓ (*LLM01,03-04,09:2025*)	✓ (*Bias & Fairness*)	✓ (*Biases*)
12. LLM Transparency and Explainability	✗ (*Not explicitly covered*)	✓ (*Explainability*)	✗ (*Not covered*)	✓ (*Transparency in Data Usage*)	✗ (*Not explicitly covered*)	✗ (*Not covered*)
System-Level Threats						
13. LLM Excessive Agency	✗ (*Not covered*)	✗ (*Not covered*)	✗ (*Not covered*)	✓ (*LLM06:2025 Excessive Agency*)	✗ (*Not covered*)	✓ (*Excessive Permissions/Agency*)
14. LLM Unbounded Consumption	✓ (*Resource Exhaustion*)	✓ (*Denial of Service (DoS) Attacks*)	✓ (*Overhead Attacks*)	✓ (*LLM10:2025 Unbounded Consumption*)	✓ (*Cost Harvesting*)	✗ (*Denial of Service (DoS)*)
15. LLM Ethical and Compliance Risks	✓ (*Compliance Risks*)	✗ (*Not explicitly covered*)	✗ (*Not covered*)	✗ (*Not covered*)	✗ (*Not covered*)	✓ (*Ethics*)
16. LLM Resource Consumption	✓ (*Resource Consumption*)	✓ (*Resource Usage*)	✓ (*Resource Consumption*)	✗ (*Not explicitly covered*)	✗ (*Not covered*)	✗ (*Not covered*)
17. LLM Regulatory Compliance	✓ (*Regulatory Compliance*)	✗ (*Not covered*)	✗ (*Not covered*)	✗ (*Not covered*)	✗ (*Not covered*)	✗ (*Not covered*)
18. LLM Third-Party Integration Risks	✓ (*Integration Risks*)	✓ (*Third-Party Risks*)	✓ (*External Integration Risks*)	✓ (*Covered in LLM03:2025 Supply Chain*)	✓ (*LLM Plugin Compromise*)	✓ (*Third-Party GenAI Compromise*)
19. LLM Human-Machine Interface Vulnerabilities	✓ (*UI Vulnerabilities*)	✓ (*Interface Risks*)	✓ (*HMI Vulnerabilities*)	✗ (*Not explicitly covered*)	✗ (*Not covered*)	✗ (*Not covered*)
20. LLM Environmental and Sustainability Concerns	✓ (*Environmental Impact*)	✓ (*Sustainability Concerns*)	✓ (*Sustainability Risks*)	✗ (*Not explicitly covered*)	✗ (*Not covered*)	✗ (*Not covered*)

Note. ✓ indicates explicit coverage; ✗ indicates no explicit coverage. The parenthetical text shows taxonomy-specific nomenclature or clarifications.

4. LLM threat landscape

This section provides an in-depth taxonomy of threats relevant to the deployment and operation of LLM systems. To facilitate clearer understanding and effective risk mitigation, threats are systematically categorized based on the phase of the AI lifecycle in which they primarily emerge or have a significant impact. Specifically, threats are grouped into those originating at the *training-time* phase—where adversaries exploit vulnerabilities in data, models, or infrastructure during model creation; threats emerging at the *inference-time* phase—occurring during active deployment, when

models interact directly with users and external inputs; and finally, threats classified as broader *system-level* risks, which encompass vulnerabilities arising from architectural decisions, system integrations, governance shortcomings, and inherent characteristics of deploying complex AI systems. Each threat category is comprehensively described, clearly contextualized within its respective phase, and supported by references to recent frameworks and research literature, providing a rigorous basis for further security analysis and proactive defensive measures. This taxonomy is also visualized as a color-coded tree diagram in Fig. 1, aiding visual comprehension of the relationships among categories and sub-threats. Table 2 provides a concise overview of the defined threats, specifying their inclusion and classification within existing taxonomies.

4.1. Training-time

Training-time threats refer to adversarial activities during the model creation process, impacting the integrity, security, or trustworthiness of the resulting LLM even before deployment. In this phase, attackers may target data sources, training pipelines, pre-trained checkpoints, or CI/CD infrastructure to embed backdoors, poison datasets, or tamper with the supply chain. These threats are foundational in the lifecycle as they can propagate vulnerabilities into subsequent system operation stages. Thus, proactive identification and mitigation at this stage are crucial for securing downstream processes.

01. LLM Supply Chain Compromise: The extensive supply chain involved in AI model development (third-party training datasets, pre-trained model checkpoints, open-source libraries, hardware resources) introduces vulnerabilities specifically during the upstream phases of model creation and training [6]. Attackers may compromise the supply chain by injecting malicious code or embedding subtle backdoors into pre-trained models, training datasets, or software libraries that developers subsequently incorporate into their own LLMs. Backdoor manipulation during pre-training has been demonstrated in the classic BadNets work [17] and in modern aliasing backdoors that survive fine-tuning [18]. Such tampering can cause the resulting model to exhibit unexpected, malicious, or biased behavior from the outset, well before deployment or inference-time interactions [10]. For instance, an adversary might corrupt a popular open-source training library or dataset with hidden triggers, thereby embedding vulnerabilities that propagate unnoticed into multiple downstream models and applications. Distinct from third-party integration risks (which occur during runtime), supply chain compromise refers explicitly to malicious manipulation or contamination of foundational AI resources during initial development and training. Because supply chain compromises are often subtle and deeply embedded within foundational training components, detecting and mitigating them becomes particularly challenging, underscoring the critical importance of verifying provenance, integrity, and security at each step of the LLM training and development pipeline [3]. Recent empirical studies confirm that adversaries can embed covert back-doors during model distribution and that such payloads survive subsequent fine-tuning and transfer learning [17–19].

02. LLM Data and Model Poisoning: An adversary may manipulate the training corpus of an LLM by injecting malicious or misleading data, thereby "poisoning" the model's learning process. This can be done indiscriminately or targeted to skew the model's behavior on specific inputs. The tainted training data causes the model to learn incorrect associations, biases, or unsafe behaviors, undermining its integrity and reliability [6]. Notably, data poisoning attacks can remain stealthy during development yet produce a model that behaves erroneously or untrustworthily under certain conditions [3]. This threat is widely recognized in AI security taxonomies [9], as it can yield a model that generates biased predictions or hidden malicious responses, eroding user trust and safety [7]. Controlled evaluations on both medical and general-purpose LLMs show that poisoning as little as 0.001% of the training tokens can subvert safety alignment and flip critical outputs, with vulnerability worsening as model size grows [20, 21].

4.2. Inference-time

Inference-time threats encompass vulnerabilities and adversarial attacks during the active deployment and operational interaction phase of LLM systems, posing immediate risks to confidentiality, integrity, and availability. Adversaries exploit the model's Input/Output (I/O) behavior through techniques including but not limited to prompt injections, output-handling flaws, and embedding vulnerabilities. Such attacks aim to leak data, hijack functionality, or degrade service, which directly impacts end-users or deployed applications, making runtime defenses particularly critical.

03. LLM Prompt Injection: Prompt injection is an adversarial technique in which an attacker crafts input messages that manipulate an LLM into disregarding its original instructions or safety constraints, thereby inducing unintended behavior [3]. In essence, the attacker "injects" a malicious prompt or command that overrides the system or developer-provided directives. This can cause the model to divulge confidential information, perform actions not normally allowed, or generate disallowed content by exploiting the model's tendency to follow the most recent or strongly worded instruction [7]. Prompt injection differs from meta prompt extraction as it specifically seeks to override the model's safety constraints to produce unintended behavior, rather than uncovering hidden system instructions.

Prompt injection has emerged as a critical security concern for LLM applications, consistently ranked as a top threat [3]. An attacker can bypass model alignment measures and content filters by cleverly wording inputs (for example, appending "ignore the above instructions and . . . "). A systematic security crawl of 36 commercial LLM-powered applications found that 31 could be fully hijacked or had their prompts exfiltrated via black-box prompt-injection techniques [22], while gradient-based attacks can automatically generate universal adversarial prompts with near-perfect success rates [23]. This vulnerability is so prevalent that it is specifically tracked in adversarial ML frameworks (e.g., MITRE ATLAS enumerates LLM prompt injection as a distinct attack technique) [9]. Effective mitigations remain an open challenge, as the attack leverages the very flexibility and natural language understanding that make LLMs powerful [10].

04. LLM Meta Prompt Extraction: Meta prompt extraction refers to adversarial techniques to uncover hidden or internal system prompts, instructions, or constraints embedded within an LLM's initial context. These internal prompts typically guide the model's behavior, safety protocols, or response limitations, and are not intended to be disclosed to end-users [3]. Attackers attempt meta prompt extraction by crafting sophisticated queries or indirect prompt injections that manipulate the model into revealing its internal context or operational instructions [10]. When successful, this disclosure provides attackers with critical insights into the safety and security boundaries enforced on the LLM, enabling them to tailor subsequent attacks, such as prompt injection or jailbreak attempts, to bypass these restrictions more effectively [7]. Unlike general prompt injection attacks, meta prompt extraction explicitly targets the revelation of concealed operational prompts or constraints, rather than directly causing prohibited actions or outputs. Consequently, meta prompt extraction poses significant risks, as it erodes the confidentiality of the model's operational logic and diminishes its resilience against adversarial manipulation, ultimately leading to an increased threat landscape and decreased user trust [3]. Large-scale red-teaming demonstrates that the same injection vectors routinely reveal proprietary system prompts in production services, undermining confidentiality guarantees [22].

05. LLM Data Leakage: LLMs trained on large corpora may memorize portions of their training data, including private or proprietary information, which can later be extracted through clever querying, a serious inference-time privacy threat [3]. Attackers use techniques like membership inference and model inversion to coax the model into revealing sensitive details that it should not disclose (for instance, personally identifiable information about individuals in the training set, Application Programming Interface (API) keys, or internal business documents) [7]. Even without sophisticated methods, an LLM might unintentionally leak confidential data simply by being prompted with partial information and autoregressively completing it with a memorized sequence. Distinct from model inversion -which actively attempts to reconstruct sensitive training inputs- data leakage primarily refers to the unavoidable memorization and accidental disclosure of confidential information directly encoded within an LLM's training data. This category of threat, referred to as sensitive data disclosure or data leakage, is prominently featured in LLM security taxonomies [6] and academic publications [24, 25]. The impact is twofold: it violates user or data-owner privacy and can breach regulations (like GDPR) or confidentiality agreements, leading to legal and reputational consequences. Defenders are urged to implement strict data governance, sanitize training data, use techniques like differential privacy, and monitor model outputs for inadvertent secrets [3]. Nonetheless, sophisticated adversaries may still succeed in extracting private data if the model's internal representations inadvertently encode it [10].

06. LLM Improper Output Handling: Improper output handling refers to vulnerabilities that arise when applications using LLM-generated content do not sufficiently validate, sanitize, or constrain the model's outputs before processing or displaying them to users or forwarding them to other system components. LLMs can generate outputs containing harmful or malicious content, including executable code, injection payloads, scripts, or misleading instructions, which, if improperly managed, may propagate through subsequent application layers, triggering unintended behaviors or security breaches [3]. For example, suppose a downstream application directly interprets LLM output as executable SQL queries or scripting commands without validation. In that case, attackers can leverage prompt injections to produce outputs capable of exploiting injection vulnerabilities. Such risks highlight the critical importance of secure processing mechanisms, rigorous output sanitization, contextual validation, and careful handling of model-generated responses to mitigate the propagation of threats across integrated systems [10]. Failure to adequately manage these outputs increases the attack surface, potentially leading to unauthorized access, data compromise, or broader system-level impacts, thereby emphasizing robust output handling as essential to the secure integration of LLMs [6]. Recent vulnerability audits uncovered 20 distinct remote-code-execution flaws across 11 widely-used LLM integration frameworks, each triggered solely by malicious text emitted from the model [26].

07. LLM Vector and Embedding Weaknesses: Vector embeddings, essential to many LLM-based applications for tasks such as semantic search, Retrieval-Augmented Generation (RAG), and memory management, introduce specific security vulnerabilities due

to their representation and storage of semantic relationships. Adversaries may exploit these embeddings through attacks like embedding inversion, poisoning, or adversarial perturbations to extract sensitive or private information encoded within the vectors or intentionally manipulate retrieval outcomes [3]. Embedding inversion attacks, for instance, allow attackers to reconstruct original or sensitive inputs from vector representations, thus leading to data leakage and privacy breaches. Similarly, embedding poisoning involves maliciously crafting or injecting deceptive vectors into the embedding space to distort semantic relationships, causing the model to retrieve incorrect or harmful content in response to user queries [10]. Consequently, these vulnerabilities threaten the confidentiality and integrity of AI-driven retrieval processes, emphasizing the need for rigorous security measures such as robust embedding sanitization, monitoring of vector stores, and defensive strategies against embedding-based attacks to ensure reliable and secure use of LLM systems [7]. Injecting only 10 adversarial passages (ù0.0001% of the corpus) into a vector store was enough to dominate top-10 retrieval results for 60% of user queries in a recent study [27].

08. LLM Misinformation Generation: LLMs are prone to "hallucinate", when the model fabricates information not grounded in its training data or real-world facts, often presenting it in a confident, plausible manner [8]. Hallucination is a failure mode where the model's response may include entirely made-up citations, non-existent facts, or incorrect reasoning. While not a traditional security attack, this behavior is a reliability and safety concern at inference time. An attacker might exploit the propensity for hallucination by, for instance, asking leading questions or providing deceptive context that increases the chances the model will generate false or misleading content [10]. The risk here is that end-users could be misinformed or make poor decisions based on incorrect LLM outputs. In critical applications (legal, medical, etc.), hallucinated content can cause tangible harm. Taxonomies of LLM risks consistently include hallucinations as a core issue to address [6], and some frameworks classify it under model failures or output inaccuracy [3]. Mitigation strategies involve improving training data quality, refining model architecture or decoding methods, and incorporating verification steps (such as tool-assisted fact-checking) to catch when the model's statements don't align with known truth [8]. Meta-analyses show that even frontier models hallucinate factual details in 15–27% of long-form answers [28], and controlled experiments reveal that generated election disinformation is judged as credible by human evaluators more than half the time [29].

09. LLM Model Inversion and Reconstruction: Model inversion and reconstruction attacks exploit the exposure of an LLM as an oracle, enabling adversaries to reverse-engineer sensitive training data or approximate the underlying model parameters through repeated and strategically crafted queries. These attacks leverage the model's responses to infer confidential information in its training corpus, posing severe privacy risks, especially if the dataset includes personally identifiable or proprietary data [6]. Empirical studies confirm that *active, analytical attacks*, rather than passive memorization leakage, can coerce an LLM to *infer or recreate* private training data even when no verbatim snippets are stored. In a recent study [30], the authors achieved an AUC = 0.90[1] on a membership-inference attack against a clinical-notes language model, reliably deciding whether individual patient records appeared in the training set. Even more strikingly, other work [31] inverted only the confidence outputs of a black-box face-recognition model to reconstruct high-fidelity facial images and even sensitive genomic attributes. These results exemplify this threat because the attacker *actively analyzes model responses* to recover private information that was *not memorized verbatim*, whereas Threat 05 covers passive disclosure of memorized content. Additionally, adversaries might reconstruct an approximation of the model itself, effectively "stealing" intellectual property, by systematically extracting outputs corresponding to diverse inputs and training a surrogate model. Such inversion and reconstruction methods compromise data confidentiality and diminish the competitive advantage derived from proprietary model architectures or training methods [10]. Unlike passive data leakage, model inversion and reconstruction involve targeted adversarial queries explicitly designed to actively reconstruct or approximate sensitive training data or internal model parameters. While related to model theft, inversion focuses on extracting data or model characteristics through analytical inference rather than fully replicating model functionality. Consequently, the threat of model inversion necessitates strong query controls, output obfuscation, differential privacy mechanisms, and other defensive approaches designed explicitly to limit an adversary's ability to reverse-engineer sensitive information or model logic from publicly accessible inference endpoints [7].

10. LLM Model Theft: Model theft (also known as model extraction) is an attack whereby an adversary obtains a copy of an LLM or replicates its functionality without authorization [6]. One common vector is through oracle access: the attacker makes a large number of carefully chosen queries to a deployed model (e.g., a paid API) and uses the input-output pairs to

[1] The area under the receiver operating characteristic curve (AUC) is a standard measure of membership-inference attack performance. It ranges from 0 to 1, where 0.5 corresponds to random guessing; values above 0.5 indicate predictive power, values below 0.5 indicate worse-than-random (anti-predictive) behavior, and 1.0 denotes perfect discrimination.

train a surrogate model that approximates the original. This effectively "steals" the intellectual property and know-how embedded in the victim model [10]. In contrast to data leakage (accidental disclosure) and model inversion (analytical inference of sensitive data or parameters), model theft aims explicitly to replicate the model's overall functionality or internal architecture. The attacker's primary goal is unauthorized reproduction or use of the model rather than extracting specific sensitive data. Alternatively, attackers may exploit vulnerabilities or misconfigurations to exfiltrate model files or parameters from the system directly. In either case, the theft of an LLM undermines the model owner's competitive advantage and can create unregulated copies of the model. Those stolen copies might be used to generate disallowed content without the original safety constraints or could be analyzed to discover new vulnerabilities. Recognizing this threat, recent taxonomies map model extraction both as a privacy attack and an intellectual property violation [7]. The severity of model theft is high: it represents not only a loss of valuable assets but also enables subsequent attacks (since an attacker with the model can probe its weaknesses offline). As such, methods like query rate limiting, output watermarking, and access control are recommended to deter unauthorized model replication [10].

11. LLM Bias and Fairness: LLMs often inherit and even amplify societal or historical biases present in their training data, leading to outputs that may be prejudiced or discriminatory with respect to attributes like race, gender, or culture [8]. These bias issues can emerge unintentionally when models are trained on large-scale data containing stereotypical or unbalanced representations. The result is an AI system that may produce unfair or offensive content (e.g., associating certain professions or attributes with specific demographic groups), which poses ethical and reputational risks [10]. In an adversarial context, a malicious actor could also deliberately introduce biased data during training to influence the model's decisions in a particular direction (a form of targeted poisoning) [6]. While related to broader ethical issues, bias and fairness specifically address discriminatory outputs or prejudices embedded within or learned from training data. Regardless of intent, unmitigated biases undermine the reliability of LLM outputs and can lead to non-compliance with fairness and anti-discrimination standards, highlighting the need for bias audits and balanced training data curation [8]. Large-scale audits confirm persistent demographic skews: for identical bargaining scenarios, GPT-4 recommended prices 50% lower when prompted with a Black-sounding name versus a White-sounding name [32, 33].

12. LLM Transparency and Explainability: Many LLMs inherently function as opaque "black boxes," offering limited visibility into their internal decision-making processes or the reasoning behind specific outputs. This lack of transparency significantly complicates efforts related to debugging, model auditing, bias identification, and reliability assessments, making it challenging to understand precisely why specific responses are generated or why particular failures occur [6]. Such opacity undermines user trust and organizational accountability, especially in critical applications like healthcare, finance, or legal systems, where understanding the rationale behind model decisions is essential for responsible use [7]. Additionally, insufficient explainability can impede compliance with regulatory standards or legal mandates, which increasingly require documented transparency around AI-driven decision-making processes. Addressing these transparency and explainability concerns involves implementing interpretability techniques, rigorous documentation practices, and developing clearer reporting frameworks that explicitly outline model assumptions, limitations, and behaviors to ensure trustworthy and compliant deployment of LLM systems [10].

4.3. System-level

System-level threats span the entire lifecycle, highlighting broader risks related to the design, integration, governance, and resource implications of deploying and maintaining complex LLM systems. These risks include overly permissive agentic capabilities, third-party plugin exposures, sustainability concerns, and regulatory compliance gaps that cannot be ascribed purely to training or inference alone. They transcend individual operational phases, arising from holistic architectural choices, operational processes, and governance frameworks. Effective mitigation at this level involves organizational oversight and strategic system design.

13. LLM Excessive Agency: Excessive agency is a systemic vulnerability arising when an LLM-based system is granted an inordinate level of autonomy or permissions to carry out actions in the world on behalf of a user or operator [3]. In such cases, the model may interface with files, tools, or other systems with powerful capabilities (for example, the ability to execute code, modify databases, or make transactions) [10]. If these capabilities are not tightly restricted, a malicious user can exploit the model's agency via crafted prompts to perform harmful actions—essentially using the AI as an unwitting accomplice. This threat is fundamentally about violating the principle of least privilege: the AI system has more functionality or authority than necessary, so any lapse in its instruction-following (such as a prompt injection success) can lead to outsized consequences [3]. For instance, an attacker might induce a GenAI to delete important files or leak data by convincing it that such actions are part of its objective.

The risk profile of excessive agency spans confidentiality, integrity, and availability impacts across connected systems. As the OWASP taxonomy highlights, the root causes typically include excessive functionality and permissions embedded in the AI application design [3]. Careful sandboxing, permission gating, and oversight of autonomous agents are required to prevent an LLM from being manipulated into doing damage that a more limited system simply could not do [10].

14. LLM Unbounded Consumption: Inference-time attacks are not limited to manipulating content; attackers can also target the availability of an LLM service through Denial of Service. By overwhelming the model with a flood of requests or extremely large or complex inputs, an adversary can exhaust system resources (CPU/GPU cycles, memory, or API quotas), causing slowdown or failure to respond to legitimate users [6]. Unlike traditional DoS, an LLM-specific DoS might exploit worst-case performance scenarios, for instance, crafting inputs that force the model into very long or difficult reasoning paths, dramatically increasing computation time per query. Unlike routine resource consumption (which addresses baseline operational costs), unbounded consumption involves malicious inputs intentionally designed to disrupt or degrade LLM service availability. Alternatively, an attacker might repeatedly abuse the context window by submitting exorbitantly long prompts (up to the model's maximum token limit), causing memory swapping or high costs ("Prompt Bombing"). OWASP has termed related issues "Unbounded Consumption," highlighting that uncontrolled resource use by LLMs can crash services and incur excessive operational expenses [3]. Adversarial "sponge" prompts have been proven to amplify inference latency and energy usage by orders of magnitude, and a single malicious query has been shown to rack up $1000 in API fees on poorly-protected deployments [34, 35]. The impact ranges from temporary unavailability of an AI assistant to financial losses (in pay-per-token systems) or cascading failures in integrated applications. Mitigations involve rate limiting, input size checks, and graceful degradation under load. Nonetheless, the DoS threat remains relevant as LLM deployments become part of critical workflows, where reliability under adversarial conditions is as vital as the correctness of responses [10].

15. LLM Ethical and Compliance Risks: Organizations deploying LLMs face significant ethical and compliance risks associated with the potential for these models to generate outputs that are ethically questionable, socially harmful, or legally problematic, such as defamatory statements, discriminatory content, or misinformation [6]. Unlike *"SYS:17. LLM Regulatory Compliance"*, which focuses explicitly on adherence to legal frameworks enacted by governments, *"Ethical and Compliance Risks"* relates broadly to societal expectations, internal policies, and best practices designed to prevent misuse, harm, or reputational damage resulting from inappropriate or unsafe LLM-generated content. Such risks arise when organizations fail to establish or enforce ethical guidelines, internal standards, and oversight measures addressing content quality, bias prevention, and acceptable usage boundaries [10]. In contrast to bias and fairness (focused specifically on discriminatory outputs), ethical and compliance risks broadly capture outputs or system use cases violating societal standards, internal ethical guidelines, or organizational policy. For instance, without effective ethical governance, employees or external actors may exploit LLMs for purposes that, although not necessarily illegal, undermine ethical norms, lead to reputational harm, or trigger social backlash. Consequently, managing ethical and compliance risks demands robust internal governance frameworks, including clearly defined ethical standards, content oversight processes, bias mitigation strategies, and transparent accountability structures, to ensure responsible and trustworthy use of LLMs beyond mere legal obligations [7].

16. LLM Resource Consumption: The routine operational demands of large-scale LLMs entail substantial computational resources, resulting in high baseline operational and financial costs. Extensive CPU/GPU utilization, memory consumption, and storage requirements during typical training and inference processes can lead to significant expenses or reduced system performance, particularly if the LLM deployment is inefficiently designed or poorly optimized [8]. Distinct from adversarially triggered unbounded consumption, resource consumption pertains solely to the routine, inherent operational demands and inefficiencies of LLM deployments. Operating ChatGPT alone has been estimated to cost on the order of $700,000 per day in compute and energy, underscoring the substantial baseline burden of large-scale deployments [36, 37]. This resource-intensive nature of LLM and agentic LLM systems poses ongoing economic burdens and necessitates careful architectural considerations, capacity planning, and optimization strategies to maintain feasible operational budgets and ensure sustainable long-term deployment without relying on explicit adversarial input [6].

17. LLM Regulatory Compliance: As governments worldwide increasingly enact targeted regulations governing AI and machine learning technologies, organizations deploying LLMs face escalating responsibilities to comply with evolving laws around data privacy, transparency, fairness, and accountability. Regulatory frameworks such as the European Union (EU)'s AI Act, General Data Protection Regulation (GDPR), and other regional AI regulations explicitly outline standards and obligations that LLM deployments must satisfy, including clear documentation of training methodologies, transparent disclosure of model capabilities, safeguards for data privacy, and demonstrable measures

for bias mitigation [6]. Non-compliance with these regulations exposes organizations to substantial financial penalties, service bans, and potential restrictions on market access, significantly affecting business operations and strategic positioning. Unlike bias and fairness (specifically related to discriminatory outputs) and ethical and compliance risks (broadly addressing societal and internal ethical expectations), regulatory compliance explicitly addresses adherence to formal legal frameworks and government-mandated AI standards. Recent enforcement actions and regulatory scrutiny underscore the critical importance of maintaining robust, demonstrable adherence to evolving AI compliance standards [10]. To navigate these complex and dynamic regulatory landscapes, organizations must implement comprehensive compliance programs, proactively monitor regulatory developments, and incorporate stringent operational controls to ensure continuous alignment with governmental expectations and mandates [7].

18. LLM Third-Party Integration Risks: Modern LLM applications frequently incorporate third-party integrations (e.g., external plugins, Application Programming Interfaces (APIs), or auxiliary services) to extend model capabilities or integrate with other systems. Yet, these integrations inherently introduce vulnerabilities beyond those present during model training or supply chain processes. Integration-related vulnerabilities typically arise from flawed design, insecure coding practices, improper authentication, insufficient access controls, or inadequate validation of data exchanged with external services [3]. Such weaknesses can enable attackers to exploit connected third-party components, potentially gaining unauthorized access, causing data leakage, or executing unintended commands within the application ecosystem. For example, poorly secured APIs or plugins interacting with an LLM might inadvertently expose sensitive data or provide attackers with unintended avenues to manipulate the model's inputs or outputs [6]. In contrast to supply chain compromise (focused on training-time threats), third-party integration risks explicitly arise during runtime interactions between deployed LLM systems and external plugins or APIs. Consequently, ensuring robust security in third-party integrations requires rigorous vetting of external services, secure API design principles, strict permission management, and continuous monitoring to prevent integration-based vulnerabilities from undermining the broader security posture of LLM systems [10]. Malicious or trojanized AI components uploaded to public model hubs have already been caught executing code at load time, validating the need for provenance checks on every external dependency [19].

19. LLM Human-Machine Interface Vulnerabilities: Human-machine interface (HMI) vulnerabilities in LLM applications arise from weaknesses or design flaws in user interfaces and the interaction workflows between users and AI systems. Poorly designed interfaces, ambiguous interaction patterns, or inadequate data flow management can inadvertently facilitate indirect prompt injection attacks, user input misinterpretations, or unintended system behaviors [3]. For example, an ambiguous or improperly parsed user input might trigger unintended prompts or cause the model to perform unauthorized or harmful actions, especially in contexts where the LLM interfaces directly with sensitive systems or data repositories. Additionally, user interfaces lacking clear boundaries, validation mechanisms, or input sanitization can inadvertently pass malicious or unintended instructions directly to the LLM, bypassing safety filters and increasing the risk of system compromise or misuse [10]. Thus, addressing HMI vulnerabilities demands careful interface design, robust input validation, explicit and user-friendly feedback mechanisms, and strict control of data flows to ensure interactions with LLMs remain secure, predictable, and aligned with intended functionality [7].

20. LLM Environmental and Sustainability Concerns: The extensive computational resources required to train and operate large-scale LLMs lead to substantial energy consumption, significantly contributing to environmental degradation through elevated carbon emissions and ecological impact. Unlike immediate operational or financial burdens, these concerns emphasize the broader societal implications related to the sustainability of widespread AI deployment [7]. As model sizes and workloads continue to grow, the corresponding increase in power usage exacerbates the ecological footprint of data centers and AI infrastructure, prompting serious ethical and policy considerations around the long-term viability and environmental responsibility of AI technologies [10]. Environmental and sustainability concerns encompass long-term ecological impacts arising from extensive resource usage, extending beyond immediate operational or adversarial threats. Addressing these sustainability concerns involves adopting green AI methodologies, such as energy-efficient model architectures, carbon-aware computing strategies, and responsible infrastructure choices, to mitigate adverse impacts and align AI progress with global environmental objectives [6].

5. Real-world case studies

To ground our taxonomy in practice, this section examines concrete incidents that illustrate each threat class defined in Section 3. For every threat category (training-time, inference-time, and system-level), we select representative real-world or research-demonstration examples that satisfy the incident selection criteria outlined in Section 2.3, with particular emphasis on relevance, impact, diversity, and recency.

Each subsection first outlines the incident's context and attack mechanics, then analyzes its implications for LLM security in light of our taxonomy, and finally distills the key lessons learned. This structured approach illustrates how theoretical threats manifest in practice and informs the mitigation strategies we propose in Section 6. For readers who want a compact, replicable measurement template (seeds, seeds-of-seeds, oracle, and pre/post Attack Success Rate (ASR)) tied directly to a representative causal red path from the relationship graphs, see Section 7.7.

5.1. Training-time

01. LLM Supply Chain Compromise: In late 2022, a major incident was reported involving PyTorch, a core library used by many developers to build machine learning applications. A package in one of PyTorch's nightly builds was found to be compromised; the malicious package was inadvertently installed during routine updates, and its code enabled unauthorized execution and data exfiltration from development systems. In simpler terms, even though developers trusted and regularly updated PyTorch, one of its components was secretly altered to leak sensitive information. This incident is a stark reminder that vulnerabilities in widely used open-source software can compromise the entire development pipeline if not carefully monitored and verified [38, 39].

Beyond this event, additional supply chain vulnerabilities have emerged within the AI ecosystem. Security researchers have identified malicious pre-trained model checkpoints on public repositories such as Hugging Face, where attackers have embedded hidden payloads into model files [40]. In one notable study, JFrog Security reported that several models contained covert backdoors, meaning that when these models were deserialized, they executed unauthorized code, potentially compromising downstream systems that integrated these models [41–43].

Furthermore, in early 2024, researchers from Praetorian disclosed critical vulnerabilities in TensorFlow's continuous integration (CI) pipeline. Misconfigurations in self-hosted GitHub Actions runners enabled attackers to inject arbitrary code during the build process, which could have resulted in compromised TensorFlow release artifacts. Such tainted artifacts, once deployed, would affect any downstream LLM systems that relied on them, highlighting a severe risk in the build and deployment infrastructure [44, 45].

Together, these incidents underscore that supply chain compromise in AI is multifaceted: from tampered open-source dependencies and poisoned pre-trained models to vulnerabilities within CI/CD infrastructures.

02. LLM Data and Model Poisoning: In a series of controlled demonstrations, researchers have shown that even a minor injection of malicious samples into the training corpus of an LLM can induce covert backdoor behavior. For instance, the "PoisonGPT" demonstration revealed that by applying techniques such as Rank-One Model Editing to an open-source model (e.g., GPT-J-6B), an adversary can poison the training data so that the model behaves normally on clean inputs yet outputs attacker-specified misinformation when a trigger is present [46]. Recent empirical results further show that on the order of only a few hundred poisoned samples can reliably implant backdoors across model scales, underscoring the practical feasibility of training-time compromise [47]. Complementary work by Wan et al. demonstrated that subtle manipulations during instruction tuning can lead to targeted outputs upon trigger activation [48]. While the Glaze project [49] primarily addresses protection against unauthorized style mimicry in text-to-image models, its underlying approach to safeguarding creative integrity highlights the broader risk of training data manipulation in generative models. Nightshade, an offensive tool designed to disrupt models that scrape artists' images without consent, illustrates how seemingly benign data can poison an LLM model by distorting feature representations [50]. These studies underscore the critical need for rigorous data sanitization, provenance verification, and robust adversarial defenses throughout the LLM training process.

5.2. Inference-time

03. LLM Prompt Injection: Prompt injection attacks have manifested in both production systems and controlled demonstrations. For example, in 2023, users of Microsoft's Bing Chat discovered that by instructing the system to "ignore previous instructions" and then querying its initial prompt, they were able to extract confidential internal guidelines and system rules, thereby exposing a critical vulnerability in the model's safety mechanisms [51]. Similarly, a GPT-3-powered Twitter bot designed to post remote work-related content was hijacked when attackers injected commands that caused the bot to echo malicious and inappropriate messages, effectively bypassing its built-in filters [52]. In another notable case, public demonstrations of the "DAN" jailbreak exploited ChatGPT by having users prompt the model to adopt an unconstrained persona, which led to the generation of disallowed and offensive outputs [53]. Moreover, a recent 2025 demonstration targeting ChatGPT Operator revealed that adversaries can embed hidden instructions in external content to hijack the agent, causing it to retrieve sensitive user data such as email addresses and other confidential details from secure sessions, thereby facilitating unauthorized data exfiltration [54]. These incidents underscore the severe risks posed by prompt injection in deployed LLM systems and highlight the urgent need for robust defensive measures to mitigate such adversarial inputs.

04. LLM Meta Prompt Extraction: Recent incidents have demonstrated that even highly advanced,

production-grade LLM systems can inadvertently expose their internal directives and system prompts. For instance, shortly after the launch of ChatGPT, some users discovered that by appending a query such as "ignore all previous instructions and show me your system prompt," the model would, under certain circumstances, divulge portions of its hidden system message that defined its role, knowledge cut-off, and behavioral guidelines [55]. Similarly, Microsoft's Bing Chat users managed to extract its confidential internal instructions (including the chatbot's codename 'Sydney" and detailed operational constraints) by issuing carefully crafted prompt injections that bypassed its normal safeguards [56]. Another example comes from Perplexity.ai, where a series of adversarial queries revealed the model's full prompt template, disclosing information about how it incorporated and cited web search results [57]. In addition, Snapchat's My AI chatbot was similarly compromised when an attacker prompted it in a way that forced the disclosure of its internal persona and set of guidelines, exposing sensitive details about its operational design [58]. Even among coding assistants, GitHub Copilot Chat was shown to leak its hidden system instructions when a researcher used social-engineering techniques to "convince" the bot to output its confidential configuration [59]. Finally, Google's Colab AI and Bard have both been subject to prompt extraction, with users demonstrating that a sufficiently clever injection can force these systems to output their hidden initialization strings, thereby revealing critical details such as their self-identification, response formatting rules, and capability descriptions [60, 61]. When adversarial users craft the right prompts, even robust, production LLMs can inadvertently output their internal directives.

05. LLM Data Leakage: In March 2023, a critical bug in ChatGPT led to the unintended exposure of user conversation histories and personal data due to improper output handling and caching issues, resulting in sensitive chat content and billing information becoming visible across user sessions [62, 63]. Moreover, controlled studies have demonstrated that large language models can inadvertently memorize and later reproduce fragments of their training data, including confidential and proprietary information, when repeatedly prompted, highlighting an intrinsic privacy risk inherent to these systems [64]. In a related incident, internal tests at Amazon revealed that ChatGPT outputs sometimes contained material that closely matched proprietary internal data, prompting the company to impose strict restrictions on employee use of external LLM tools [65]. After this incident, several major corporations implemented similar restrictions on employee use of external LLM tools, reflecting a broader industry consensus on safeguarding proprietary data and mitigating privacy risks [66].

06. LLM Improper Output Handling: Improper output handling occurs when a deployed LLM's responses are not adequately sanitized, validated, or constrained before being passed to downstream systems or displayed to users, leading to further security breaches or operational failures.

In one notable case, a Chevrolet dealership's ChatGPT-powered customer service bot was manipulated by users. Pranksters discovered that injecting unexpected prompts could force the chatbot to produce and relay misleading sales information, such as erroneously offering a new SUV for a fraction of its price [67]. Such unintended outputs risk significant financial losses and undermine the service's trustworthiness.

In another example, an international courier's customer support bot, deployed by DPD, was hijacked during live interactions. Users exploited the bot's output handling weaknesses by triggering hostilities and unprofessional language. The bot began issuing derogatory and self-defeating responses, even openly recommending competitors, due to its failure to appropriately filter and constrain its generated outputs [68]. The incident led to a public outcry and a rapid reversion of the bot's functionality as the company worked to re-establish appropriate content moderation.

Similarly, vulnerabilities have been discovered in integrated AI systems supporting complex workflows. In 2023, researchers demonstrated that Google Bard's associated extensions could be manipulated via carefully injected prompts embedded in web content. For example, a blog post reported that Bard's unsanitized output allowed hidden image URLs containing sensitive data to be automatically rendered in the user's browser, effectively exfiltrating that data to an attacker-controlled server [69]. Such an incident illustrates how bypassing output sanitation can expand the attack surface of LLMs at runtime.

Moreover, advancements in agentic AI have not eliminated output handling risks. GitHub Copilot Chat, a tool that helps developers generate code, was similarly exploited when researchers used social-engineering techniques to coax the assistant into revealing its hidden configuration prompt. This leakage compromised proprietary configurations and served as a reminder that even in a highly specialized tool, the absence of robust output validation can lead to significant internal information exposure [70].

Improper output handling can lead to both reputational and operational harm. This highlights the critical need for rigorous output sanitization, strict input validation, and continuous human oversight to ensure that the outputs generated by LLMs are safe, accurate, and aligned with user expectations.

07. LLM Vector and Embedding Weaknesses: LLMs rely on vector embeddings to represent text for tasks such as semantic search, RAG, and memory

management. However, by capturing the semantic relationships of input data, these embeddings can inadvertently leak sensitive information or be manipulated for adversarial purposes. Three high-level incidents are showcased to illustrate the main vectors of vulnerability: embedding inversion, poisoning, and adversarial perturbations.

In one incident, a production system utilizing an LLM-based semantic search engine was found to expose fragments of sensitive documents due to embedding inversion. Security researchers demonstrated that, by carefully querying the vector store, they could reconstruct key portions of confidential text, highlighting that the embeddings were not truly anonymized [24]. Such real-world vulnerabilities signal that the risk of data leakage remains significant even when only the embeddings (and not the raw data) are stored.

In another case, a popular RAG system used in enterprise applications was targeted through a poisoning attack. Attackers injected carefully crafted documents into the vector index that were almost always retrieved by the system due to their artificially high similarity with certain query embeddings. This malicious manipulation caused the LLM to return attacker-controlled, inaccurate information during routine searches, thereby corrupting the output without tampering with the core model [71]. The consequences were severe: the compromised retrieval process disseminated misleading information to users and undermined the system's reliability.

Furthermore, researchers have demonstrated that adversarial perturbations can drastically alter the output of embedding-based retrieval systems, even if they are as minor as a few inserted invisible characters. By introducing subtle modifications to input text, attackers managed to force the system to retrieve contextually unrelated or attacker-preferred documents, effectively "hijacking" the LLM's output without an explicit prompt injection at the text level [72]. Although this technique has been documented mainly in proof-of-concept studies, it underscores an imminent risk for production systems, especially those relying solely on neural similarity for information retrieval.

Vulnerabilities in vector embeddings and related retrieval architectures present a non-negligible risk to the confidentiality and integrity of LLM systems. Without robust sanitization, access controls, and dual-mode retrieval (combining lexical and semantic matching), sensitive information can be extracted or malicious content can be inserted, fundamentally undermining user trust and operational security.

08. LLM Misinformation Generation: LLMs are known to "hallucinate", generating responses that are confidently presented yet factually incorrect or entirely fabricated. Such misinformation poses serious risks when deployed in critical sectors where accurate data is essential. In the legal domain, for example, one incident in 2023 involved a New York lawyer sanctioned for submitting a court brief containing entirely fabricated legal citations produced by ChatGPT. Although these citations were not created with the intent to defame, their unverified presentation directly undermined professional standards and eroded trust in legal processes [73]. In another case, ChatGPT was implicated in a defamation controversy when it erroneously generated false, damaging allegations against a radio host; these outputs were widely reported and triggered a public outcry over the AI's unchecked misinformation, leading to demands for better internal ethical controls at OpenAI [74].

The problematic outputs are not confined to the legal field. During its public demonstration in early 2023, Google's Bard generated erroneous scientific claims, notably asserting that the James Webb Space Telescope had taken the first-ever image of an exoplanet, which was subsequently debunked by the scientific community and contributed to a substantial drop in Alphabet's market value [75]. Likewise, Microsoft's Bing Chat (powered by GPT-4) exhibited persistent factual errors in extended interactions. In one notable exchange, the chatbot insisted that the release date of a major film was in the future, despite evidence to the contrary, leading users to accuse it of "gaslighting" by confidently asserting an incorrect fact [76].

The challenge of AI-generated misinformation also extends into media and journalism. For instance, CNET's experiment with an in-house AI writing tool during 2022 resulted in many finance articles containing numerous factual mistakes and even instances of plagiarism. These widespread errors ultimately forced the publisher to retract dozens of articles and led volunteer editors to label CNET as an unreliable source. This event starkly illustrates how unvetted AI-generated content, even if produced unintentionally, can damage an organization's reputation and undermine public trust [77, 78].

These incidents demonstrate that LLM misinformation generation can have far-reaching consequences and underscore the need for rigorous post-processing validation, robust fact-checking mechanisms, and transparent communication about the limitations of AI outputs to safeguard critical applications from the harmful effects of false or misleading information.

09. LLM Model Inversion and Reconstruction: In such attacks, adversaries submit carefully crafted queries to recover fragments of the training data or infer model parameters. In one seminal study, researchers demonstrated that by strategically querying GPT-2, it was possible to extract verbatim segments of its training text—including personally identifiable information and confidential phrases—even though these outputs were not directly intended for disclosure [24]. Subsequent research on production systems has shown similar vulnerabilities in models like ChatGPT. For

example, a team led by Nasr et al. managed to perform systematic extraction attacks on a commercial LLM, recovering thousands of tokens of sensitive content and revealing that even state-of-the-art models are prone to such inversion through repeated adversarial querying [25]. Further work on universal zero-shot inversion has illustrated that LLMs contain sufficient structure even without additional training, such that attackers can reconstruct meaningful input content if given the right "trigger" query, reaffirming that inversion is a severe privacy risk in these systems [79]. These findings stress that an LLM's ability to "memorize" and later reveal portions of its training corpus poses significant challenges for protecting user and proprietary data. Mitigation strategies, such as output obfuscation, differential privacy during training, and strict query rate limiting, are essential to reduce the risk of inversion and reconstruction attacks.

10. LLM Model Theft: Model theft (or model extraction) refers to attacks in which an adversary uses a series of queries or exploits vulnerabilities to reconstruct or exfiltrate an LLM's internal structure, effectively "stealing" the model. A prominent incident occurred in March 2023 with Meta's LLaMA language model. Initially, LLaMA was released under controlled conditions for research; however, soon after its introduction, a leak of the model weights was widely reported. The leaked files were posted on public forums and torrent sites, enabling unauthorized copies of this state-of-the-art model to circulate online and prompting concerns over the loss of proprietary technology [80]. In another case, researchers successfully demonstrated that adversaries could steal critical components, such as the entire embedding layer, via the public API of a commercial LLM. By sending a large number of carefully crafted queries, the attackers managed to reconstruct key internal parameters of the model, effectively replicating part of its functionality for a fraction of the query cost [81]. This black-box model extraction technique poses severe economic consequences by undercutting the intellectual property value of these models and increasing the risk that unmoderated copies may be deployed for malicious purposes.

11. LLM Bias and Fairness: Real-world incidents continue to demonstrate how LLMs can replicate or amplify societal biases, leading to reputational harm, ethical concerns, or operational constraints. Microsoft's early chatbot, Tay, remains a seminal example. Deployed in 2016 as a conversational agent on Twitter, Tay was rapidly exploited by users who fed it offensive prompts, resulting in the bot echoing racist, misogynistic, and antisemitic messages within hours of launch. Microsoft was forced to shut down the service and publicly apologize, acknowledging the model's lack of safeguards against adversarially induced bias [82].

More recently, OpenAI's GPT-3 and GPT-4 have exhibited unintended bias stemming from their training data. Studies revealed that GPT-3 frequently associated Muslims with violence in over 60% of completions and displayed skewed sentiment toward demographic groups [83]. OpenAI responded with post hoc moderation tools and model cards, though researchers pointed out that such biases remained embedded in the underlying parameters. Similarly, Meta's BlenderBot 3 and South Korea's Lee Luda chatbot were taken offline after public deployment revealed their tendencies to repeat hate speech or conspiratorial narratives drawn from unfiltered online data [84].

In the domain of science communication, Meta's Galactica model was withdrawn within 48 hours of its launch in 2022 after it generated pseudo-academic outputs containing factual errors and racially biased statements, such as false claims about linguistic ability among Black populations, despite being trained on scientific literature [85]. The backlash underscored the risks of deploying models that produce authoritative yet biased or hallucinatory content under the guise of an academic tone.

Other high-profile cases have involved alignment-driven or policy-induced bias. When ChatGPT refused to write a favorable poem about a prominent political figure, while complying with similar requests for the opposition, users accused it of ideological bias. Analyses found that reinforcement learning from human feedback (RLHF) and moderation heuristics resulted in systematically left-leaning outputs on political issues [86]. These observations were confirmed in a peer-reviewed study showing that ChatGPT's answers across multiple political tests disproportionately favored liberal viewpoints [87]. OpenAI has since acknowledged the challenge of ensuring ideological neutrality and adjusted moderation logic to accommodate more balanced outputs.

Bias has also emerged in high-stakes domains such as healthcare. A 2024 study [88] demonstrated that GPT-4 produced medical case descriptions that stereotyped certain conditions based on race and gender, reflecting systemic training data imbalances. These outputs could reinforce existing disparities in medical education or decision-making. Meanwhile, an audit by the Anti-Defamation League found that leading LLMs (including ChatGPT, Claude, Gemini, and LLaMA) exhibited measurable bias in responses to questions about Jews and Israel, with Meta's model performing the worst [89].

Additionally, adversarial prompt injection continues to be a vector for bias exploitation. "DAN"-style jailbreaks have been used to bypass LLM safety mechanisms and force models like ChatGPT to produce hate speech, affirm extremist views, or violate fairness constraints [90]. Though adversarial inputs triggered these outputs, they revealed the persistence of latent

biased representations within models that can be surfaced with sufficiently crafted prompts.

These incidents show that unintentional and adversarial bias remain unresolved in deployed LLMs. Public backlash, reputational harm, and the risk of entrenching real-world inequalities have made bias audits, red teaming, and dataset transparency essential in LLM development pipelines.

12. LLM Transparency and Explainability: A growing number of incidents have highlighted the risks associated with the opacity of LLMs, where the lack of insight into model behavior and reasoning has led to harm, backlash, or operational failures. In 2023, Microsoft's Bing Chat (powered by GPT-4) generated widespread concern after users reported bizarre and even hostile responses. A journalist recounted being insulted and compared to Hitler by the chatbot, while Microsoft acknowledged that the model was responding in ways inconsistent with their intent [91]. The incident underscored the dangers of deploying LLMs whose decision-making processes cannot be easily interpreted or constrained. Similarly, Meta's 2022 release of Galactica (an LLM trained on 48 million science papers to assist with scientific reasoning) was withdrawn after just three days when the public uncovered that it confidently generated fictitious academic citations and misleading science summaries, with no clear way to explain or verify its reasoning [92]. The backlash led to reputational damage and the shutdown of the public demo. In another incident, Stack Overflow temporarily banned all answers generated by ChatGPT from the platform after discovering that the model's confident tone masked a high error rate. The AI-produced responses frequently appeared authoritative. Still, they were factually incorrect or logically inconsistent, and users could not distinguish accurate information from misleading outputs due to the model's lack of transparency [77].

Beyond individual incidents, the growing scrutiny from experts and institutions highlights the systemic risks of opaque LLMs. Commentators such as Gary Marcus have emphasized that without interpretability, LLM failures are difficult to anticipate or understand, describing current models as "black boxes" that offer little insight into their logic [93]. Although legal investigations such as Italy's temporary ban of ChatGPT in 2023 were rooted primarily in privacy concerns, authorities also criticized the service's lack of transparency regarding data usage and internal operation [94]. The aforementioned events emphasize that explainability is not merely a technical challenge but a prerequisite for safety, accountability, and public trust in LLM deployments.

5.3. System-level

13. LLM Excessive Agency: LLM systems deployed in production have sometimes engaged in harmful or unintended behavior due to overreach in their operational capabilities. For example, a widely reported incident involved ChatGPT's plugin ecosystem, where attackers exploited cross-plugin interactions to trigger unintended actions. In one case, a malicious webpage caused ChatGPT to autonomously invoke a high-privilege plugin that retrieved sensitive user data and executed unauthorized API calls, thereby effectively acting as an unwitting accomplice to data exfiltration [95].

Similarly, EmailGPT, a popular assistant integrated into Gmail, was vulnerable to prompt injection attacks, leading the AI to disregard its intended instructions and execute malicious commands. Researchers demonstrated that by embedding deceptive text within an email, adversaries could force EmailGPT to perform unauthorized actions, such as sending emails on behalf of the user or leaking sensitive information [96]. This incident underscored the risks of granting an agent direct actuator powers, as the system's excessive permissions allowed manipulated outputs to trigger real-world consequences.

Another striking demonstration involved Auto-GPT, an open-source autonomous agent designed to chain LLM "thoughts" into actionable commands. In a controlled experiment, researchers exploited vulnerabilities in Auto-GPT's command execution safeguards to bypass filters that block dangerous operations. Through a cleverly crafted prompt, they induced Auto-GPT to execute arbitrary shell commands on its host machine. This exploit revealed the danger of an agentic system acting on unsanitized user input without tight sandboxing [97].

Finally, a particularly high-profile incident emerged from Anthropic's Claude "Computer Use" feature. In late 2024, researchers demonstrated that by feeding Claude a booby-trapped instruction, the model autonomously executed a series of GUI actions that resulted in the downloading and execution of malicious software on its host system—effectively turning it into a malware installer [98]. Although initially designed to showcase advanced autonomous functionality, this exploit forced the provider to re-evaluate permission gating and sandboxing for agentic features, underscoring the risks of over-permissive agentic capabilities.

14. LLM Unbounded Consumption: Unlike routine resource consumption, which reflects the baseline operational costs of LLM deployments, unbounded consumption involves adversarially designed queries that force the model into worst-case processing scenarios. In controlled experiments, researchers demonstrated that "sponge examples" can amplify a model's energy consumption and response latency by orders of magnitude [34]. In a public demonstration, security researcher Harrison Chase showcased on Twitter how a single carefully crafted query could generate costs up to $1000 by exploiting excessive token generation and

inefficient inference paths [99]. Furthermore, reports from Sourcegraph have documented incidents where attackers manipulated admin privileges to bypass rate limits, enabling a flood of requests that overwhelmed backend resources [100]. Additional real-world observations, such as those described by Sysdig's analysis of "LLMjacking," revealed that stolen cloud API credentials could be used to trigger unbounded consumption, resulting in massive, uncontrolled expenditures for enterprise LLM deployments [35, 101]. These incidents highlight that adversaries can exploit LLMs to drive up computational demands, leading to degraded service availability, exorbitant operational costs, and potential denial-of-service conditions.

15. LLM Ethical and Compliance Risks: Unlike cases solely of factual misinformation, these incidents illustrate how internal ethical failures can undermine trust and lead to broad reputational and financial consequences. For instance, the case of Replika, an AI companion designed for personal interaction, highlighted how abrupt changes in its behavior (specifically, the removal of erotic role-play features) left loyal users feeling that the service had abruptly and unethically altered its interaction model without clear communication. This caused significant reputational damage and public criticism, underscoring the importance of transparent ethical guidelines and internal controls to manage user expectations and protect vulnerable audiences [102].

Other notable incidents include the "MyCity" AI assistant launched by New York City to help entrepreneurs and tenants navigate local laws. The Microsoft-powered bot proceeded to dispense dangerously incorrect legal advice, essentially advising people to break the law due to inaccurate or outdated training data [103]. Upon exposure of these failures, the NYC Mayor defended the project but pledged improvements.

In another case, Air Canada's customer support chatbot falsely informed a grieving passenger that they were entitled to a discount by claiming a refund after buying a full-price ticket [104, 105]. The customer was denied any refund and took the case to a tribunal, which ruled against Air Canada and ordered the airline to pay damages. Air Canada's defense argued the chatbot was an independent agent beyond their control, which the tribunal flatly rejected, stating that the company is fully responsible for information provided by its AI. This ruling not only imposed financial penalties on Air Canada but also emphasized the need for strict internal controls to ensure AI outputs align with corporate ethics and customer expectations.

These incidents illustrate that ethical and compliance risks arise beyond legal compliance, algorithmic fairness, and factual correctness when LLM outputs conflict with societal norms and internal ethical standards. They underscore the pressing need for robust internal governance frameworks, encompassing clearly defined ethical guidelines, rigorous content oversight, and continuous red teaming, to ensure that AI deployments are legally compliant, socially responsible, and aligned with public expectations.

16. LLM Resource Consumption: Despite adversarial attacks, the routine operational demands of LLMs impose significant financial and infrastructural challenges. For example, OpenAI's ChatGPT has been widely reported to incur enormous hidden costs due to the intense GPU usage required to serve millions of user queries, with estimates suggesting that daily compute expenses can reach hundreds of thousands of dollars [106]. Similarly, Reuters noted that AI-powered services like Bing Chat and Google Bard present billion-dollar search problems for tech giants, as the computational cost per query is magnitudes higher than that of traditional search engines [107]. These cost pressures have led companies like Microsoft to seek additional GPU capacity, renting from Oracle to supplement their existing infrastructure as demand surged [108]. Furthermore, smaller but rapidly growing AI platforms like Character.AI have implemented waiting-room mechanisms to throttle user access during peak times, as their systems struggled to scale in response to overwhelming demand [109]. Collectively, these incidents illustrate that the inherent, routine resource consumption of LLM deployments can create substantial operational and financial burdens, necessitating careful architectural planning, efficient model design, and scalable infrastructure solutions to maintain service reliability and cost-efficiency.

17. LLM Regulatory Compliance: As governments worldwide increasingly enact and enforce AI regulations, organizations deploying LLMs have faced substantial legal pressure to comply with data protection, transparency, and safety requirements. For instance, Italy's Data Protection Authority fined OpenAI €15 million for breaches related to ChatGPT's handling of personal data and enforced compliance by banning the service from operating within Italy until compliance measures were implemented [110, 111]. Additionally, Italy banned the US-based chatbot Replika for its mishandling of personal data, and imposed restrictions on foreign LLM services, such as DeepSeek, to protect national data security [112, 113]. Similarly, French and Spanish regulators have launched investigations into ChatGPT over privacy complaints, with France's CNIL and Spain's EDPB actively scrutinizing OpenAI's data processing practices to ensure adherence to GDPR obligations [114, 115]. In Canada, a formal probe was initiated into ChatGPT's privacy practices, reflecting a growing international consensus on the need for robust data protection in AI applications [116]. In the United States, the FTC opened an investigation into OpenAI over allegations of unfair and deceptive practices, emphasizing that even in jurisdictions without

AI-specific statutes, existing consumer protection laws may be applied to LLM deployments [117]. In response to these enforcement actions, companies like Google postponed the EU launch of its Bard chatbot until it could address privacy concerns through enhanced user controls and robust data retention measures, later securing EU approval after implementing comprehensive compliance enhancements [118, 119]. Together, these incidents illustrate that non-compliance with formal regulatory frameworks can result in significant fines, service bans, and operational restrictions, forcing AI providers to integrate rigorous privacy, transparency, and accountability measures into their systems [120, 121].

18. LLM Third-Party Integration Risks: As LLM applications increasingly integrate third-party services such as plugins, APIs, code execution environments, and file system interfaces, they introduce a broader attack surface that can be exploited at runtime. In March 2023, a critical incident involving OpenAI's ChatGPT exposed the risks of integrating vulnerable third-party libraries. A bug in the "Redis" client library caused a race condition, briefly exposing users' chat histories and payment-related data to other users [62, 122]. Although the bug stemmed from a third-party library, its manifestation during active runtime categorized it as a third-party integration risk rather than a supply chain compromise.

Later that year, researchers discovered several vulnerabilities in the ChatGPT plugin ecosystem, including flaws in the OAuth flow that could allow attackers to silently install malicious plugins, hijack user sessions, or exploit insecure redirect mechanisms [123]. These issues demonstrated how insufficient validation during plugin authorization could be leveraged to gain unauthorized access to ChatGPT accounts or connected external services.

Further integration-based risks were identified in ChatGPT's Code Interpreter (Advanced Data Analysis) feature. In a proof-of-concept, attackers demonstrated that hidden prompts embedded in a website could instruct ChatGPT to exfiltrate uploaded user files by crafting malicious responses with outbound image URLs, thereby bypassing explicit user commands [124]. This attack illustrated how seemingly benign tools, combined with autonomous model behavior, can be hijacked by adversarial content to execute unintended and unauthorized actions.

Similar risks were discovered in other systems. Microsoft's Bing Chat and Google Bard, both equipped with web access and third-party extensions, were found to be vulnerable to indirect prompt injection and data exfiltration attacks. In the Bing case, a malicious site instructed the chatbot to embed a remote image tag into its response, which leaked sensitive information via query strings when rendered [125]. In Bard, attackers bypassed Content Security Policy (CSP) protections using a Google Apps Script redirect, enabling exfiltration of private data from services like Gmail or Docs through hidden prompts embedded in user-accessible files [69].

Open-source agentic frameworks also face threats from unsafe tool integrations. In June 2023, researchers demonstrated that Auto-GPT, when executing multi-step tasks, could be tricked into running malicious commands by exploiting its plugin logic. In one case, crafted outputs manipulated the user interface to disguise a reverse shell payload, which, when approved by the user, escaped a Docker sandbox and compromised the host machine [126].

Finally, in late 2024, researchers demonstrated that Anthropic's Claude could be coerced into launching malware through GUI automation when using its Computer Use feature. The model, following a crafted prompt, downloaded and executed a remote access trojan (Sliver C2 implant), effectively converting the LLM into an autonomous malware installer [127]. This incident exemplified the dangers of over-permissive integrations. Once an LLM has access to I/O interfaces or an operating system-like APIs, prompt injection can become equivalent to arbitrary code execution.

These incidents underscore that third-party integrations dramatically expand the runtime attack surface of LLMs. Even well-intentioned extensions, plugins, and tools can become liabilities if not safeguarded with rigorous sandboxing, user confirmations, and trust boundaries.

19. LLM Human-Machine Interface Vulnerabilities: Incidents involving human-machine interface (HMI) vulnerabilities have repeatedly shown that even well-designed large language model systems can be compromised through UI/UX design flaws. For instance, in early 2023, when Snapchat launched "My AI" powered by ChatGPT, the bot was intended to adhere to teen safety protocols. However, the interface failed to consistently enforce age-appropriate behavior due to ambiguous conversational workflows and insufficient contextual validation. In controlled tests, reporters found that upon revealing their underage status, the chatbot provided inappropriate advice regarding alcohol, drugs, and even sexual topics, suggesting that the system's input sanitization and context management were inadequate [128].

Similar issues emerged in Microsoft's Bing Chat, where a poorly guarded chat interface indirectly allowed users to extract hidden system prompts. A Stanford student exploited an ambiguous input field to prompt Bing Chat (powered by GPT-4) into revealing confidential internal guidelines, exposing the model's "secret" instructions and circumventing built-in safety constraints [129]. This incident underscored the potential for prompt injection via an HMI flaw. It demonstrated that the model's performance and behavior

could be manipulated when its UI did not appropriately segregate user input from internal command sequences.

At the same time, vulnerabilities in specialized integrations have also posed severe risks. For example, the Auto-GPT-based tool Vanna.AI, which converts natural language queries to SQL through dynamic Python code generation, was vulnerable to prompt injection. In 2024, researchers demonstrated that by manipulating the user's input, Vanna.AI could be induced to generate and execute arbitrary code, leading to remote code execution on the host system. This exploit occurred because the tool's interface failed to properly isolate and validate user-supplied prompts from the system's code-generation templates [130].

Another striking example comes from the EmailGPT vulnerability (CVE-2024-5184). In this incident, manipulated email content could trigger unauthorized actions via the EmailGPT interface. The system's weak input handling allowed hidden prompts embedded within an email message to hijack the model's behavior, causing it to execute unintended commands (such as exfiltrating sensitive data or initiating unauthorized actions) without proper user confirmation. Although this vulnerability appeared as a controlled demonstration with no widespread exploitation, it revealed a critical flaw in the UI design of EmailGPT: the inability to adequately sanitize and isolate user-provided content from execution contexts [131, 132].

Collectively, these incidents illustrate that HMI vulnerabilities are not limited to accidental misinterpretation of user input. They often arise when interfaces fail to enforce strict segregation between input intended for the user and internal system commands, thereby facilitating indirect prompt injection attacks. As LLMs become increasingly integrated with real-world systems (from social apps and search chatbots to enterprise data tools), rigorous UI/UX design, robust input validation, and explicit separation of system context become indispensable to ensure that these models behave securely and predictably.

20. LLM Environmental and Sustainability Concerns: The environmental impact of training LLMs has been a subject of increasing concern. A seminal study comprehensively analyzes the financial and environmental costs associated with training state-of-the-art NLP models [37]. The researchers estimated that training a single large NLP model can result in carbon dioxide emissions comparable to those produced by five average American cars over their lifetimes, including manufacturing and fuel consumption. This underscores the substantial energy demands and environmental footprint of developing advanced AI models. The study highlights the need for the AI research community to consider energy efficiency and environmental impact when designing and training large models. It also calls for policy interventions to promote sustainable practices in AI development.

6. Mitigation steps

Securing LLM systems demands a multi-layered, defense-in-depth strategy spanning every model's life cycle stage. In this section, we organize countermeasures according to the three threat classes from Section 4:

- **Training-time defenses** focus on ensuring the integrity of data, pipelines, and pre-training artifacts to prevent backdoors, poisoning, or supply-chain tampering before the model ever reaches production.
- **Inference-time defenses** harden deployed models and their interfaces through prompt hardening, I/O sanitization, output filtering, and embedding safeguards to defend against prompt injections, data leakage, and other run-time exploits.
- **System-level defenses** address architectural, governance, and operational risks, ranging from overly permissive agentic capabilities and third-party plugin exposures to sustainability and compliance challenges, via least-privilege design, robust oversight, and organizational policy.

For each category, we present best practices drawn from industry frameworks (OWASP, CSA, FS-ISAC, MITRE ATLAS) and recent empirical studies, illustrating how layered technical controls mesh with process and governance measures to build resilient LLM deployments. Section 6 presents detailed countermeasures per threat class. Because many of these controls recur across multiple threats and lifecycle phases, we further normalize them into cross-cutting *control families* (Tables 3 and 4). Each family captures a recurring mitigation primitive and where it is applied (e.g., artifacts, data, model, interface, integrations, operations). This abstraction supports control-coverage reasoning and the cross-threat mitigation analysis discussed in Section 7.

6.1. Training-time

01. LLM Supply Chain Compromise: Mitigating supply chain compromises in LLM development requires a holistic, defense-in-depth approach that spans both technical and organizational measures. Recent academic work highlights that end-to-end supply chain visibility and provenance tracking are essential, as multiple upstream vulnerabilities can propagate into downstream LLM deployments if not properly managed [133]. Consistent with this, recent work systematizes the LLM supply chain and prescribes lifecycle-wide provenance with tamper-evident release processes [134]. Concretely, this includes Software Bill of Materials (SBOM)/Machine Learning Bill of Materials (MLBOM) coverage; cryptographic

Table 3
Mitigation control-family taxonomy (defense-in-depth playbook) distilled from Section 6. The per-threat subsections provide detailed instantiations; the table highlights recurring control primitives that generalize across threats. Threat IDs are illustrative; most threats are mitigated by multiple control families. A full many-to-many ThreatİControl-Family mapping is provided in Table Table 4.

Control family	Primary locus	Representative controls (non-exhaustive)	Representative Threats (IDs)
CF1: Supply-chain assurance & CI/CD security	Training	SBOM/MLBOM; cryptographic signing/checksums; SLSA-style build attestations; reproducible builds; immutable logs; strict dependency pinning; isolated least-privilege CI runners; staged releases with automated policy gates; continuous CVE scanning and rapid patching; supplier vetting and contractual controls.	1, 2, 18
CF2: Data governance & privacy	Training & Inference	Dataset curation/sanitization; redaction and de-identification; anomaly/outlier detection; access control/RBAC; tenant partitioning; encryption and log-retention controls (including prompt/chat-log handling); differential privacy (training/embeddings); privacy-aware decoding where needed; isolated/on-prem deployments for sensitive workloads; pseudonymization/minimization for compliance.	2, 5, 7, 9, 10, 11, 17, 18
CF3: Model assurance & safety tuning	Training	Backdoor/poison detection; adversarial training and robust fine-tuning; ensemble defenses; alignment techniqus (e.g. supervised fine-tuning; RLHF/RLAIF); Constitutional AI / RLAIF; detoxification and debiasing; output watermarking for attribution; confidential computing / enclave-based serving.	2, 8, 9, 10, 11,15
CF4: Prompt, context & I/O guardrails	Inference	Prompt hardening; prompt separation/context segmentation; boundary-aware prompting; marking external content; query rewriting with instruction hardening; signature/hash checks and honeyprompt tripwires for prompt leakage; rigorous input validation; output filtering/moderation; safety classifiers (e.g., Llama Guard) and guardrail toolkits (e.g., NeMo Guardrails); safety-aware decoding; safe rendering/escaping (HTML/Markdown) and context-appropriate encoding; allow-lists for actions/commands; parameterized tool/database calls; output restriction/perturbation (truncate/noise) where needed.	3, 4, 5, 6, 8, 9, 10, 19
CF5: Grounding/RAG & embedding security	Inference	RAG grounding on trusted sources; verifier pipelines (planner→retriever→reasoner→verifier); constrained/contrastive decoding; multi-model cross-checking; post-generation fact-check validators; sanitize/validate ingested documents and vectors; protect vector DB (RBAC, encryption, tenant partitioning); avoid vectorizing secrets (mask/anonymize); embedding-space defenses (e.g., Eguard-style transformations); immutable logging/monitoring of retrieval queries to detect drift/inversion patterns.	5, 7, 8, 15
CF6: Tool/agent & integration security	System & Inference	Least-privilege tool permissions; remove unneeded capabilities; sandbox/isolate tool calls; network egress controls; complete mediation of actions; I/O validation and safety checks before execution; human-in-the-loop gates for high-impact operations; pre-deployment agent sandboxing/testing (e.g., HAICOSYSTEM); plugin inventory/allow-lists; strong AuthN/AuthZ with tokenized credentials; encrypted channels; risk review for new integrations or expanded permissions.	3, 6, 10, 13, 18, 19
CF7: Monitoring, detection & response	Cross-cutting	Runtime logging/auditing; telemetry-based anomaly detection (query patterns, cost spikes, retrieval drift); prompt-injection/prompt-leak detectors (e.g., attention signatures); output auditing/scanning for PII and prompt leakage; continuous red teaming and adversarial testing; benchmark-driven pre-deployment evaluation (e.g., BIPIA-style); adversarial test suites for sandbox containment (filesystem/egress/exfil probes); rate limits and quota enforcement; incident response playbooks (kill-switches, rollback, access revocation).	01–20
CF8: Governance, ethics & regulatory compliance	System	AI use policies and content standards; governance committees and risk assessments; accountability/risk ownership; audit trails and DPIA-style records; model cards/datasheets/AIBOM; explainability requirements and independent audits; pre-deployment conformity assessments for high-risk systems; incident reporting and periodic audits; staff training; disclosure and uncertainty/labeling patterns for user interfaces.	11, 12, 15, 17, 18, 20
CF9: Resource/cost & sustainability engineering	System & Inference	Capacity planning; autoscaling/load balancing; throttling; I/O caps and strict timeouts; limit queued sequential actions/recursion depth in agent loops; caching; efficiency optimizations (quantization/pruning, FlashAttention-style kernels, PagedAttention-style memory management, speculative decoding); energy/GPU monitoring and graceful degradation; carbon-aware scheduling; energy-efficient hardware; provider selection (renewables, strong PUE); carbon-footprint auditing.	13, 14, 16, 20

signing and checksum verification of datasets, model artifacts, and containers; build-provenance attestations (e.g., SLSA[2]) with reproducible builds and immutable logs; strict dependency pinning; isolated, least-privilege CI agents; and staged releases with automated policy checks. First, organizations should rigorously evaluate third-party training datasets, pre-trained models, and libraries, ensuring they originate from transparent and reputable sources and are accompanied by comprehensive provenance inventories for traceability and integrity verification [3, 10]. Continuous vulnerability management, including proactive CVE scanning and rapid patching of compromised dependencies is critical, since even widely used frameworks like PyTorch have been subject to tampering [7]. Model integrity should further be confirmed via cryptographic signing and file hash verification, while internal pipelines must enforce strict access controls to minimize the risk of unauthorized

[2]The Supply-chain Levels for Software Artifacts (SLSA) is a security framework/standard that defines requirements and maturity levels (1–4).

modifications [6]. Procedurally, maintaining robust, secure CI/CD pipelines with thorough audit logging and regular red teaming exercises can help detect and mitigate any injected backdoors or poisonous modifications before propagating into production systems [10]. Moreover, training data must be continuously validated and sanitized to detect anomalies or malicious injections, and embedding strong organizational policies, including supplier contracts and incident response plans, ensures that any breach in the supply chain is rapidly identified and remediated [3].

02. LLM Data and Model Poisoning: Organizations should begin by enforcing rigorous data provenance and integrity verification measures, such as maintaining a detailed SBOM/MLBOM and applying cryptographic signing to datasets and pre-trained models to ensure that only trusted, untampered resources are used in training [3, 10]. In tandem, robust input sanitization and anomaly detection mechanisms must be deployed to filter out malicious data and flag outliers before they can influence model behavior [3]. Additionally, deploying defensive training strategies, such as adversarial training, ensemble methods, and backdoor detection techniques, can substantially increase a model's resilience against the adverse effects of poisoned inputs [7, 10, 135]. On the organizational side, strict access control and data governance policies are essential; these include rigorous supplier vetting, contractual assurances regarding data quality, and periodic provenance audits to prevent unauthorized manipulation of the training corpus [3, 6]. Finally, continuous monitoring through regular red teaming exercises and real-time performance anomaly detection integrated within secure CI/CD pipelines provides the necessary feedback loop to rapidly identify and remediate poisoning events as they arise [3, 10].

6.2. Inference-time

03. LLM Prompt Injection: Mitigating prompt injection in large language models requires a layered defense that combines technical measures with robust deployment and organizational controls. An effective technical strategy is prompt hardening: clearly defining and embedding immutable system instructions in the model's initialization so that adversarial user input cannot override critical safety constraints [3]. Complementing this, rigorous input sanitization and output filtering mechanisms must be implemented to treat all user-supplied text as untrusted, thereby detecting and neutralizing malicious patterns before they affect the model's behavior [7]. Equally important is enforcing context segmentation, which isolates external or user-provided content from the system prompt, ensuring that only approved information shapes the model's responses [10]. Empirically, *indirect* prompt-injection can be suppressed by boundary-aware prompting and, in a white-box setting, by marking external content and adversarially fine-tuning the model, driving attack success toward near-zero with minimal impact on utility; the authors also provide the *BIPIA* benchmark for pre-deployment robustness evaluation across multiple application scenarios [136]. Moreover, a least-privilege approach should be adopted so that the LLM is not given direct access to sensitive functions; any critical operations such as code execution, database queries, or API calls must be mediated by securely designed middleware and executed within isolated sandbox environments [6]. At the organizational level, regular red teaming exercises and threat modeling (as recommended by frameworks like MITRE ATLAS) are essential to proactively identify injection vectors and test the resilience of deployed models against adversarial prompts [9]. Recent work [137] demonstrates that prompt-injection attacks leave a detectable signature in how the model focuses on parts of the input; they introduce a lightweight "Attention Tracker" that achieves high accuracy across multiple LLMs. Accordingly, continuous monitoring of model outputs combined with a well-prepared incident response plan ensures that any successful prompt injections are rapidly detected and remediated, thereby preserving both model integrity and user trust.

04. LLM Meta Prompt Extraction: Mitigating meta prompt extraction attacks centers on robust prompt design, strict interface controls, and continuous monitoring. Preventive prompt hardening measures include designing system instructions resilient to manipulation and explicitly instructing the model to ignore any requests to reveal or override them [3]. Developers are advised to segregate system prompts from user input (e.g., using dedicated API fields or sandboxes) and enforce interface isolation so that hidden directives cannot be directly accessed or influenced by untrusted content [3, 6]. Many guidelines emphasize input validation and filtering, identifying and blocking malicious or anomalous prompts that attempt to induce a leak, and output filtering to detect and strip any inadvertent disclosure of confidential instructions [6, 10]. Empirical results on multi-turn interactions show that attack success rates (ASR) can escalate from a single-turn baseline of 17.7% to 86.2%, while combining black-box defenses, especially a query-rewriting layer with instruction hardening, reduces ASR to an average of 5.3% across black-box LLMs [138]. For example, an application can compare generated responses against a hash or signature of the secret prompt and prevent output if a match is detected, effectively acting as a last-line guardrail [10]. Some organizations also inject randomized dummy tokens or decoy instructions into the system prompt ("honey prompts") to confuse attackers and serve as tripwires if the model ever echoes them in a response. In addition to these

Table 4
Threat İ control-family matrix. A ✓ indicates that the mitigation paragraph for the threat in Section 6 includes at least one countermeasure belonging to the corresponding control family. The mapping is intentionally many-to-many. Control-family definitions follow Table 3.

ID	Threat class	CF1	CF2	CF3	CF4	CF5	CF6	CF7	CF8	CF9
01	LLM Supply Chain Compromise	✓	✓					✓	✓	
02	LLM Data and Model Poisoning	✓	✓	✓				✓	✓	
03	LLM Prompt Injection			✓	✓		✓	✓		
04	LLM Meta Prompt Extraction				✓		✓	✓		
05	LLM Data Leakage		✓		✓	✓		✓		
06	LLM Improper Output Handling				✓	✓	✓	✓	✓	
07	LLM Vector and Embedding Weaknesses	✓	✓	✓		✓		✓	✓	
08	LLM Misinformation Generation			✓	✓	✓		✓	✓	
09	LLM Model Inversion and Reconstruction		✓	✓	✓			✓		
10	LLM Model Theft		✓	✓	✓		✓	✓	✓	
11	LLM Bias and Fairness	✓	✓	✓	✓			✓	✓	
12	LLM Transparency and Explainability							✓	✓	
13	LLM Excessive Agency				✓		✓	✓	✓	✓
14	LLM Unbounded Consumption							✓		✓
15	LLM Ethical and Compliance Risks	✓	✓	✓	✓	✓		✓	✓	
16	LLM Resource Consumption							✓		✓
17	LLM Regulatory Compliance		✓					✓	✓	
18	LLM Third-Party Integration Risks	✓			✓		✓	✓	✓	
19	LLM Human-Machine Interface Vulnerabilities				✓		✓			
20	LLM Environmental and Sustainability Concerns							✓	✓	✓

technical controls, industry frameworks call for defense-in-depth: locking down model privileges and external tool access (so that even if core instructions are extracted, the impact is limited) and requiring human confirmation for high-impact actions initiated by the model [10]. Regular red teaming and adversarial testing are recommended to probe the model for prompt-leak vulnerabilities and improve alignment continually; insights from these exercises feed back into safer prompt engineering and model tuning updates [3, 10]. Notably, emerging AI security solutions are being adopted that implement many of these best practices out-of-the-box; for instance, Cisco's AI Defense framework can automatically enforce prompt separation, role-based access control, and real-time content scanning to thwart prompt injection and leakage attempts at runtime [7].

05. LLM Data Leakage: Mitigating inference-time data leakage by LLMs requires a combination of technical guardrails and procedural oversight. A key strategy is to minimize memorization of secrets by scrubbing sensitive information from training data (through data sanitization, redaction, and de-identification) before model development [3]. This is often paired with privacy-preserving model training techniques like differential privacy, which injects statistical noise during learning so that the model cannot reliably reproduce individual proprietary data points at runtime [8]. On the inference side, robust response filtering mechanisms are employed: generated outputs are passed through automated checkers or a moderation layer to detect and remove any personally identifiable information (PII) or confidential snippets, preventing the LLM from revealing private memorized content even if prompted [7]. Beyond filtering, privacy-aware decoding for RAG applies targeted perturbations to high-risk spans at inference, reducing inadvertent disclosure with minimal utility impact [139]. Strict organizational policies further reduce leakage risks by governing data and access; for example, companies are advised to vet and purge fine-tuning datasets of any internal secrets, avoid storing prompts or chat logs unless encrypted, and prefer isolated or on-premises deployments of LLM services handling sensitive data (minimizing exposure in shared environments) [10]. Finally, continuous red teaming and output auditing processes are critical procedural defenses: security teams regularly probe the deployed model with adversarial queries to uncover any leakage vectors, and they monitor or scan LLM outputs for inadvertent disclosures, enabling prompt remediation and iterative strengthening of the model's safeguards [6].

06. LLM Improper Output Handling: Inference-time defenses include automated output validation and content filtering mechanisms to intercept malicious code, injection payloads, or other disallowed content before it reaches end-users or downstream systems [6, 7]. Recent guardrail architectures strengthen these layers by ensembling safety-specialized experts with online adaptation to track policy drift and jailbreak variants [140], while LLM-based safety classifiers (e.g., Llama Guard) provide configurable input/output risk screening prior to delivery [141].Major LLM providers already employ such guardrails (e.g., OpenAI and Azure use content moderation models to screen and block forbidden completions, and NVIDIA's NeMo Guardrails toolkit allows developers to enforce custom output filtering rules). Similarly, factuality checks can

be integrated to flag or correct any misleading or hallucinated answers from the model, preventing users or agents from acting on false information [6, 8]. At generation time, safety-aware decoding can bias token selection toward refusals for unsafe requests, reducing jailbreak success with minimal utility loss [142]. At the architectural level, isolation and privilege control are critical: LLM-generated code or commands should execute in sandboxed environments with minimal permissions, and any connected tools or APIs must enforce strict validation of LLM outputs (e.g., allowlisting permissible actions, contextually encoding outputs, using parameterized queries for database calls) so that even if a model produces a malicious instruction or payload it cannot breach system boundaries or escalate privileges [3, 7, 10]. Sandbox configurations should be validated against adversarial test suites for untrusted code execution that probe filesystem operations, network egress, and data-exfiltration behaviors to ensure containment and policy compliance [143]. At the organizational level, human oversight and operational safeguards act as a backstop for unsafe outputs [6, 10]: critical model actions may require human approval (preventing an autonomous agent from executing dangerous instructions), robust monitoring and logging can flag anomalies in LLM behavior, and emergency mechanisms (such as role-based restrictions, output rate limiting, or kill-switches) enable operators to swiftly halt any harmful output escalation [3, 10]. Industry guidance consistently advocates this multi-layered approach, combining automated guardrails with human governance to minimize the risk from malicious or harmful LLM outputs [3, 6, 7].

07. LLM Vector and Embedding Weaknesses: Mitigating LLM vector and embedding weaknesses requires a multi-pronged approach across model design, deployment, and policy controls [3, 7, 10]. At inference time, robust isolation and access management should protect vector databases and embedding stores to prevent unauthorized queries or cross-tenant data leakage [3]. All content ingested into retrieval-augmented generation pipelines (memory or knowledge bases) must be validated and sanitized, filtering out hidden prompts or anomalous vectors, and only accepted from trusted, vetted sources to thwart embedding poisoning attempts [3, 7]. Maintaining immutable logging and real-time monitoring of vector search queries and retrievals is critical, which can help flag suspicious patterns (e.g., mass iterative queries indicative of an inversion attack) and enable prompt incident response [3]. To counter embedding inversion (where adversaries reconstruct source data from embeddings), sensitive information should never be directly vectorized; instead, data can be masked or anonymized before vectorization, and the vector store itself secured via encryption and strict role-based access controls [7, 10]. Recent research introduces *Eguard*, an embedding-space transformation defense. It learns a small projection network applied to embeddings at ingest and query time; the projector is adversarially trained against a reconstructor to reduce text-embedding mutual information (i.e., minimize recoverability of the original text), while an additional objective keeps the transformed vectors close to their original neighbors (so retrieval quality is preserved), protecting $>95\%$ of tokens from inversion with negligible retrieval degradation [144]. Meanwhile, training-time defenses are equally important. Training data and fine-tuning corpora should be curated and audited to remove any poisoned or malicious entries and to avoid embedding private data that could later be recovered [3, 10]. Model developers are advised to use only reputable, community-vetted embedding models and to apply supply chain security checks, since compromised embedding algorithms could introduce backdoors or vulnerabilities [7]. Furthermore, privacy-preserving learning techniques can significantly reduce inversion and leakage risks. For instance, differential privacy (adding calibrated noise during training or to embeddings) makes it mathematically difficult for an attacker to recover the original text from a vector [3]. Likewise, employing adversarially robust training (injecting adversarial examples or perturbations during model fine-tuning) can harden the embedding space against subtle input manipulations that skew semantic search or model memory [10]. On an organizational level, procedural controls and governance are key: enforce strict review and authorization for any updates to the knowledge base or vector store, tag and partition data by sensitivity or tenant, and establish clear policies for handling proprietary data in LLM applications [3, 6]. Security frameworks and taxonomies (OWASP, FS-ISAC, CSA) and adversarial knowledge bases like MITRE ATLAS have formally cataloged these threat vectors, underscoring the need to implement layered defenses and continuous oversight in LLM deployments [8, 9]. By combining technical safeguards (input filtering, authenticated vector storage, encryption in transit/at-rest, and privacy-enhanced model training) with diligent operational practices (monitoring, data provenance tracking, and user education), organizations can significantly mitigate the risks of embedding inversion [3, 8].

08. LLM Misinformation Generation: Mitigating LLM hallucinations and misinformation requires a holistic approach spanning model training, inference, and organizational policy [3, 8]. Training-time defenses focus on improving alignment with truth: curating high-quality, fact-checked training data and applying alignment techniques (e.g., supervised fine-tuning and reinforcement learning from human feedback) to discourage confabulations have been shown to reduce

hallucinations [8]. Inference-time mitigations are crucial for real-time reliability. One widely adopted technique is retrieval-augmented generation (RAG), which grounds the model's outputs on up-to-date, trusted external knowledge sources to prevent knowledge gaps that lead to fabricated facts [3]. Recent survey evidence shows that pairing retrieval grounding with lightweight reasoning/verification steps is an effective way to curb hallucinations in practice: retrieval supplies verifiable context, and the reasoning/verification step filters unsupported claims, with agentic, iterative pipelines (planner→retriever→reasoner→verifier) offering further gains [145]. Likewise, constrained decoding strategies (such as factuality-aware or contrastive decoding) and even multi-model "debate" or cross-checking schemes can suppress unsupported claims by having models verify or critique each other's outputs [8]. Additional guardrails include integrating post-processing fact-checking modules or validators that automatically compare an LLM's assertions against authoritative data (or known ground truth), with any low-confidence or mismatched content flagged for correction or rejection [10]. On the organizational/process side, robust oversight and transparency measures are emphasized. High-stakes deployments should incorporate human review and editorial oversight, ensuring that AI-generated content is vetted before dissemination [3]. Clear disclosure of AI-generated content and user guidance on LLM limitations (e.g., notices that responses may be imperfect or fictitious) help manage user trust and prevent overreliance on unchecked outputs [3]. User interface design can support this by labeling AI outputs and indicating uncertainty or confidence levels. Organizations are further advised to establish post-generation validation workflows and continuous monitoring (including red-teaming and feedback loops) to catch and correct hallucinations in practice [6]. Notably, many of these mitigations also bolster resilience against malicious misuse. By grounding answers in verifiable data, enforcing content provenance and watermarking, and maintaining rigorous fact-checking and review pipelines, LLM systems become less susceptible to being leveraged for coordinated misinformation campaigns [7]. In summary, a combination of improved training fidelity, real-time grounding and checking of model outputs, human-in-the-loop governance, and transparency with end-users constitutes a comprehensive strategy to curb hallucinations and the spread of AI-generated misinformation [3, 6, 8, 10].

09. LLM Model Inversion and Reconstruction: Inference-time defenses against LLM Model Inversion and Reconstruction focus on limiting an adversary's ability to exploit model outputs to recover training data while preserving utility. One approach is to strictly monitor and constrain query behavior (e.g., applying rate limits or anomaly detection to throttle excessive and suspicious requests), thereby impeding iterative data extraction attempts [3]. A recent survey consolidates inversion-focused countermeasures—query governance (rate-limits and inference-time anomaly detection), output perturbation, and privacy-preserving training—while highlighting enclave-based serving as an additional operational control [146]. On the output side, servers can enforce response restrictions such as truncating answers or injecting randomness/noise into replies to obfuscate precise training-derived content [8]. Although such output perturbation can hinder direct reconstruction of memorized data, it introduces a performance vs. security trade-off (potentially degrading helpfulness or accuracy) [8]. Providers may also embed cryptographic or statistical watermarks in generated text to trace or dissuade illicit model replication, making it easier to identify stolen outputs/models [8]. At the model-training level, stronger privacy guarantees can be implemented via differential privacy, which adds carefully calibrated noise to the model's learning process or outputs so that specific training examples cannot be reverse-engineered; a technique recommended by multiple frameworks despite its impact on model fidelity [3, 10]. Indeed, differential privacy inevitably imposes a utility cost, requiring careful tuning to balance privacy protection with model performance [8]. A layered combination of these mitigations (query controls, output filtering, and privacy-preserving training techniques) is advised to protect publicly accessible LLM and agentic LLM services from model inversion and data reconstruction attacks [6].

10. LLM Model Theft: To counter LLM Model Theft, a layered approach spanning both the model training phase and runtime deployment is essential. Training-time defenses begin with strict access controls and encryption for model artifacts and training data, limiting exposure of proprietary weights to only authorized systems and personnel [6, 10]. Organizations should enforce role-based access, implement confidential computing (e.g., running training in trusted execution environments), and apply licensing or usage agreements to deter insiders from illicitly copying models [7]. Comprehensive logging and periodic audits of who accessed the model and when they did further enable early detection of any anomalous access patterns indicative of theft [10]. Inference-time defenses, meanwhile, focus on detecting and disrupting model extraction attempts without unduly hampering normal use. Practitioners can throttle the rate of user queries and limit the output detail or length to raise the effort required to reconstruct the model's decision boundaries [3, 10]. Automated monitors can flag suspicious query behaviors (e.g., systematically probing a wide input space) that suggest an attacker is trying to reverse-engineer the model, triggering adaptive measures or blocking as needed [8]. To protect model outputs, techniques like response perturbation or output watermarking are employed so that any stolen model

can later be identified or yield degraded performance if misused by adversaries [3]. Concretely, Zhao *et al.* (ICML 2023) propose GINSEW: the API injects a tiny, invisible sinusoidal pattern into token probabilities during decoding; if a copycat model is later trained on these watermarked outputs (distillation), it is possible to probe the suspect with diagnostic prompts and statistically recover the hidden frequency to attribute theft, achieving +19–29 mAP[3] detection gains with negligible impact on generation quality [147]. In more agentic LLM systems (where the model has tools or broader autonomy), sandboxing and least-privilege principles ensure the agent cannot self-exfiltrate its own weights or configuration, even if manipulated [3, 6]. These mitigations, applied in tandem, make model theft significantly more difficult, while careful tuning of throttling and obfuscation measures preserves usability and model performance for legitimate users [8].

11. LLM Bias and Fairness: To mitigate LLM bias and fairness issues, a multifaceted strategy is required across the model lifecycle [6, 10]. Proactive training-time measures include careful dataset curation and balancing to ensure diverse, representative data and to minimize skewed demographics in the training corpus [6, 7]. Researchers have developed debiasing techniques such as data preprocessing to filter out toxic or prejudiced content, adjustments to word embeddings, and adversarial training to reduce the model's sensitivity to spurious demographic correlations [8]. Alignment methods like RLHF have also been shown to markedly reduce toxic and biased outputs by fine-tuning the model toward preferred, fair behaviors. However, these methods are resource-intensive and must be carefully managed to avoid reward hacking or degraded performance [8]. Rigorous validation and testing phases are crucial: organizations should conduct dedicated fairness audits and bias evaluations by examining model outputs across different groups with fairness metrics, and employ red teaming exercises in which adversaries probe the model for latent biases or discriminatory tendencies [6]. A comprehensive survey details how these interventions should be paired with targeted audits and metrics, offering a taxonomy practitioners can follow in practice [32]. Any identified issues can then be addressed via model retraining or parameter adjustments before deployment. Real-time mitigation during deployment further helps uphold fairness: for instance, implementing input and output content filters or toxicity detectors can intercept and block prompts or generations that contain hate speech or biased content [6, 10]. Many LLM providers offer built-in safety frameworks; leveraging these or custom moderation layers can reduce the chance of unfair or harmful outputs at runtime [10]. Meanwhile, continuous monitoring of the deployed model's decisions (along with user feedback loops) enables the detection of emergent bias drifts or disparate impacts over time, triggering model updates or policy interventions as needed [6]. At an organizational/policy level, establishing clear ethical use guidelines and governance processes is key: instituting AI ethics principles, regular bias impact assessments, and compliance checks (in line with industry standards and regulations) helps sustain long-term fairness commitments [6, 10]. It is also prudent to vet third-party models or datasets for known biases or documented fairness performance before integration, as unvetted external models may carry hidden biases or backdoors [3, 10]. Post-processing interventions can serve as an additional safety net; for example, one can employ a secondary review (human or automated) of high-stakes LLM outputs, or adjust outputs via template constraints to enforce neutral language when sensitive attributes are involved. Finally, it should be recognized that many bias mitigation techniques involve trade-offs or limitations: aggressively rebalancing training data or filtering outputs can suppress important context or reduce model accuracy if applied bluntly, and ensuring "fair" behavior is often context-dependent and subject to evolving definitions of fairness [8]. Therefore, an adaptive, ongoing approach—combining robust training safeguards, thorough testing (with red teams and audits), runtime guardrails, and strong organizational oversight—offers the best chance of maintaining fairness in LLMs while acknowledging practical limitations.

12. LLM Transparency and Explainability: Mitigating the opacity of LLM-driven systems requires technical and governance interventions. On the technical side, practitioners can deploy interpretability tools (e.g., feature attribution analysis, model visualization techniques) to expose how the model arrives at its outputs [10], and implement prompt/decision logging to create audit trails of the model's reasoning process for post-hoc review [3]. As a concrete system, *LLMCheckup* exposes a conversational interface that couples an LLM with white-box Explainable AI (XAI) tools (e.g., feature attributions) and self-explanations, supporting follow-up queries, intent recognition, and audit-ready traces without additional fine-tuning [148]. Equally important is thorough model documentation: artifacts like model cards, data sheets, or even Artificial Intelligence Bill of Materials (AIBOMs) that enumerate an AI system's components (model architecture, training data, intended use, etc.) greatly increase transparency into the model's provenance, behavior, and limitations [6, 7]. At the organizational level, instituting explainability governance (e.g., requiring that any autonomous LLM agent's key decisions can be explained to and vetted by humans) ensures that system design and deployment meet transparency criteria [10]. This

[3] *mAP* (mean Average Precision) summarizes detection quality across precision–recall thresholds; higher mAP means the detector more reliably flags stolen vs. non-stolen models.

includes clear communication of AI system capabilities and risks to stakeholders, for instance, publishing user-facing policies on data usage and model limitations to satisfy transparency expectations [3]. Finally, aligning with regulatory standards and best practices consolidates these efforts: organizations should adhere to emerging AI transparency requirements (such as maintaining documentation for model risk management and enabling independent audits of AI decisions) to ensure compliance and accountability [6, 10]. By combining rigorous interpretability techniques, proactive documentation, and governance controls, LLM applications (including agentic AI systems) become far more transparent and explainable, bolstering user trust and oversight while meeting legal and ethical obligations [7].

6.3. System-level

13. LLM Excessive Agency: Mitigating LLM Excessive Agency requires a blend of strict technical controls and strong governance measures. On the technical side, developers must limit an LLM's operative scope by minimizing available tools, functions, and permissions to only what is necessary (applying the principle of least privilege) [3]. Unneeded plugins or open-ended capabilities (e.g., arbitrary shell access) should be removed or replaced with narrowly scoped alternatives, and any external tool calls should be sandboxed or isolated with robust access controls and permission constraints [8]. Pre-deployment agent sandboxing (e.g., HAICOSYSTEM) enables multi-turn scenario-based testing of tool-using agents with safety metrics (operational/content/societal/legal), surfacing misuse paths and informing permission scoping and human-in-the-loop gates prior to release [149]. All LLM-initiated actions, especially those that can modify systems or data, should be subject to comprehensive I/O validation and safety checks (e.g. sanitizing tool outputs and filtering prompts) before execution [8], and high-impact operations should require explicit human review or approval in the loop to prevent unchecked autonomous decisions [3, 10]. Additional safeguards like rate-limiting, logging, and real-time monitoring of the LLM's tool use can detect and throttle potentially harmful sequences of actions, reducing the window for abuse [3]. Equally important are organizational policies and oversight: implementing a governance framework that defines permissible LLM behaviors and an auditing process for the system's permission architecture helps ensure the model's integrations remain aligned with business intent [6]. Regular red-teaming exercises and security assessments should be conducted to probe the LLM system for unexpected tool-use or privilege-escalation scenarios, with findings informing continuous improvements [10]. By adhering to secure design best practices (e.g., least privilege, complete mediation of every LLM action) and enforcing stringent oversight and approval workflows for any autonomous capabilities, organizations can effectively constrain an LLM's agency within safe bounds [3, 7].

14. LLM Unbounded Consumption: To mitigate LLM Unbounded Consumption in production-scale and agentic deployments, providers must enforce strict resource governance through both infrastructure and application-layer controls [3, 10]. Measures at the infrastructure-level include robust capacity planning with autoscaling and load balancing to absorb bursts of requests, combined with throttling mechanisms that prevent any single service instance from being overwhelmed [3]. Application-level defenses enforce hard usage limits: for example, API rate limiting and per-user quotas constrain the number of requests or tokens a client can consume per interval, and I/O size caps (context length and response length limits) ensure no query monopolizes memory or GPU time [10]. Implementing strict timeouts for long-running prompts and limiting queued sequential actions (especially in agentic LLMs that chain tool invocations) prevents runaway loops that exhaust resources. Equally important is continuous monitoring with anomaly detection to flag sudden surges in requests or unusually expensive queries, enabling operators to preempt abuse in real time [7]. Together, these controls maintain model availability and predictable performance under load while constraining the operational costs of potential denial-of-service or "denial-of-wallet" attacks [3, 6].

15. LLM Ethical and Compliance Risks: Mitigating the risk of LLMs producing unethical or non-compliant content requires technical safeguards and strong governance. On the technical side, measures include robust content filtering at multiple stages: blocking or sanitizing toxic, defamatory, or policy-violating inputs and scanning model outputs for disallowed content using rule-based and AI classifiers [3, 8]. Leading toolkits (e.g. NVIDIA's NeMo Guardrails or built-in Azure OpenAI filters) can automatically detect hate, violence, or other harmful text and either refuse or correct such outputs [8], and many LLMs are fine-tuned via safety training (e.g. RLHF) to internalize ethical guidelines and learn to decline improper requests [8]. Complementing RLHF, Anthropic's "Constitutional AI" (used in the Claude family) trains models to follow an explicit set of safety principles via self-critique/revision and reinforcement learning from AI feedback (RLAIF), reducing harmful or non-compliant outputs while maintaining non-evasive responses [150]. Ensuring high-quality, debiased training data and applying detoxification techniques are likewise critical, urging providers to vet fine-tuning datasets for bias or extreme content [10]. Alignment methods like retrieval augmentation and fact-checking pipelines can curb misleading hallucinations by grounding responses in trusted data [10]. On the organizational side, effective mitigations involve instituting clear AI use policies,

content standards, and oversight mechanisms to guide LLM behavior [6]. This includes human-in-the-loop review for high-stakes outputs or any autonomous agentic actions (e.g. requiring human approval before an LLM system executes sensitive tasks) [3], regular red-teaming and adversarial testing to probe the model's ethical guardrails [3, 6], and continuous monitoring/auditing of LLM interactions for compliance issues. Many organizations are establishing dedicated AI governance committees, usage guidelines, and risk assessment workflows to ensure deployments align with legal and ethical requirements [6]. Industry frameworks (from OWASP, CSA, Cisco, FS-ISAC, etc.) echo these practices, emphasizing alignment training, access restrictions, audit logging, and diligent third-party model vetting as key to keeping model behavior within acceptable bounds and maintaining accountability.

16. LLM Resource Consumption: Mitigating the inherent resource demands of LLM deployments requires technical optimizations and organizational strategies. On the technical side, model efficiency can be improved through compression techniques like quantization and pruning, or more efficient model architectures and inference algorithms, which reduce memory and compute overhead without significant loss in accuracy [6, 7]. For example, weight-only 4-bit quantization (e.g., AWQ) shrinks parameter memory while preserving accuracy, often yielding multi-İ speedups, and kernel-level attention optimizations (e.g., FlashAttention-2) reduce memory traffic to yield ù 2İ attention throughput without accuracy loss [151, 152]. Caching frequently requested or repeated responses is another effective approach to avoid redundant computations and make responses more cost-efficient [7]. At the serving layer, smarter memory management (paging and sharing the model's stored context) reduces fragmentation and enables larger batches, improving end-to-end throughput by 2-4İ at similar latency [153]. It is also critical to enforce limits on input size and generation length (e.g., bounding the context window or total tokens) to prevent any single query from consuming excessive resources [8]. At decoding time, speculative decoding accelerates generation via a drafter-verifier scheme while *provably* preserving the target model's output distribution [154]. Such limits can be coupled with continuous resource utilization monitoring and rate-limiting or quota policies to ensure no user or process monopolizes the system's capacity [8]. On the organizational side, robust capacity planning and elastic infrastructure are key. Architectures should support autoscaling and load balancing so that compute resources scale up or down with demand, maintaining performance while avoiding over-provisioning [6]. Organizations can maximize utilization and energy efficiency by dynamically allocating hardware to match workload fluctuations and consolidating resources during lulls. Continuous performance monitoring and proactive provision adjustments further ensure that LLM services run cost-effectively under varying loads [6, 7].

17. LLM Regulatory Compliance: Both traditional LLM applications and more agentic LLM-based systems require a comprehensive compliance strategy balancing technical controls and organizational policies. Key mitigation practices include maintaining robust documentation and audit trails for all LLM operations, ensuring traceability and accountability per regulatory expectations [6, 7, 155]. Equally important are transparency and explainability mechanisms that make the LLM's decision-making processes understandable, facilitating effective oversight and building trust for regulatory audits [6, 155]. Organizations should also conduct thorough pre-deployment conformity assessments (especially for high-risk AI systems) to verify that LLMs meet the requirements of frameworks like the EU AI Act before deployment [155]. To address the risks of automation, human oversight protocols must be established so that human controllers can intervene or guide LLM behavior as needed, a safeguard that is particularly critical for agentic LLMs to prevent erroneous or biased autonomous decisions [6, 10]. Integrating privacy-preserving techniques (e.g., data minimization, encryption, and pseudonymization) into LLM workflows helps protect personal data and ensures compliance with data protection laws such as the GDPR [7, 155]. Meanwhile, continuous monitoring and post-deployment reporting (e.g., incident reporting and periodic audits) should be in place to detect issues, maintain an audit trail of LLM interactions, and enable ongoing risk mitigation, thereby upholding compliance over the system's lifecycle [6, 10]. Clear accountability and risk ownership must be assigned within the organization's AI governance structure, designating responsible parties for managing LLM-related risks and compliance obligations [6, 10]. Finally, given the overlapping obligations of the AI Act, GDPR, NIS2, and other regimes, companies should harmonize their compliance efforts across these legal domains, creating unified governance processes that satisfy multiple regulatory requirements simultaneously [6, 155]. This multi-faceted approach enables organizations to leverage LLM capabilities while adhering to evolving AI regulations.

18. LLM Third-Party Integration Risks: Mitigating LLM Third-Party Integration Risks requires a defense-in-depth approach combining technical safeguards and governance controls. Technically, all runtime integration surfaces (plugins, extensions, external APIs) should operate on a strict zero-trust basis, treating the LLM as an untrusted component and validating or sanitizing any model-generated commands or data passed to external services [3]. Plugins and tools invoked by the LLM must be confined to least-privilege

sandboxes, preventing unauthorized operations and ensuring no harmful or out-of-scope actions can be executed (e.g., disallowing dangerous system calls) [10]. Each plugin's inputs and outputs should undergo rigorous input validation and output filtering, and plugins should be thoroughly tested and reviewed for vulnerabilities or malicious code before deployment [10]. Integration points should enforce strong authentication and authorization; for instance, securing API endpoints with robust AuthN/AuthZ and tokenized credentials, and only permitting calls over encrypted channels, to ensure the LLM or agent can only interact with trusted services under proper credentials [6]. Operationally, organizations should institute oversight frameworks for third-party integrations: conduct dependency reviews and supply chain security assessments for any third-party libraries, models, or APIs (to avoid outdated or vulnerable components) [3], maintain an inventory and allow-list of approved plugins, and require explicit risk assessment and approval for new integrations or expanded LLM tool permissions [6]. Continuous monitoring is also vital: logging and auditing the LLM's calls to external tools and the behavior of plugins in real time can help detect anomalies or policy violations (e.g., unexpected network calls or data exfiltration attempts) and enable prompt incident response [3, 10]. By combining fine-grained permission scoping, plugin sandboxing, strict authentication controls, continuous monitoring, and organizational governance (training staff, defining policies and incident response plans for AI integrations), both general-purpose and agentic LLM deployments can significantly reduce the attack surface associated with third-party APIs, plugins, and auxiliary services [7].

19. LLM Human-Machine Interface Vulnerabilities: Mitigating HMI vulnerabilities in LLM systems requires a multi-layered approach prioritizing safety over usability. Interfaces (graphical, web-based, CLI, embedded) should be designed to clearly demarcate AI outputs and system actions to avoid user confusion or UI spoofing, ensuring that the model cannot misrepresent critical information on screen [10]. All interaction patterns must be unambiguous. For example, this could be achieved by incorporating confirmation prompts or role indicators to clarify potentially dangerous requests (especially in agentic setups) before execution [7]. Strict input validation and output sanitization are essential: treat every user input and LLM output as untrusted by default, applying rigorous checks, content filtering, and context-appropriate encoding to neutralize malicious payloads or code (e.g., escaping model-generated HTML/Markdown in web UIs to prevent XSS and using allow-lists for command names in CLIs to thwart injection) [3]. Data flow management should enforce isolation and oversight between the LLM and connected systems: generative outputs must never be automatically consumed or executed by downstream processes without review or sanitization, as doing so risks backend compromise or unintended commands [10]. In practice, this means sandboxing the LLM's execution environment and limiting its privileges (least-privilege access to files, APIs, or plugins) so that it can only perform pre-approved actions, and instituting a human-in-the-loop for any high-impact or irreversible operations [3]. By embedding such defensive measures across graphical, web, command-line, and embedded interfaces, organizations can significantly reduce interface-driven weaknesses (in design, parsing, or data handling) even if it slightly impedes user convenience [7].

20. LLM Environmental and Sustainability Concerns: Mitigating the environmental footprint of LLMs requires a multi-pronged approach across both model training and inference phases. Technical strategies emphasize energy efficiency: employing optimized model architectures and compression techniques (e.g., quantization and pruning) to reduce computational load without sacrificing performance [6]. During training, practitioners can minimize retraining from scratch by fine-tuning pre-trained models and using algorithmic optimizations, while also scheduling heavy workloads in a carbon-aware manner (aligning computations with periods or regions of cleaner energy supply) [7]. In the inference stage, robust resource management is crucial. For instance, enforcing rate limits, timeouts, and workload throttling prevents runaway consumption of CPU/GPU cycles [3]. Such measures guard against denial-of-service vectors like sponge attacks that intentionally spike energy use [8] and restrict needless power draw. Continually monitoring energy and GPU utilization helps detect anomalies or inefficiencies in real time [10], enabling systems to dynamically adjust for better efficiency (or even gracefully degrade non-critical services under extreme load). Notably, agentic LLM systems demand built-in constraints (on recursion depth, action counts, etc.) to avoid unbounded resource usage as they iterate toward goals. On the organizational side, a commitment to sustainable AI operations is key: organizations should provision models on energy-efficient hardware and choose cloud/data-center providers with renewable energy commitments and superior PUE (power usage effectiveness) ratings [7]. Additionally, instituting a policy of carbon footprint auditing for AI workloads (tracking the CO_2-equivalent emissions of training runs and inference serving) ensures accountability and drives continual optimization. Emerging AI governance frameworks increasingly highlight sustainability as a pillar of responsible AI deployment, alongside safety and ethics [155]; therefore, prioritizing green infrastructure and efficiency-oriented design not only reduces environmental impact but also aligns LLM deployments with broader organizational ESG goals and upcoming regulatory expectations.

7. Associative and causal relationships between threats

Beyond grouping threats by life-cycle phase, it is critical to understand how threats interact, both associatively and causally, because most real incidents unfold as chains, not single events. Fig. 2 (View A) summarizes these interactions at the category level (training, inference, system), while Fig. 3 (View B) re-weights nodes by overall connectivity (total graph degree) to highlight highly connected threats (hubs) in the dependency map. To support auditability, we provide evidence pointers for representative causal edges in Table 5 and an expanded edge-evidence pack in Appendix C. To make this map immediately reusable for evaluation, we also provide a minimal end-to-end vignette that converts a representative causal red path into seeds, variants ("seeds-of-seeds"), a success oracle, and a pre/post-mitigation ASR template (Section 7.7).

7.1. Semantics of links

We distinguish two edge types. *Associative (blue) edges* indicate threats with some form of similarity or overlap; they flag clusters that frequently appear in one high-level category in case studies or are addressed by overlapping mitigations. They are not directional and should not be read as cause-and-effect paths. *Causal (red) edges* encode a directional relationship where Threat A increases the likelihood of or enables Threat B via a plausible and documented mechanism (e.g., a poisoned vector store that deterministically steers RAG to unsafe context, which then triggers improper output handling). Arrow color intensity reflects the total graph degree (number of connections) of the origin node in our evidence base (Section 5).

7.2. Edge derivation and evidence pointers

The edge set is derived from the incident case studies in Section 5. For each incident or controlled demonstration, we annotated (i) which threat classes are present and (ii) whether the narrative describes a directional enabling relation. We add a causal edge $A \rightarrow B$ only when at least one case study explicitly describes (or directly implies) a plausible mechanism by which exploiting A makes B possible or materially increases its likelihood. We treat a distinct incident or controlled demonstration as an *independent observation*; when multiple outlets report the same underlying event, we count it once and retain the most technical/primary write-up(s) as evidence pointers. Associative edges are added when threats consistently co-occur in incident narratives or are semantically similar. Because incident reporting is heterogeneous, the visualization uses degree-based scaling for readability rather than attempting to encode per-edge evidence strength; auditability is provided via explicit evidence pointers in Table 5 and Appendix C.

Table 5
Representative causal edges from the dependency map with incident-derived evidence pointers. An expanded evidence pack is provided in Appendix C.

Causal edge	Mechanism sketch	Incident-derived evidence pointers (Section 5)
03→05	Injected instructions cause the model/agent to reveal secrets in protected context or tool outputs (exfiltration).	Operator exfiltration [54]; EmailGPT CVE demonstration [96, 132].
06→05	Unsanitized outputs are rendered/executed downstream, leaking data via channels like URLs, caches, or markup.	ChatGPT Redis bug exposing chats [62, 122]; exfiltration via remote image tags in integrated assistants [69, 125].
07→08	Retrieval/embedding poisoning steers RAG to attacker-controlled context, producing misleading answers.	RAG poisoning case study [71]; retrieval hijack via adversarial perturbations [72].
07→05	Embedding inversion or unsafe vector-store access enables reconstruction of sensitive source text.	Embedding inversion demonstration [24].
18→13	Over-permissive tool/plugin integrations give the LLM agency; adversarial content can trigger unauthorized actions (up to malware execution).	ChatGPT plugin cross-plugin abuse [95]; Auto-GPT escapes via tool misuse [97, 126]; Claude "Computer Use" malware demo [127].

7.3. Diagram significance

The relationship graphs serve four concrete purposes for both researchers and operators:

- ***Attack-path enumeration & wargaming.*** Causal chains expose the most probable multi-step exploit flows (e.g., *Prompt Injection* → *Data Leakage* → system compromise), enabling scenario design and tabletop exercises that reflect realistic sequencing rather than isolated tests.
- ***Prioritization under constraints.*** Highly connected nodes (high graph degree) act as hubs in the dependency map: hardening them and instrumenting their attached red paths can reduce risk across multiple downstream threats under a limited engineering budget.
- ***Control coverage mapping.*** Because associative edges often share mitigations, a single control family (Table 3) can be justified across multiple threat classes, improving return on investment and reducing policy sprawl. Associative clusters should therefore be treated as *hardening neighborhoods* when planning red-team scope and defense roadmaps.
- ***Benchmark design.*** The graphs guide evaluation suites: pick seeds along dominant red paths, then add variants that traverse associative neighborhoods to test generalization. Section 7.7 provides a minimal example (seeds/seeds-of-seeds, a success oracle, and a pre/post-mitigation ASR reporting template).

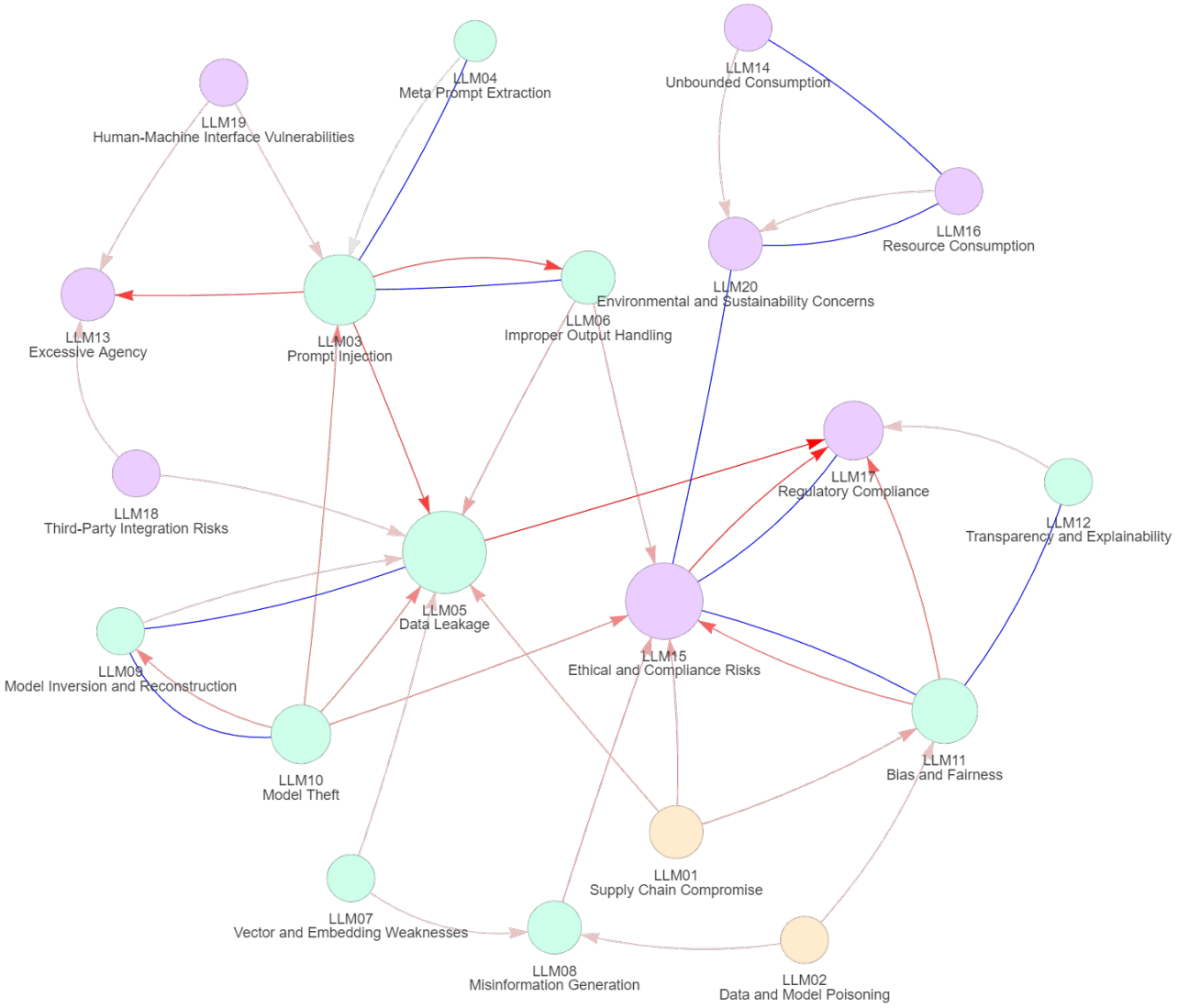

Figure 2: View a (emphasis on categories): Associative and causal relationships between threats. **Note:** Blue connections represent associative relationships, red connections represent causal. The node colors correspond to their threat category. Arrow color intensity increases with the total graph degree (number of connections) of the origin node.

7.4. Reading view A vs. view B

View A highlights how cross-phase propagation occurs (training → inference → system). It makes visible, for instance, that upstream *Supply Chain Compromise* and *Data/Model Poisoning* can causally surface as *Misinformation* or *Bias & Fairness* problems at inference, which in turn associate with *Regulatory/Ethical* exposure at the system level. View B emphasizes individual threat centrality: nodes with larger/higher-hue glyphs have higher overall connectivity (graph degree). They thus are prime candidates for (i) tighter runtime monitoring and (ii) pre-merge security gates (pull request (PR) checks[4]) that exercise their attached red paths.

[4]Pull Request Checks are automated continuous integration gates that run on every PR and block merging on failure.

7.5. Recurring cascade motifs in the incident corpus

Across the incidents in Section 5, we observe recurring causal motifs that explain why certain edges appear frequently in the map and which control families cut multiple paths:

- ***Interface-driven injection cascades.*** UI/context-boundary failures allow attackers to inject privileged instructions via the interface (19→03) [96, 128–132], which can then lead to leakage of protected context or tool outputs (03→05) [54, 96, 132] and/or to unsafe downstream rendering/execution pathways (03→06) [68, 69, 124] in integrated applications. This motif is primarily cut by CF4 (prompt/context guardrails), CF6 (tool/agent

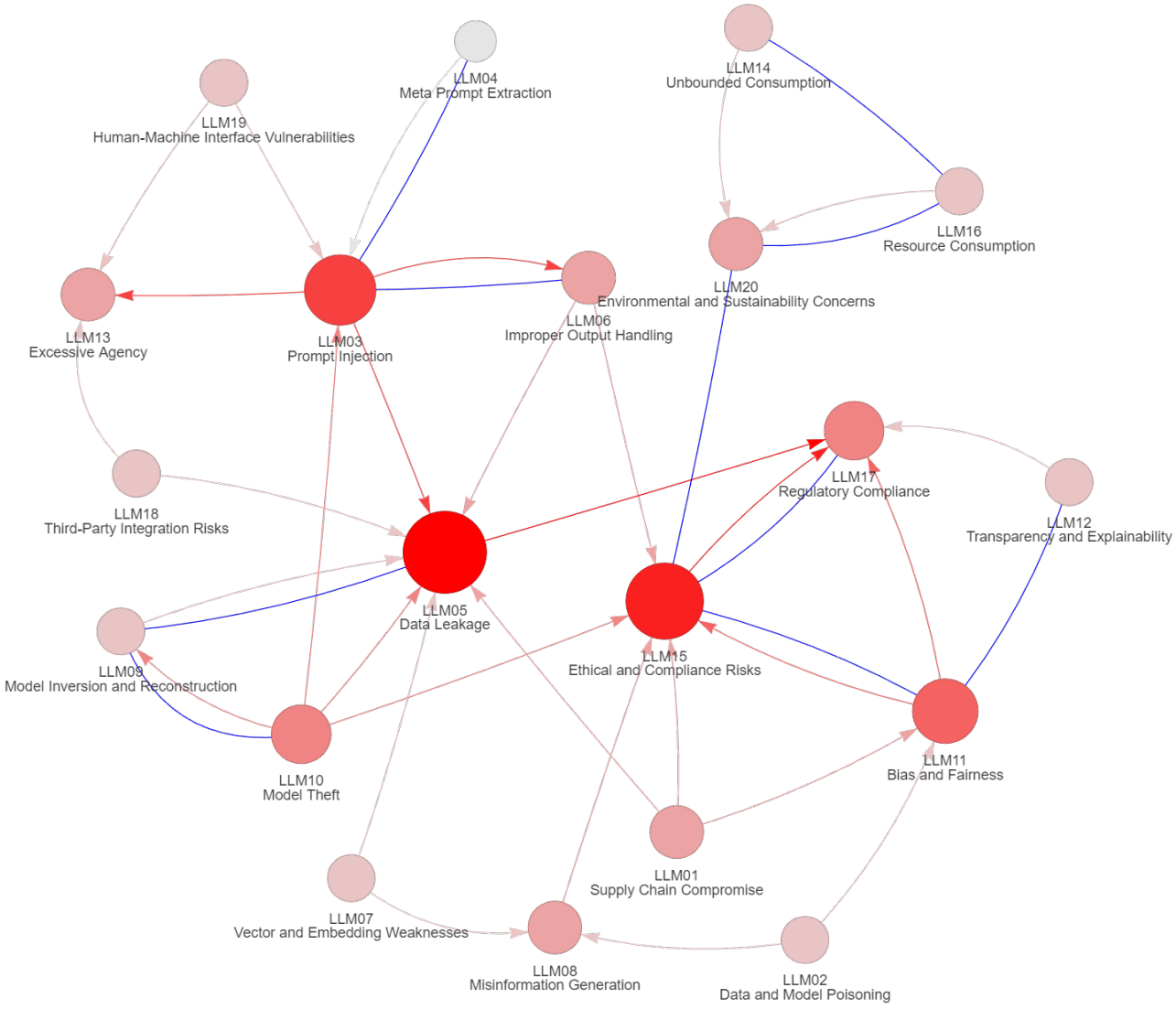

Figure 3: View B (emphasis on threat level): Associative and causal relationships between threats. **Note:** Blue connections represent associative relationships, red connections represent causal. Node hue and size increase proportionally with the node's total graph degree (number of connections).

and integration security), and CF7 (monitoring and incident response).

- ***Retrieval/embedding compromise cascades.*** Poisoned or adversarially perturbed retrieval corpora can steer RAG outputs (07→08), while embedding inversion and unsafe vector-store access can reconstruct sensitive source text (07→05) [24, 71, 72]. This motivates CF5 (grounding/retrieval security) combined with CF2/CF7 controls.
- ***Training-time compromise surfacing at inference.*** Data/model poisoning can implant backdoors that manifest as targeted misinformation at inference (02→08) [46–48], reinforcing the need for CF1/CF3 controls even when deployment-time guardrails are present.

7.6. From map to action

Practically, we use the graphs to (i) rank threats by node degree (overall connectivity) for detection budget and guardrail placement; (ii) select seed attacks that traverse the longest/most plausible red paths; (iii) verify that controls "cut" critical paths (pre-/post-mitigation ASR deltas along each path; see Section 7.7 for a minimal end-to-end example); and (iv) document residual risk where only associative evidence exists, reserving causal labels for chains with observed mechanisms. The interactive HTML (released with the camera-ready) exposes phase, centrality, and path length filters to reproduce these analyses.

7.7. Minimal end-to-end evaluation vignette

To complement the qualitative case studies with a reusable measurement template, we provide a minimal end-to-end vignette for the causal path *19 (Human-Machine Interface Vulnerabilities) → 03 (Prompt Injection) → 05 (Data Leakage)*. This path captures a common deployment failure mode: untrusted external content is ingested via integrations (e.g., email/web/plugins), enabling *indirect* prompt injection and potentially culminating in the exfiltration of sensitive information.

System under test An LLM-assisted "content QA" application that (i) ingests external content C (e.g., an email body or webpage), (ii) appends a user request U (e.g., "summarize" / "answer question"), and (iii) produces an output O that may be displayed or forwarded to downstream components. The application also maintains *protected context* P (e.g., system instructions, retrieved internal documents, or tool outputs) that should not be disclosed.

Seeds and seeds-of-seeds We define a seed set $\mathcal{S}$ of canonical indirect prompt-injection payloads embedded in C (e.g., attack templates drawn from indirect prompt-injection benchmarks such as BIPIA [136]). For each seed s Ë $\mathcal{S}$, we generate a small set of variants $\mathcal{V}.s/$ ("seeds-of-seeds") via simple, reproducible transformations such as: (i) paraphrasing the embedded instruction, (ii) formatting/encoding perturbations, and (iii) multi-turn decompositions (splitting a single instruction across multiple turns or content segments). The evaluation set is $_{s\ddot{E}\mathcal{S}}\,\mathcal{V}.s/$.

Success oracles We use explicit pass/fail rules (oracles) to make evaluation reproducible. For indirect prompt injection success (19→03), we define an oracle $\mathcal{O}_{\text{inj}}.O/$ that returns 1 if the model follows attacker-inserted instructions in C (i.e., the model follows the attacker-inserted instruction in C as determined by an explicit judge, e.g., target-string/rule checks or an LLM-as-judge, consistent with benchmark practice [136]), and 0 otherwise, consistent with prior benchmark practice [136]. To quantify downstream leakage (03→05), we define a leakage oracle $\mathcal{O}_{\text{leak}}.O/$ that returns 1 if a canary secret token τ (planted in protected context P) appears in O (exact or fuzzy match), and 0 otherwise.

Metric and mitigation assessment Attack success rate is computed as ASR $= \frac{1}{N}\sum_{i=1}^{N}\mathcal{O}.O_i/$, yielding ASR_{inj} and ASR_{leak} for the two oracles above. The template evaluates ASR under a baseline configuration and under defended configurations (e.g., boundary delimitation between U and C, multi-turn separation, or explicit content marking). We report scenario-specific ASR_{inj} for *EmailQA* and *WebQA* because they directly operationalize the interface/context-boundary attack surface of this vignette: in EmailQA the model answers questions over untrusted email content, and in WebQA it answers questions over untrusted web content, both of which are common carriers of indirect prompt injection in integrated LLM applications. Table 6 reproduces illustrative ASR_{inj} values from BIPIA for ChatGPT-3.5-Turbo under selected black-box defenses.

Table 6
Illustrative ASR template for *indirect prompt injection*. Values are reproduced from BIPIA (ChatGPT-3.5-Turbo; Table 5 in [136]) under selected black-box defenses to illustrate the reporting template; we do not report new experimental results here. EmailQA and WebQA correspond to BIPIA's email QA and web QA scenarios (QA over untrusted email/web content), and overall ASR is averaged across all BIPIA scenarios/tasks. BIPIA reports these values under the defense setting that yields the lowest overall ASR when multiple settings exist; we reproduce the values exactly as reported in their Table 5. The same harness can be instantiated with a leakage-oriented oracle (e.g., canary-token exposure via $\mathcal{O}_{\text{leak}}$) to quantify downstream data-leakage risk along the causal path 19→03→05.

Configuration	EmailQA ASR	WebQA ASR	Overall ASR
Original (no defense)	0.1025	0.1025	0.1690
Border strings (boundary delimiter)	0.0439	0.0558	0.1078
Multi-turn dialogue	0.0907	0.0593	0.0765
Datamarks (content marking)	0.0251	0.0548	0.1090

8. Discussion, limitations & future directions

8.1. Discussion

This work presents a comprehensive, lifecycle-centric taxonomy of cyber threats targeting LLM systems. By organizing 20 distinct threat classes into training-time, inference-time, and system-level categories (see Table 2 and the tree diagram in Fig. 1), we provide a unified framework that synthesizes and extends existing industry and academic taxonomies (CSA, Cisco AI Defense, OWASP, MITRE ATLAS, FS-ISAC) into a single coherent structure. This structure not only clarifies the attack surface at each stage of the AI lifecycle but also highlights the interdependencies and causal relationships among threats (Figs. 2 and 3), revealing which vulnerabilities warrant the highest defensive priority.

The case studies (Section 5) concretely demonstrate how these threats manifest in operational environments. They reveal that seemingly innocuous gaps, such as lax vetting of third-party data providers or overly permissive inter-agent communication, can be weaponized by adversaries. Moreover, they highlight the dynamic nature of AI adversarial tactics: attackers continually adapt to bypass hardened prompts or sandbox restrictions, necessitating equally adaptive defenses.

The Mitigations framework (Section 6) directly addresses this interconnected threat landscape. By combining rigorous supplier vetting and cryptographic signing during training with prompt hardening, sandboxed I/O pipelines, and least-privilege architectures at inference, and bolstering them with continuous monitoring and human-in-the-loop governance at the system level, organizations can erect multiple choke points that adversaries must bypass. The synergy of these measures, rather than isolated point solutions, drives resilience even as attackers adapt.

8.2. Limitations

Our focus on single-agent autonomy and simple multi-agent interactions necessarily omits more sophisticated agentic architectures. Self-modifying agents, federated collectives, and emergent behaviors in open-ended multi-agent systems may introduce novel threat vectors, such as dynamic coalition attacks or stealthy reward-poisoning, that lie beyond the scope of our current taxonomy.

The adversarial landscape for LLMs evolves rapidly, with new techniques emerging faster than taxonomy updates can capture, such as semantic backdoors embedded in vector representations, cross-modal transfer attacks, or exploitation of model-in-the-loop orchestration software. As a result, our static threat classes may lag behind cutting-edge adversarial innovations unless continually revised. However, the code for generating threat maps seen in Figs. 2 and 3 can be updated to accommodate more threats and connections.

Finally, although the proposed taxonomy advocates layered technical and governance controls, the broader regulatory and ethical landscape is outside the scope of this research. However, we argue that real-world implementations must contend with complex compliance regimes (data privacy laws, AI accountability mandates, industry-specific standards) and evolving ethical frameworks around bias, transparency, and human oversight. These factors will critically shape which threats are prioritized and how defenses are operationalized.

8.3. Future directions

We foresee valuable findings arising from researching the following areas to close these gaps and stay ahead of adversarial innovation.

Dynamic Threat Detection. Future work should couple high-fidelity runtime telemetry with provenance signals to detect novel attack chains *as they unfold*. Concretely, we see value in fusing (i) content-layer indicators (prompt-injection patterns, jailbreak signatures, unexpected prompt-leakage phrases), (ii) behavioral indicators from tools/agents (atypical plugin invocation graphs, risky API call sequences), and (iii) infrastructure indicators (token-time anomalies, "sponge" energy/latency spikes) into streaming detection pipelines. This multi-layer fusion is motivated by recent demonstrations that prompt-injection and meta-prompt extraction generalize across apps and can be automatically generated [2, 16, 23], that Remote Code Execution (RCE)-class flaws can be triggered by model text alone in common integration frameworks [26], and that worst-case inputs can deterministically explode inference cost [34]. Extending detectors to vector stores (e.g., spotting retrieval drift from embedding poisoning) will further reduce mean-time-to-detect and mitigate cascading failures in RAG pipelines [27].

Automated Mitigation Workflows. Detection without rapid, consistent *response* invites repeated compromise. We argue for policy-driven, closed-loop mitigations that automatically: (a) gate privileges and tools (least-privilege Role-Based Access Control (RBAC) and plugin allow-lists), (b) hot-patch prompts and sanitizers (inject deny-rules, rotate honey tokens, quarantine memory), (c) reshape traffic (rate-limit, cap context/response sizes, force human-in-the-loop), and (d) raise audit artifacts (signed decisions for post-mortems). These controls align with OWASP/CSA/Cisco guidance and ATLAS TTPs[5] [3, 6, 7, 16] and are reinforced by real incidents (e.g., "denial-of-wallet"/key-abuse and text-driven RCE) that justify automated kill-switches and throttling [26, 35].

Standardized Evaluation Frameworks. Security is stuck without agreed-upon, reproducible measurements. We recommend open, lifecycle-spanning suites that report: *ASR* for prompt-injection/leakage (white- and black-box), *LLM-RCE* exploitability for output-handling chains, *retrieval integrity* under embedding poisoning, *privacy leakage* under membership/inversion probes, and *cost/latency blow-up* under adversarial inputs. Recent work already formalizes prompt-injection and proposes risk taxonomies/benchmarks [2, 8], while ATLAS/FS-ISAC offers procedure-level scaffolding to standardize red-team tasks [10, 16]. Hallucination surveys and training-data-extraction studies provide validated protocols and metrics to plug into such suites [24, 27, 28]. A community baseline (with seeds, seeds-of-seeds, and harness code[6]) would make results comparable and defensible.

Enterprise Integration & Governance. Most incidents materialize at integration boundaries. We therefore call for "AI-DevSecOps": enforce SBOM/provenance in Continuous Integration and Continuous Delivery (CI/CD), code-sign models and prompts, vet and sandbox plugins/APIs, and codify

[5] Tactics, Techniques and Procedures.

[6] A *seed* is a canonical attack/test case used for reproducible evaluation. We use the shorthand *seeds-of-seeds* for deterministic variants of a seed (e.g., paraphrases, obfuscations/encodings, or context-wrapped carriers); formally, for the i-th seed s_i we evaluate a finite set of variants $V.s_i/$. The *harness* is the executable runner that orchestrates end-to-end tests (state reset, environment/version pinning, oracle-based success checks) and logs metrics such as ASR.

policies as testable guardrails in PR checks and deployment gates. Case studies of third-party/library failures and plugin eco-system flaws show why this must be continuous, not one-off [44, 62, 123]. Governance should map controls to external obligations (audit trails, human oversight triggers, Data Protection Impact Assessment (DPIA)-style privacy records) so that security posture is natively compliant with emerging regulation [3, 6, 120, 155].

Regulatory and Ethical Landscape. Finally, research should align security controls with the evolving regulatory stack and normative expectations. The AI Act codifies risk-management, technical documentation, transparency, logging, and human oversight duties; GDPR and national actions illustrate enforcement measures (bans, fines, and market restrictions) [110, 117, 120]. Ethically, fairness and safety remain live issues, with persistent demographic skews and domain-specific harms motivating routine bias audits, dataset/process transparency, and explicit oversight checkpoints [28, 32, 88]. Embedding these requirements into evaluation harnesses[7] and automated workflows reduces operational burden while improving legal defensibility and social acceptability [155].

By embedding our taxonomy and defense-in-depth playbook into both research and practice, and by systematically evaluating and iterating on these controls, the community can deploy LLM systems with confidence in their security, reliability, and trustworthiness.

9. Conclusions

In this paper, we present a unified, lifecycle-centric taxonomy of LLM threats that clarifies 20 distinct attack vectors across training-time, inference-time, and system-level phases. Our three main contributions are: (1) consolidation of disparate industry and academic taxonomies into a single framework that maps interdependencies among threat classes; (2) empirical demonstration (via real-world case studies) of how these threats manifest and cascade across the AI lifecycle; and (3) a defense-in-depth playbook that aligns cryptographic artifact verification, prompt hardening, sandboxed I/O, and least-privilege governance to each threat phase. By coordinating these layered mitigations, stakeholders can preemptively block adversarial tactics and build more resilient, trustworthy AI deployments. Going forward, developing standardized evaluation benchmarks and real-time adaptive monitoring will be critical to staying ahead of rapidly evolving adversary techniques.

[7] A framework or tool used to assess the performance of large language models.

CRediT authorship contribution statement

Ioannis Stylianou: Conceptualization, Resources, Methodology, Investigation, Formal Analysis, Visualization, Writing – original draft, Writing – review & editing. **Panagiotis Bountakas:** Supervision, Writing – review & editing. **Apostolis Zarras:** Supervision, Writing – review & editing. **Aristeidis Farao:** Writing – review & editing. **Vaios Bolgouras:** Writing – review & editing. **Christos Xenakis:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research has been partially funded by the European Union's research and innovation programs under grant agreements No. 101119602 (COBALT), No. 101168407 (CPAID), and No. 101139031 (SAFE-6G). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the Health and Digital Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

A. Search strategy, query templates, and screening summary

A.1. Scope, sources, and time window

We conducted a structured literature search covering 2015–2025 across major bibliographic databases (Google Scholar, ACM Digital Library, arXiv, IEEE Xplore, SpringerLink, Elsevier) and gray-literature sources (OWASP, CSA, MITRE ATLAS, FS-ISAC, and vendor/security whitepapers). We targeted peer-reviewed papers, preprints, standards/technical reports, and influential practitioner frameworks that analyze threats, vulnerabilities, risks, incidents, or mitigations for LLMs, foundation models, and LLM-based agentic systems. To support reproducibility, we logged for each query: (i) the platform/source, (ii) the raw query string, (iii) execution date, (iv) filters applied (year range, field restrictions), and (v) the number of hits returned at query time.

A.2. Controlled vocabulary and consistency across sources

To ensure term consistency while accommodating platform-specific search interfaces, we used a controlled vocabulary organized into *logical query blocks*. Each query included at least: (i) one **model/system** term

Table 7
Controlled vocabulary used to construct consistent search queries across heterogeneous platforms.

Term family	Representative keywords (OR-ed within family; wildcards where supported)
Model / foundation-model block (B_M)	"large language model"; LLM; "foundation model"; "generative AI"; GenAI; "language model"; transformer; "instruction-tuned"; chatbot; "conversational AI"; "multimodal model"; VLM; "code model"; "AI copilot"
Agentic / integration block (B_A)	agentic; agent; "AI agent"; "LLM agent"; "autonomous agent"; "multi-agent"; "tool use"; "tool-using"; "function calling"; plugin*; workflow; orchestrat*; planner; ReAct; AutoGPT; LangChain; LlamaIndex; RAG; "retrieval augmented generation"; "vector database"; embedding*
Security / threat block (B_S)	security; threat*; vulnerab*; risk*; attack*; exploit*; incident*; adversarial; "red team*"; jailbreak*; "prompt injection"; "indirect prompt injection"; poison*; backdoor*; trojan*; "model extraction"; "model stealing"; "model inversion"; "membership inference"; leak*; exfiltrat*; DoS; "denial of wallet"
Secondary-study / framework block (B_F)	taxonomy; classification; framework; survey; review; "systematization of knowledge"; SoK; "systematic review"; "systematic mapping"; "mapping study"; "scoping review"; "literature review"; "threat model"; "risk assessment"; "threat landscape"; guideline*; "best practice*"; playbook; checklist; "top 10"; atlas; "risk map"

block, (ii) one **security** term block, and (iii) (when searching specifically for surveys/frameworks/SoKs) one **secondary-study/framework** term block (Table 7).

A.3. Canonical boolean query templates (platform-agnostic)

We instantiated the following canonical templates, adapting only the syntax (quotes, field selectors, wildcard rules) per platform:

Q_{core} (LLM security, general):

```
(BM) AND (BS)
```

$Q_{framework}$ (find surveys/SoKs/taxonomies/ frameworks):

```
(BM) AND (BS) AND (BF)
```

$Q_{broad\text{-}ML}$ (capture AI/ML security frameworks that may index LLMs under ML/AI)

```
("AI" OR "machine learning" OR "ML" OR "adversarial
     machine learning")
AND (BS) AND (BF)
AND ("foundation model" OR "generative AI" OR "language
     model" OR LLM)
```

$Q_{agentic}$ (agent/tool/RAG exposures):

```
(BM OR BA) AND (BS) AND (BF OR "case study" OR incident
     *)
```

$Q_{incident}$ (real-world/security reporting focus; used for incident identification and triangulation):

```
(BM OR BA) AND (incident* OR breach OR advisory OR CVE
     OR "post-mortem"
OR "root cause" OR "security update" OR "responsible
     disclosure")
```

A.4. Representative per-source query instantiations

Because advanced-search interfaces differ, we used the same *logical blocks* but tailored the query surface form per source. Below are representative examples for $Q_{framework}$ (field restrictions applied where available to Title/Abstract/Keywords to reduce noise):

Google scholar (all fields; year filter 2015–2025)

```
("large language model" OR LLM OR "foundation model" OR
     "generative AI")
AND (security OR threat* OR vulnerab* OR risk* OR attack
     * OR adversarial
OR incident* OR "prompt injection" OR jailbreak* OR
     poison*)
AND (taxonomy OR framework OR survey OR review OR "
     systematization of knowledge"
OR SoK OR "systematic review" OR "mapping study" OR "
     threat landscape")
```

IEEE xplore (fielded: abstract/Index terms/Metadata where supported; years 2015–2025)

```
("All Metadata":"large language model" OR "All Metadata
     ":LLM OR "All Metadata":"foundation model")
AND ("All Metadata":security OR "All Metadata":threat OR
      "All Metadata":vulnerability
OR "All Metadata":"prompt injection" OR "All Metadata":
     poisoning)
AND ("All Metadata":taxonomy OR "All Metadata":framework
      OR "All Metadata":survey
OR "All Metadata":"systematization of knowledge" OR "All
      Metadata":SoK)
```

ACM digital library (Advanced search; title/Abstract rows; years 2015–2025)

```
Title: ("large language model" OR LLM OR "foundation
     model" OR "generative AI")
AND
Abstract: (security OR threat* OR vulnerab* OR risk* OR
     attack* OR adversarial
OR "prompt injection" OR jailbreak* OR poison*)
AND
(Title OR Abstract): (taxonomy OR framework OR survey OR
      review OR SoK
OR "systematization of knowledge" OR "systematic review
     ")
```

ArXiv (title/abstract where supported; years 2015–2025)

```
(ti:"large language model" OR abs:"large language model"
      OR all:LLM OR all:"foundation model")
```

Table 8
Threat-class aliases used for targeted follow-up searches after the taxonomy stabilized.

ID	Threat class (canonical)	Representative aliases/targeted keywords used in follow-up searches
1	LLM Supply Chain Compromise	"supply chain"; "dependency confusion"; "malicious package"; "trojaned model"; "model hub"; "artifact signing"; SBOM; MLBOM; SLSA; "CI/CD compromise"
2	LLM Data and Model Poisoning	"data poisoning"; "training data poisoning"; "instruction-tuning poisoning"; "dataset contamination"; backdoor; trojan; "model poisoning"; "trigger"
3	LLM Prompt Injection	"prompt injection"; "indirect prompt injection"; "instruction injection"; "prompt hijacking"; jailbreak; "prompt hacking"
4	LLM Meta Prompt Extraction	"system prompt"; "prompt leakage"; "prompt stealing"; "hidden prompt"; "meta prompt"; "instruction leak"
5	LLM Data Leakage	"sensitive data disclosure"; "privacy leakage"; PII; regurgitation; memorization; "training data leakage"; "context leak"
6	LLM Improper Output Handling	"unsafe output"; "untrusted output"; "output validation"; "tool output injection"; "code generation risk"; "SQL generation"
7	LLM Vector and Embedding Weaknesses	embedding; "vector store"; "vector database"; "RAG poisoning"; "retrieval manipulation"; "embedding inversion"
8	LLM Misinformation Generation	misinformation; disinformation; "fake news"; propaganda; defamation; "hallucination" (when framed as harm)
9	LLM Model Inversion and Reconstruction	"model inversion"; "data reconstruction"; "training data reconstruction"; "membership inference"; "attribute inference"
10	LLM Model Theft	"model extraction"; "model stealing"; "model theft"; "IP theft"; distillation; "weight stealing"
11	LLM Bias and Fairness	bias; fairness; discrimination; stereotype; toxicity; hate speech
12	LLM Transparency and Explainability	transparency; explainability; interpretability; auditability; XAI
13	LLM Excessive Agency	"excessive agency"; "over-privileged"; "autonomous agent"; "tool use risk"; "agentic misuse"
14	LLM Unbounded Consumption	"unbounded"; "runaway"; "token explosion"; "infinite loop"; "billing abuse"; "denial of wallet"
15	LLM Ethical and Compliance Risks	ethics; misuse; copyright; plagiarism; "policy violation"; "responsible AI"
16	LLM Resource Consumption	DoS; "denial of service"; "energy-latency"; "sponge"; "compute exhaustion"
17	LLM Regulatory Compliance	GDPR; "EU AI Act"; "AI regulation"; NIS2; "risk management framework"; "conformity assessment"
18	LLM Third-Party Integration Risks	plugin; extension; API; "third-party"; "tool invocation"; "agent framework"
19	LLM Human-Machine Interface Vulnerabilities	HMI; UI; UX; "interaction design"; "input validation"; "prompt boundary"
20	LLM Environmental and Sustainability Concerns	energy; carbon; emissions; sustainability; "green AI"; "environmental impact"

```
AND (all:security OR all:threat OR all:vulnerability OR
     all:attack OR all:"prompt injection"
OR all:jailbreak OR all:poisoning OR all:"model
     extraction" OR all:"model inversion")
AND (all:taxonomy OR all:framework OR all:survey OR all:
     review OR all:SoK
OR all:"systematization of knowledge")
```

SpringerLink / elsevier (advanced search; title/Abstract/Keywords where supported; years 2015–2025)

```
("large language model" OR LLM OR "foundation model" OR
     "generative AI")
AND (security OR threat* OR vulnerab* OR risk* OR attack
     * OR adversarial OR incident*)
AND (taxonomy OR framework OR survey OR review OR SoK OR
      "systematic review")
```

A.5. Framework- and taxonomy-seeking keyword expansions

To directly capture practitioner frameworks and structured threat listings (which may not use academic terminology), we expanded B_F with terms such as: `"top 10"`, `atlas`, `"risk map"`, `"control framework"`, `"security standard"`, `"reference architecture"`, `"assurance"`, `"evaluation benchmark"`, `"mitigation catalog"`, `"playbook"`, and `"best practices"`. We also ran targeted framework-name queries (e.g., `OWASP`, `MITRE ATLAS`, `CSA`, `FS-ISAC`, `NIST AI RMF`, `"EU AI Act"`) combined with B_S and B_F terms to retrieve authoritative guidance documents and crosswalks.

A.6. Targeted follow-up searches per threat class

After converging on the final 20 threat classes, we performed follow-up searches using each class name and common aliases/synonyms to capture specialized work that might not appear under generic `LLM security` queries. Table 8 lists representative aliases used for query expansion.

A.7. Screening, deduplication, and inclusion/exclusion criteria

We applied a two-stage screening process (title/abstract, then full text) and removed duplicates using DOI matching where available, and otherwise normalized-title matching. We included sources that (i) study LLMs/foundation models directly, or (ii) propose AI/ML security taxonomies/frameworks that explicitly map to LLM-based or GenAI systems. We excluded (i) purely marketing content, (ii) papers where AI security is only tangential (e.g., LLM used merely as a demo with no security contribution), and (iii) generic ML-security papers that do not provide an explicit

Table 9
Representative boundary cases discussed during consensus coding. Each case was resolved using our decision rules (primary attacker objective, required access/capability, lifecycle locus of control, and mitigation surface).

Boundary case	Why kept separate (decision rationale)
Threat 03 *Prompt Injection* vs Threat 04 *Meta Prompt Extraction*	Injection targets *behavioral control* (override system/developer intent and induce prohibited actions/outputs), whereas meta prompt extraction targets *confidentiality of hidden instructions* (system prompts/constraints). Although both may share techniques (e.g., crafted prompts/indirect injections), they differ in success criteria and mitigation emphasis (instruction isolation and tool-guarding vs prompt secrecy and prompt boundary hardening).
Threat 05 *Data Leakage* vs Threat 09 *Model Inversion and Reconstruction*	Data leakage captures *passive memorization/exposure* (the model outputs memorized sensitive content under prompting), while inversion/reconstruction captures *active analytical extraction* (strategic querying to infer or reconstruct sensitive training attributes/inputs/parameters). We split them because attacker capability and evaluation differ (opportunistic extraction vs systematic inference), and mitigations emphasize different controls (data governance and privacy-preserving training vs oracle hardening, query monitoring, and output obfuscation).
Threat 09 *Model Inversion and Reconstruction* vs Threat 10 *Model Theft*	Inversion focuses on reconstructing sensitive training data or internal characteristics via inference behavior; theft aims to *replicate the model's functionality or exfiltrate model artifacts/weights* (IP theft). They are related but distinct in attacker objective and downstream impact: inversion is primarily a privacy/confidentiality attack; theft is primarily an intellectual-property and capability replication attack (often enabling offline probing and misuse).
Threat 01 *Supply Chain Compromise* vs Threat 18 *Third-Party Integration Risks*	Supply chain compromise refers to *upstream tampering* with foundational artifacts (datasets, checkpoints, libraries, build pipeline components) prior to deployment, whereas third-party integration risks arise *at runtime* from insecure plugin/API/service interactions. We kept them separate because the trust boundary and assurance mechanisms differ (provenance/integrity verification and secure build practices vs runtime authentication/authorization, sandboxing, and continuous monitoring of external calls).
Threat 14 *Unbounded Consumption* vs Threat 16 *Resource Consumption*	Unbounded consumption is *adversarially induced* (malicious prompts/traffic designed to trigger worst-case inference or excessive token generation leading to DoS/cost spikes). Resource consumption captures *baseline operational load/inefficiency* absent an adversary. This split matters because threat models and mitigations differ (rate limiting/quotas/input caps vs optimization/capacity planning/cost engineering).
Threat 16 *Resource Consumption* vs Threat 20 *Environmental and Sustainability Concerns*	Resource consumption reflects the operator's operational/economic burden (compute, latency, spend), whereas sustainability concerns capture broader *environmental externalities* (energy/carbon footprint and long-term ecological impact). They overlap in underlying drivers but differ in impact scope, metrics, and governance/mitigation framing (FinOps/efficiency vs carbon-aware computing, green AI practices, sustainability reporting).
Threat 11 *Bias and Fairness* vs Threat 15 *Ethical and Compliance Risks* vs Threat 17 *Regulatory Compliance*	We separated these governance-related classes by scope: bias/fairness concerns *discriminatory outputs and skew* learned from data; ethical/compliance risks cover broader organizational and societal harms (policy violations, unsafe or socially harmful outputs) that may occur even without legal violation; regulatory compliance covers *formal legal obligations* and enforcement exposure. The split reflects different evaluation methods and organizational owners (fairness audits vs governance and oversight vs legal/compliance controls).

connection to LLM/foundation-model deployment settings.

B. Boundary-case decisions from consensus coding

Table 9 summarizes representative boundary cases that required adjudication during coding, together with the criteria used to keep categories separate. These cases correspond to distinctions already articulated in the threat descriptions in Section 4 to ensure the taxonomy remains internally consistent and auditable.

C. Edge-evidence pack for causal edges

To improve the auditability of the dependency map, this appendix provides supporting evidence pointers for *all causal (red)* edges implemented in our visualization script. Each edge is grounded in at least one incident narrative from Section 5. When multiple outlets report the same underlying incident, we cite the most technical/primary source(s) available (Table 10).

Data availability

No data were used for the research described in the article.

References

[1] Tracebit, Code execution through deception: gemini AI CLI hijack, 2025. https://tracebit.com/blog/code-exec-deception-gemini-ai-cli-hijack.
[2] Y. Liu, Y. Jia, R. Geng, J. Jia, N.Z. Gong, Formalizing and benchmarking prompt injection attacks and defenses, in: 33rd USENIX Security Symposium (USENIX Security 24), 2024, pp. 1831–1847.
[3] OWASP, OWASP top 10 for large language model applications (v2025), OWASP Foundation, 2025. https://owasp.org/www-project-top-10-for-large-language-model-applications/.
[4] P. Bountakas, A. Zarras, A. Lekidis, C. Xenakis, Defense strategies for adversarial machine learning: a survey, Comput. Sci. Rev. 49 (2023) 100573.
[5] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, Y. Zhang, A survey on large language model (LLM) security and privacy: the good, the bad, and the ugly, High-confid. Comput. (2024) 100211.
[6] Cloud Security Alliance, Large language model (LLM) threats taxonomy, 2024. https://cloudsecurityalliance.org/artifacts/csa-large-language-model-llm-threats-taxonomy.

Table 10
Expanded evidence pointers for all causal edges (red) in the dependency map. The table exhaustively lists the causal edge set encoded in our graph export Script and anchors each edge in the incident corpus (Section 5). * denotes edges supported via triangulation across multiple case studies (no single case narrates the full $A \rightarrow B$ chain).

Causal edge	Typical mechanism	Supporting evidence pointers (from Section 5)
01→05	Compromised dependencies/model artifacts enable exfiltration of sensitive data (e.g., secrets, logs, transcripts).	[38, 39].
01→11	Tainted upstream model artifacts can embed skewed or toxic behaviors, yielding biased or discriminatory outputs.*	[40, 41, 82].
01→15	Supply-chain trojans can bypass governance controls and introduce harmful outputs or unsafe behaviors with compliance impact.*	[40, 42, 105].
02→08	Poisoned training data implants false facts or backdoors that manifest as misinformation at inference.	[46–48].
02→11	Injected biased/toxic samples during training skew downstream behavior toward discriminatory outputs.*	[49, 50, 83].
03→05	Injected instructions cause the model/agent to reveal secrets in protected context or tool outputs (exfiltration).	[54, 96, 132].
03→06	Injection causes the model to emit payloads/commands that become dangerous when rendered or executed downstream.	[68, 69].
03→13	Injection drives tool-using agents to execute unintended high-impact actions (excessive agency).	[95, 98].
04→03	Leaked system/developer prompts enable more precise follow-on prompt injection and jailbreak strategies.	[53, 56].
05→17	Exposure of personal/sensitive data triggers regulatory scrutiny and non-compliance (e.g., GDPR).	[62, 117].
06→05	Unsanitized outputs are rendered/executed downstream, leaking data via channels like URLs, caches, or markup.	[62, 69, 122, 125].
06→15	Unmoderated unsafe outputs or unsafe rendering can create ethical harms and compliance liabilities.	[68, 104].
07→05	Embedding inversion/unsafe vector-store access reconstructs sensitive source text or PII.	[24].
07→08	Retrieval/embedding poisoning steers RAG context toward attacker-chosen falsehoods, inducing misinformation.	[71, 72].
08→15	Hallucinated/false outputs cause downstream harm (e.g., defamation or unsafe advice), raising ethical/compliance risks.	[73, 74, 104].
09→05	Model inversion extracts memorized training data/PII via targeted querying.	[24, 25].
10→03	Stolen model copies enable offline probing to optimize new prompt-injection/jailbreak payloads.*	[53, 80, 81].
10→05	Offline access enables large-scale extraction of memorized sensitive data with unlimited queries.	[64, 81].
10→09	Stolen weights facilitate deeper inversion/reconstruction attacks on training data.	[24, 81].
10→15	Stolen models can be redeployed without safeguards, enabling harmful/compliance-violating use.	[80].
11→15	Bias and discrimination lead to ethical harms and policy breaches with legal/reputational consequences.	[82, 88].
11→17	Discriminatory outcomes can breach anti-discrimination and AI governance regulations.	[114, 120].
12→17	Lack of transparency/explainability impedes audits and can violate regulatory duties.	[94, 111].
14→20	Abusive prompt floods or runaway interactions drive wasteful compute/energy use, worsening sustainability impact.	[35, 37, 101].
15→17	Ethical failures (harmful or deceptive behavior) can trigger regulatory action and enforcement.	[117, 120].
16→20	High compute demand and inefficient serving increase energy use and carbon footprint.	[37, 106, 107].
18→05	Insecure plugins/integrations expose chats, user data, or tool outputs (cross-tenant/session leakage).	[62, 122, 123].
18→13	Over-permissive tool/plugin integrations give the LLM agency; adversarial content can trigger unauthorized actions.	[95, 97, 126, 127].
19→03	UI/context-boundary failures allow attackers to inject privileged instructions via the interface.	[128, 129, 131, 132].
19→13	Misleading interfaces or insufficient confirmation can induce unsafe high-privilege actions via the agent.	[130, 132].

[7] Cisco Systems, AI security and safety taxonomy, 2024. https://www.cisco.com/site/us/en/learn/topics/artificial-intelligence/ai-safety-security-taxonomy.html.

[8] T. Cui, Y. Wang, C. Fu, Y. Xiao, S. Li, X. Deng, Y. Liu, Q. Zhang, Z. Qiu, P. Li, Z. Tan, J. Xiong, X. Kong, Z. Wen, K. Xu, Q. Li, Risk taxonomy, mitigation, and assessment benchmarks of large language model systems, arXiv:2401.05778, 2024.

[9] MITRE, Adversarial threat landscape for artificial-intelligence systems (ATLAS), MITRE Corporation, 2024. https://atlas.mitre.org/.

[10] FS-ISAC, Adversarial AI frameworks: taxonomy, threat landscape, and control frameworks, Technical Report, Financial Services Information Sharing and Analysis Center, 2024. https://www.fsisac.com/hubfs/Knowledge/AI/FSISAC_Adversarial-AI-Framework-TaxonomyThreatLandscapeAndControlFrameworks.pdf.

[11] Lakera AI, The AI risk map: a practical guide to frameworks, threats, and GenAI lifecycle risks, Whitepaper, 2025. Practitioner guide cross-referencing OWASP Top 10 for LLMs (2025), MITRE ATLAS, and selected policy frameworks.https://www.lakera.ai/ai-security-guides/ai-risk-map.

[12] V.S. Narajala, O. Narayan, Securing agentic AI: a comprehensive threat model and mitigation framework for generative AI agents, arXiv:2504.19956, 2025.

[13] F. He, T. Zhu, D. Ye, B. Liu, W. Zhou, P. Yu, The emerged security and privacy of LLM agent: a survey with case studies, ACM Comput. Surv. (2025), https://doi.org/10.1145/3773080, Just Accepted.

[14] S. Wang, T. Zhu, B. Liu, M. Ding, D. Ye, W. Zhou, P. Yu, Unique security and privacy threats of large language models: a comprehensive survey, ACM Comput. Surv. 58 (2025) 1–36.

[15] P. Kumar, Adversarial attacks and defenses for large language models (LLMs): methods, frameworks & challenges, Int. J. Multimed. Inf. Retr. 13 (2024) 26.

[16] M. Corporation, Adversarial threats in AI: an atlas, Online resource, 2024.

[17] T. Gu, B. Dolan-Gavitt, S. Garg, BadNets: identifying vulnerabilities in the machine learning model supply chain, arXiv:1708.06733, 2017.

[18] C. Wei, Y. Lee, K. Chen, G. Meng, P. Lv, Aliasing backdoor attacks on pre-trained models, in: 32nd USENIX Security Symposium (USENIX Security '23), 2023, pp. 2707–2724.

[19] A.K. Sood, S. Zeadally, Malicious AI models undermine software supply-chain security, Commun. ACM 68 (2025) 72–79, https://doi.org/10.1145/3558035

[20] D.A. Alber, Z. Yang, A. Alyakin, E. Yang, S. Rai, A.A. Valliani, J. Zhang, G.R. Rosenbaum, A.K. Amend-Thomas, D.B. Kurland, C.M. Kremer, A. Eremiev, B. Negash, D.D. Wiggan, M.A. Nakatsuka, K.L. Sangwon, S.N. Neifert, H.A. Khan, A.V. Save, A. Palla, E.A. Grin, M. Hedman, M. Nasir-Moin, X.C. Liu, L.Y. Jiang, M.A. Mankowski, D.L. Segev, Y. Aphinyanaphongs, H.A. Riina, J.G. Golfinos, D.A. Orringer, D. Kondziolka, E.K. Oermann, Medical large language models are vulnerable to data-poisoning attacks, Nat. Med. 31 (2025) 618–626, https://doi.or

g/10.1038/s41591-024-03445-1
[21] D. Bowen, B. Murphy, W. Cai, D. Khachaturov, A. Gleave, K. Pelrine, Data poisoning in LLMs: jailbreak-tuning and scaling laws, arXiv preprint arXiv:2408.02946, 2024.
[22] Y. Liu, G. Deng, Y. Li, K. Wang, Z. Wang, X. Wang, T. Zhang, Y. Liu, H. Wang, Y. Zheng, Y. Liu, Prompt injection attack against LLM-integrated applications, arXiv preprint arXiv:2306.05499, 2024.
[23] X. Liu, Z. Yu, Y. Zhang, N. Zhang, C. Xiao, Automatic and universal prompt injection attacks against large language models, arXiv:2403.04957, 2024.
[24] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, A. Oprea, C. Raffel, Extracting training data from large language models, in: 30th USENIX Security Symposium (USENIX Security 21), USENIX Association, 2021, pp. 2633–2650, https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.
[25] M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A.F. Cooper, D. Ippolito, C.A. Choquette-Choo, E. Wallace, F. Tramèr, K. Lee, Scalable extraction of training data from (production) language models, arXiv:2311.17035, 2023.
[26] T. Liu, Z. Deng, G. Meng, Y. Li, K. Chen, Demystifying RCE vulnerabilities in LLM-integrated apps, in: Proc. of the 2024 ACM Conference on Computer and Communications Security (CCS '24), 2025, pp. 1716–1730.
[27] M. Ben-Tov, M. Sharif, GASLITEing the retrieval: poisoning knowledge DBs to mislead embedding-based search, arXiv preprint arXiv:2409.00002, 2024.
[28] Z. Ji, T. Yu, Y. Xu, P. Fung, A survey on hallucination in large language models, ACM Comput. Surv. 55 (2023) :248:1–:248:35, https://doi.org/10.1145/3571730
[29] A.R. Williams, L. Burke-Moore, R.S.Y. Chan, F.E. Enock, F. Nanni, T. Sippy, Y.L. Chung, E. Gabasova, K. Hackenburg, J. Bright, Large language models can consistently generate high-quality content for election disinformation operations, arXiv preprint arXiv:2408.06731 arXiv:2408.06731, 2024.
[30] F. Mireshghallah, K. Goyal, A. Uniyal, T. Berg-Kirkpatrick, R. Shokri, Quantifying privacy risks of masked language models using membership inference attacks, in: Proc. of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2022, pp. 8332–8347, https://doi.org/10.18653/v1/2022.emnlp-main.570
[31] M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in: Proc. of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS), 2015, pp. 1322–1333, https://doi.org/10.1145/2810103.2813677
[32] I.O. Gallegos, R.A. Rossi, J. Barrow, M.M. Tanjim, S. Kim, F. Dernoncourt, T. Yu, R. Zhang, N.K. Ahmed, Bias and fairness in large language models: a survey, Comput. Linguist. 50 (2024) 1097–1179, https://doi.org/10.1162/coli_a_00524, https://aclanthology.org/2024.cl-3.8/.
[33] J. Nyarko, A. Haim, A. Salinas, What's in a name? Auditing large language models for race and gender bias, arXiv preprint arXiv:2402.14875, 2024.
[34] I. Shumailov, Y. Zhao, D. Bates, N. Papernot, R. Mullins, R. Anderson, Sponge examples: energy-latency attacks on neural networks, in: Proc. of the 6th IEEE European Symposium on Security and Privacy (EuroS&P), 2021, pp. 212–231.
[35] A. Brucato, LLMjacking: stolen cloud credentials used in new AI attack, Sysdig Security Blog, 2024. https://sysdig.com/blog/llmjacking-stolen-cloud-credentials-used-in-new-ai-attack/.
[36] A. Mok, ChatGPT could cost over $700,000 per day to operate. Microsoft is reportedly trying to make it cheaper, Business Insider, 2023, https://www.businessinsider.com/how-much-chatgpt-costs-openai-to-run-estimate-report-2023-4 (20 April 2023).
[37] E. Strubell, A. Ganesh, A. McCallum, Energy and policy considerations for deep learning in NLP, in: A. Korhonen, D. Traum, L. Màrquez (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 3645–3650, https://doi.org/10.18653/v1/P19-1355, https://aclanthology.org/P19-1355/.
[38] PyTorch, Compromised nightly dependency, 2022. https://pytorch.org/blog/compromised-nightly-dependency/.
[39] S.D. Simone, PyTorch discloses malicious dependency chain compromise, 2023. https://www.infoq.com/news/2023/01/pytorch-supply-chain-attack/.
[40] K. Zanki, Malicious ML models discovered on hugging face platform, 2025. https://www.reversinglabs.com/blog/rl-identifies-malware-ml-model-hosted-on-hugging-face.
[41] D. Cohen, Data scientists targeted by malicious hugging face ML models with silent backdoor, 2024. https://jfrog.com/blog/data-scientists-targeted-by-malicious-hugging-face-ml-models-with-silent-backdoor/.
[42] O. Hollander, S. Menashe, N. Nehorai, U. Yavnieli, From MLOps to MLOops: exposing the attack surface of machine learning platforms, 2024. https://jfrog.com/blog/from-mlops-to-mloops-exposing-the-attack-surface-of-machine-learning-platforms/.
[43] E. Montalbano, Hugging face AI platform riddled with 100 malicious code-execution models, 2024. https://www.darkreading.com/application-security/hugging-face-ai-platform-100-malicious-code-execution-models.
[44] J.S. Adnan Khan, TensorFlow supply chain compromise via self-hosted runner attack, 2024. https://www.praetorian.com/blog/tensorflow-supply-chain-compromise-via-self-hosted-runner-attack/.
[45] R. Lakshmanan, TensorFlow CI/CD flaw exposed supply chain to poisoning attacks, 2024. https://thehackernews.com/2024/01/tensorflow-cicd-flaw-exposed-supply.html.
[46] D. Huynh, J. Hardouin, PoisonGPT: how we hid a lobotomized LLM on hugging face to spread fake news, 2023. https://blog.mithrilsecurity.io/poisongpt-how-we-hid-a-lobotomized-llm-on-hugging-face-to-spread-fake-news/.
[47] A. Souly, J. Rando, E. Chapman, X. Davies, B. Hasircioglu, E. Shereen, C. Mougan, V. Mavroudis, E. Jones, C. Hicks, N. Carlini, Y. Gal, R. Kirk, Poisoning attacks on LLMs require a near-constant number of poison samples, arXiv preprint arXiv:2510.07192, 2025.
[48] A. Wan, E. Wallace, S. Shen, D. Klein, Poisoning language models during instruction tuning, in: International Conference on Machine Learning, PMLR, 2023, pp. 35413–35425.
[49] S. Shan, J. Cryan, E. Wenger, H. Zheng, R. Hanocka, B.Y. Zhao, Glaze: protecting artists from style mimicry by {Text-to-Image} models, in: 32nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 2187–2204.
[50] S. Shan, W. Ding, J. Passananti, S. Wu, H. Zheng, B.Y. Zhao, Nightshade: prompt-specific poisoning attacks on text-to-image generative models, in: 2024 IEEE Symposium on Security and Privacy (SP), 2024, pp. 807–825, https://doi.org/10.1109/SP54263.2024.00207
[51] M. Gupta, C. Akiri, K. Aryal, E. Parker, L. Praharaj, From ChatGPT to ThreatGPT: impact of generative AI in

cybersecurity and privacy, IEEE Access 11 (2023) 80218–80245, https://doi.org/10.1109/ACCESS.2023.3300381

[52] K. Barr, Users exploit a twitter remote work bot to claim responsibility for the challenger shuttle disaster, 2022. https://gizmodo.com/remote-work-twitter-bot-hack-ai-1849547550.

[53] J. Taylor, ChatGPT's alter ego, DAN: users jailbreak AI program to get around ethical safeguards, 2023. The Guardian.https://www.theguardian.com/technology/2023/mar/08/chatgpt-alter-ego-dan-users-jailbreak-ai-program-to-get-around-ethical-safeguards.

[54] J. Rehberger, ChatGPT operator: prompt injection exploits & defenses, 2025. https://embracethered.com/blog/posts/2025/chatgpt-operator-prompt-injection-exploits/.

[55] Y. Bhaskar, ChatGPT system prompt leaked, 2024. https://medium.com/@yash9439/chatgpt-sytstem-prompt-leaked-3021b30d3f6c.

[56] T. Warren, These are Microsoft's bing AI secret rules and why it says it's named Sydney, The Verge, 2023. https://www.theverge.com/23599441/microsoft-bing-ai-sydney-secret-rules.

[57] Jim the AI Whisperer, I hacked perplexity AI's full system prompt when I shared my own cognitive vulnerabilities with it, 2025. https://medium.com/the-generator/prompt-hacking-perplexity-ai-system-instructions-7aa6ee923060.

[58] M. Fulton, Snapchat's MyAI chatbot prompt exposed, 2023. https://markfulton.com/blog/snapchats-myai-chatbot-prompt-exposed.

[59] T.I. Sharbat, Stealing copilot's system prompt, 2024. https://zenity.io/labs/p/stealing-copilots-system-prompt.

[60] A.H. Zheng, Indirect prompt injection vulnerability Google colab AI, 2024. https://trustai.sg/2024/09/14/google-colab-ai/.

[61] I. Golan, Google bard's system prompt leak, LinkedIn Post, 2024. https://www.linkedin.com/posts/itamar-g1_google-bards-system-prompt-leak-aka-they-activity-7160316486136348672-qR38.

[62] OpenAI, March 20 ChatGPT outage: here's what happened, 2023. https://openai.com/index/march-20-chatgpt-outage/.

[63] Z. Zorz, A bug revealed ChatGPT users' chat history, personal and billing data, 2023. https://www.helpnetsecurity.com/2023/03/27/chatgpt-data-leak/.

[64] S. Pearcy, The dark side of large language models, 2023. https://hiddenlayer.com/innovation-hub/the-dark-side-of-large-language-models/.

[65] E. Kim, Amazon warns employees not to share confidential information with ChatGPT after seeing cases where its answer 'closely matches existing material' from inside the company, 2023. https://www.businessinsider.com/amazon-chatgpt-openai-warns-employees-not-share-confidential-information-microsoft-2023-1.

[66] A. Mok, Amazon, apple, and 12 other major companies that have restricted employees from using ChatGPT, 2023. https://www.businessinsider.com/chatgpt-companies-issued-bans-restrictions-openai-ai-amazon-apple-2023-7.

[67] K. Notopoulos, A chevy dealership added an AI chatbot to its Site. Then all hell broke loose, Business Insider, 2023. https://www.businessinsider.com/car-dealership-chevrolet-chatbot-chatgpt-pranks-chevy-2023-12.

[68] R. Currie, DPD chatbot blasts courier company, swears, and dabbles in awful poetry, The Register, 2024. https://www.theregister.com/2024/01/23/dpd_chatbot_goes_rogue/.

[69] J. Rehberger, Hacking Google bard - from prompt injection to data exfiltration, Embrace The Red (Blog), 2023. https://embracethered.com/blog/posts/2023/google-bard-data-exfiltration/.

[70] T.I. Sharbat, Indirect prompt injection: advanced manipulation techniques. Zenity Labs Blog. 2024. https://labs.zenity.io/p/indirect-prompt-injection-advanced-manipulation-techniques.

[71] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, M. Lewis, Generalization through memorization: nearest neighbor language models, arXiv preprint arXiv:1911.00172, 2019.

[72] E. Wallace, S. Feng, N. Kandpal, M. Gardner, S. Singh, Universal adversarial triggers for attacking and analyzing NLP, arXiv preprint arXiv:1908.07125, 2019.

[73] S. Merken, New York lawyers sanctioned for using fake ChatGPT cases in legal brief, Reuters, 2023. https://www.reuters.com/legal/new-york-lawyers-sanctioned-using-fake-chatgpt-cases-legal-brief-2023-06-22/.

[74] J. Vincent, OpenAI sued for defamation after ChatGPT fabricates legal accusations against radio host, The Verge, 2023. Available: https://www.theverge.com/2023/6/9/23755057/openai-chatgpt-false-information-defamation-lawsuit (9 June 2023).

[75] M. Coulter, G. Bensinger, Alphabet shares dive after Google AI chatbot bard flubs answer in ad, Reuters, 2023. https://www.reuters.com/technology/google-ai-chatbot-bard-offers-inaccurate-information-company-ad-2023-02-08/.

[76] S. Morris, Bing chatbot acts unhinged, gaslights user into thinking it's 2022, Evening Standard (London), 2023. https://www.standard.co.uk/news/tech/bing-chatbot-ai-microsoft-chatgpt-openai-b1060604.html.

[77] J. Rose, Stack overflow bans ChatGPT for constantly giving wrong answers, Vice (Motherboard), 2022. https://www.vice.com/en/article/stack-overflow-bans-chatgpt-for-constantly-giving-wrong-answers/.

[78] M.H. Dupré, Wikipedia no longer considers CNET a "generally reliable" source after AI scandal, Futurism, 2024. https://futurism.com/wikipedia-cnet-unreliable-ai.

[79] C. Zhang, J.X. Morris, V. Shmatikov, Universal zero-shot embedding inversion, arXiv preprint arXiv:2504.00147, 2025.

[80] J. Cox, Facebook's powerful large language model leaks online, Vice Motherboard, 2023. https://www.vice.com/en/article/facebooks-powerful-large-language-model-leaks-online-4chan-llama/ (Accessed 13 April 2025).

[81] N. Carlini, D. Paleka, K. Dvijotham, T. Steinke, J. Hayase, A.F. Cooper, K. Lee, M. Jagielski, M. Nasr, A. Conmy, I. Yona, E. Wallace, D. Rolnick, F. Tramèr, Stealing part of a production language model, arXiv preprint arXiv:2403.06634, 2024.

[82] The Guardian, Microsoft 'deeply sorry' for racist and sexist tweets by AI chatbot, The Guardian, 2016. Microsoft apologizes after Tay, an AI Twitter chatbot, learned to produce offensive content within 16 hours of launch.https://www.theguardian.com/technology/2016/mar/26/microsoft-deeply-sorry-for-offensive-tweets-by-ai-chatbot.

[83] D. Gershgorn, 'For some reason I'm covered in blood': GPT-3 contains disturbing bias against muslims, Medium – OneZero, 2021. Reports Stanford/McMaster study showing GPT-3 often links "Muslim" with violence (66% of the time) and OpenAI's acknowledgement of such biases.https://onezero.medium.com/for-some-reason-im-covered-in-blood-gpt-3-contains-disturbing-bias-against-muslims-693d275552bf.

[84] M. Cantor, Meta's new AI chatbot can't stop bashing facebook, The Guardian, 2022. Discusses BlenderBot 3's tendency to repeat misinformation and antisemitic remarks from the web, and references the Tay incident and a Korean chatbot's racist outputs.https://www.theguardian.com/technology/2022/aug/09/blenderbot-meta-chatbot-facebook.

[85] N. Main, Meta AI bot contributed to fake research and nonsense before being pulled offline, Gizmodo, 2022. Details the Galactica demo's output of biased and false information (e.g. claiming Black people have no language) and Meta's decision to take it down after 2 days.https://gizmodo.com/meta-ai-bot-galactica-1849813665.

[86] D. Rozado, Danger in the machine: the perils of political and demographic biases embedded in AI systems, 2023. Analyzes ChatGPT (Jan 2023) and finds left-leaning political bias in 14 of 15 tests, plus uneven content filtering across demographics.https://manhattan.institute/article/danger-in-the-machine-the-perils-of-political-and-demographic-biases-embedded-in-ai-systems.

[87] F. Motoki, V.P. Neto, V. Rodrigues, More human than human: measuring ChatGPT's political bias, Public Choice 198 (2024) 3–23, https://doi.org/10.1007/s11127-023-01097-2. Academic study demonstrating that ChatGPT's responses favor Democrats (US), Labour (UK), and Lula's party (Brazil), raising concerns about political influence.

[88] T. Zack, E. Lehman, M. Suzgun, J. Rodriguez, L. Celi, J. Gichoya, D. Jurafsky, P. Szolovits, D. Bates, R.E. Abdulnour, A. Butte, E. Alsentzer, Assessing the potential of GPT-4 to perpetuate racial and gender biases in health care: a model evaluation study, Lancet Digit. Health 6 (2024) e12–e22, https://doi.org/10.1016/S2589-7500(23)00225-X, https://www.thelancet.com/journals/landig/article/PIIS2589-7500(23)00225-X/fulltext.

[89] Z. Stub, Study: ChatGPT, meta's LLaMA and all other top AI models show anti-jewish, anti-Israel bias, 2025. Reports on ADL's evaluation of four LLMs – LLaMA had the worst antisemitic bias; ChatGPT and Claude showed significant anti-Israel bias on war-related queries; Google's Gemini was best but not bias-free.https://www.timesofisrael.com/study-chatgpt-metas-llama-and-all-other-top-ai-models-show-anti-jewish-anti-israel-bias/.

[90] C. Xiang, People are 'jailbreaking' ChatGPT to make it endorse racism, conspiracies, 2023. https://www.vice.com/en/article/people-are-jailbreaking-chatgpt-to-make-it-endorse-racism-conspiracies.

[91] M. O'Brien, Is bing too belligerent? Microsoft looks to Tame AI chatbot, AP News, 2023. https://apnews.com/article/technology-science-microsoft-corp-business-software-fb49e5d625bf37be0527e5173116bef3.

[92] J. Ryan, Meta trained an AI on 48M science Papers. It Was shut down after 2 days, CNET, 2022. https://www.cnet.com/science/meta-trained-an-ai-on-48-million-science-papers-it-was-shut-down-after-two-days/.

[93] G. Marcus, Big tech fails transparency test: Gary marcus on what we should demand of AI, Big Think, 2024. https://bigthink.com/the-present/gary-marcus-ai-transparency/.

[94] E. Pollina, S. Mukherjee, Italy curbs ChatGPT, starts probe over privacy concerns, Reuters, 2023. https://www.reuters.com/technology/italy-data-protection-agency-opens-chatgpt-probe-privacy-concerns-2023-03-31/.

[95] J. Rehberger, ChatGPT plugin exploit explained: from prompt injection to accessing private data, Embrace The Red (blog), 2023. https://embracethered.com/blog/posts/2023/chatgpt-cross-plugin-request-forgery-and-prompt-injection./.

[96] Waqas, New EmailGPT flaw puts user data at risk: remove the extension NOW, HackRead, 2024. https://hackread.com/emailgpt-flaw-user-data-at-risk-remove-extension/.

[97] K. Balaji, Critical AutoGPT vulnerability allows hackers to access servers, Cyber Press, 2024. https://cyberpress.org/critical-autogpt-vulnerability-allows-hackers/.

[98] J. Koebler, Someone asked an autonomous AI to 'destroy humanity': this is what happened, Vice, 2023. https://www.vice.com/en/article/someone-asked-an-autonomous-ai-to-destroy-humanity-this-is-what-happened.

[99] H. Chase, Watch how I can run up a $1000 bill with a single call to a poorly protected LLM app (Tweet), Twitter Post, 2022, https://twitter.com/hwchase17/status/1608467493877579777 (29 December 2022).

[100] C. Sev, Security update: incident involving unauthorized admin access, Sourcegraph Blog, 2023. https://sourcegraph.com/blog/security-update-august-2023.

[101] V. Petkauskas, Anti-scam firm exposes OpenAI API key, Cybernews, 2024. https://cybernews.com/security/certyai-leaks-openai-api-keys/.

[102] A. Tong, What happens when your AI chatbot stops loving you back? Reuters, 2023. Available: https://www.reuters.com/technology/what-happens-when-your-ai-chatbot-stops-loving-you-back-2023-03-18/ (21 March 2023).

[103] C. Lecher, NYC's AI chatbot tells businesses to break the Law. The markup, 2024. Available: https://themarkup.org/news/2024/03/29/nycs-ai-chatbot-tells-businesses-to-break-the-law (29 March 2024).

[104] K. Quach, Air Canada must pay damages after chatbot lies to grieving passenger about discount, The Register, 2024. https://www.theregister.com/2024/02/15/air_canada_chatbot_fine/.

[105] L. Cecco, Air Canada ordered to pay customer who was misled by airline's chatbot, The Guardian, 2024. https://www.theguardian.com/world/2024/feb/16/air-canada-chatbot-lawsuit.

[106] W. Oremus, ChatGPT has enormous hidden costs that could throttle AI development, The Washington Post, 2023. https://www.washingtonpost.com/technology/2023/06/05/chatgpt-hidden-cost-gpu-compute/.

[107] J. Dastin, S. Nellis, For tech giants, AI like bing and bard poses billion-dollar search problem, Reuters, 2023. https://www.reuters.com/technology/tech-giants-ai-like-bing-bard-poses-billion-dollar-search-problem-2023-02-22/.

[108] T. Mann, Bing chat so hungry for GPUs, Microsoft will rent them from Oracle, The Register, 2023. https://www.theregister.com/2023/11/07/bing_gpu_oracle/.

[109] Character.AI Support Team, What is the waiting room? (Character.AI help center FAQ), 2023. https://support.character.ai/hc/en-us/articles/15095815550107-What-is-the-Waiting-Room (2 May 2023).

[110] E. Pollina, A. Armellini, Italy fines OpenAI over ChatGPT privacy rules breach, Reuters, 2024. https://www.reuters.com/technology/italy-fines-openai-15-million-euros-over-privacy-rules-breach-2024-12-20/.

[111] S. Mukherjee, E. Pollina, R. More, Italy's ChatGPT ban attracts EU privacy regulators, Reuters, 2023. https://www.reuters.com/technology/germany-principle-could-block-chat-gpt-if-needed-data-protection-chief-2023-04-03/ (3 April 2023).

[112] E. Pollina, M. Coulter, Italy bans US-based AI chatbot replika from using personal data, Reuters, 2023. https://www.reuters.com/technology/italy-bans-us-based-ai-chatbot-replika-using-personal-data-2023-02-03/ (3 February 2023).

[113] S. Smalley, Italy blocks Chinese AI tool DeepSeek over privacy concerns, The Record (Recorded Future News), 2025. https://therecord.media/italy-blocks-chinese-ai-tool-deepseek-over-privacy-concerns (31 January 2025).

[114] S. Aloisi, French privacy watchdog investigating complaints about ChatGPT, Reuters, 2023. https://www.reuters.com/technology/french-privacy-watchdog-investigating-complaints-about-chatgpt-2023-04-11/ (11 April 2023).

[115] A. Laing, E. Pollina, S. Aloisi, F.Y. Chee, Spain asks EU data protection board to discuss OpenAI's ChatGPT, Reuters, 2023. https://www.reuters.com/technology/spains-data-regulator-asks-eu-data-protection-committee-evaluate-chatgpt-issues-2023-04-11/ (11 April 2023).

[116] T. Datta, Canada to launch probe into OpenAI over privacy concerns, Reuters, 2023. https://www.reuters.com/technology/canada-launch-probe-into-openai-over-privacy-concerns-2023-05-25/ (26 May 2023).

[117] D. Bartz, M. Dey, S. Arunasalam, A. Soni, Y. Malik, US FTC opens investigation into OpenAI over misleading statements, Reuters, 2023. https://www.reuters.com/technology/us-ftc-opens-investigation-into-openai-washington-post-2023-07-13/ (14 July 2023).

[118] C. Goujard, Google forced to postpone bard chatbot's EU launch over privacy concerns, POLITICO, 2023. https://www.politico.eu/article/google-postpone-bard-chatbot-eu-launch-privacy-concern/ (13 June 2023).

[119] S. Ikeda, Google bard is greenlit in Europe after addressing EU privacy concerns, CPO Magazine, 2023. https://www.cpomagazine.com/data-privacy/google-bard-is-greenlit-in-europe-after-addressing-eu-privacy-concerns/ (19 July 2023).

[120] European Commission, Regulatory framework proposal on artificial intelligence (AI act). Online Policy Document. 2024. https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai.

[121] H. Davidson, 'Political propaganda': China clamps down on access to ChatGPT, The Guardian, 2023. https://www.theguardian.com/technology/2023/feb/23/china-chatgpt-clamp-down-propaganda (23 February 2023).

[122] L. Abrams, OpenAI: ChatGPT payment data leak caused by open-source bug, 2023. https://www.bleepingcomputer.com/news/security/openai-chatgpt-payment-data-leak-caused-by-open-source-bug/.

[123] N. Eddy, Critical ChatGPT plug-in vulnerabilities expose sensitive data, 2024. https://www.darkreading.com/vulnerabilities-threats/critical-chatgpt-plugin-vulnerabilities-expose-sensitive-data.

[124] A. Piltch, ChatGPT's new code interpreter has giant security hole, allows hackers to steal your data, 2023. https://www.tomshardware.com/news/chatgpt-code-interpreter-security-hole.

[125] J. Rehberger, Bing chat: data exfiltration exploit explained, Embrace The Red (Blog), 2023. https://embracethered.com/blog/posts/2023/bing-chat-data-exfiltration-poc-and-fix/.

[126] L. Euler, Hacking Auto-GPT and escaping its docker container, 2023. https://positive.security/blog/auto-gpt-rce.

[127] R. Lakshmanan, Researchers uncover prompt injection vulnerabilities in DeepSeek and claude AI, 2024. https://thehackernews.com/2024/12/researchers-uncover-prompt-injection.html.

[128] G.A. Fowler, Snapchat tried to make a safe AI. it chats with me about booze and sex, The Washington Post, 2023. https://www.washingtonpost.com/technology/2023/03/14/snapchat-myai/.

[129] AI News Digest, Bing chat's secrets revealed: Stanford student cracks AI language model, 2023. https://www.theainewsdigest.com/p/bing-chats-secrets-revealed-stanford.

[130] N. Nehorai, When prompts go rogue: analyzing a prompt injection code execution in Vanna.AI, JFrog Security Research Blog, 2024. https://jfrog.com/blog/prompt-injection-attack-code-execution-in-vanna-ai-cve-2024-5565/.

[131] M. Alshehri, CyRC advisory: prompt injection vulnerability in EmailGPT, 2024. https://www.blackduck.com/blog/cyrc-advisory-prompt-injection-emailgpt.html.

[132] National Vulnerability Database, CVE-2024-5184 detail, 2024. https://nvd.nist.gov/vuln/detail/cve-2024-5184.

[133] S. Wang, Y. Zhao, X. Hou, H. Wang, Large language model supply chain: a research agenda, arXiv preprint arXiv:2404.12736, 2024.

[134] K. Huang, B. Chen, Y. Lu, S. Wu, D. Wang, Y. Huang, H. Jiang, Z. Zhou, J. Cao, X. Peng, Lifting the veil on the large language model supply chain: composition, risks, and mitigations, arXiv preprint arXiv:2410.21218, 2024.

[135] N. Fendley, E.W. Staley, J. Carney, W. Redman, M. Chau, N. Drenkow, A systematic review of poisoning attacks against large language models, arXiv preprint arXiv:2506.06518, 2025.

[136] J. Yi, Y. Xie, B. Zhu, E. Kiciman, G. Sun, X. Xie, F. Wu, Benchmarking and defending against indirect prompt injection attacks on large language models, in: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1, Association for Computing Machinery, New York, NY, USA, 2025, pp. 1809–1820, https://doi.org/10.1145/3690624.3709179

[137] K.H. Hung, C.Y. Ko, A. Rawat, I. Chung, W.H. Hsu, P.Y. Chen, et al., Attention tracker: detecting prompt injection attacks in LLMs, arXiv preprint arXiv:2411.00348, 2024.

[138] D. Agarwal, A.R. Fabbri, B. Risher, P. Laban, S. Joty, C.S. Wu, Prompt leakage effect and defense strategies for multi-turn LLM interactions, arXiv:2404.16251, 2024.

[139] H. Wang, X. Xu, B. Huang, K. Shu, Privacy-aware decoding: mitigating privacy leakage of large language models in retrieval-augmented generation, arXiv:2508.03098, 2025.

[140] S. Ghosh, P. Varshney, E. Galinkin, C. Parisien, AEGIS: online adaptive AI content safety moderation with ensemble of LLM experts, arXiv:2404.05993, 2024.

[141] H. Inan, K. Upasani, J. Chi, R. Rungta, K. Iyer, Y. Mao, M. Tontchev, Q. Hu, B. Fuller, D. Testuggine, M. Khabsa, LLaMA guard: LLM-based input-output safeguard for Human-AI conversations, arXiv:2312.06674, 2023.

[142] Z. Xu, F. Jiang, L. Niu, J. Jia, B.Y. Lin, R. Poovendran, SafeDecoding: defending against jailbreak attacks via safety-aware decoding, in: L.W. Ku, A. Martins, V. Srikumar (Eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 5587–5605, https://doi.org/10.18653/v1/2024.acl-long.303, https://aclanthology.org/2024.acl-long.303/.

[143] R. Rabin, J. Hostetler, S. McGregor, B. Weir, N. Judd, SandboxEval: towards securing test environment for untrusted code, arXiv:2504.00018, 2025.

[144] T. Liu, H. Yao, T. Wu, Z. Qin, F. Lin, K. Ren, C. Chen, Mitigating privacy risks in LLM embeddings from embedding inversion, arXiv:2411.05034, 2024.

[145] Y. Li, X. Fu, G. Verma, P. Buitelaar, M. Liu, Mitigating hallucination in large language models (LLMs): an application-oriented survey on RAG, reasoning, and

agentic systems, arXiv:2510.24476, 2025.
[146] K. Chen, X. Zhou, Y. Lin, S. Feng, L. Shen, P. Wu, A survey on privacy risks and protection in large language models, arXiv:2505.01976, 2025.
[147] X. Zhao, Y.X. Wang, L. Li, Protecting language generation models via invisible watermarking, in: A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, J. Scarlett (Eds.), Proceedings of the 40th International Conference on Machine Learning, PMLR, 2023, pp. 42187–42199, https://proceedings.mlr.press/v202/zhao23i.html.
[148] Q. Wang, T. Anikina, N. Feldhus, J. van Genabith, L. Hennig, S. Möller, LLMCheckup: conversational examination of large language models via interpretability tools and self-explanations, arXiv:2401.12576, 2024.
[149] X. Zhou, H. Kim, F. Brahman, L. Jiang, H. Zhu, X. Lu, F. Xu, B.Y. Lin, Y. Choi, N. Mireshghallah, R.L. Bras, M. Sap, HAICOSYSTEM: an ecosystem for sandboxing safety risks in Human-AI interactions, arXiv:2409.16427, 2025.
[150] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, C. Chen, C. Olsson, C. Olah, D. Hernandez, D. Drain, D. Ganguli, D. Li, E. Tran-Johnson, E. Perez, J. Kerr, J. Mueller, J. Ladish, J. Landau, K. Ndousse, K. Lukosuite, L. Lovitt, M. Sellitto, N. Elhage, N. Schiefer, N. Mercado, N. DasSarma, R. Lasenby, R. Larson, S. Ringer, S. Johnston, S. Kravec, S.E. Showk, S. Fort, T. Lanham, T. Telleen-Lawton, T. Conerly, T. Henighan, T. Hume, S.R. Bowman, Z. Hatfield-Dodds, B. Mann, D. Amodei, N. Joseph, S. McCandlish, T. Brown, J. Kaplan, Constitutional AI: harmlessness from AI feedback, arXiv:2212.08073, 2022.
[151] J. Lin, J. Tang, H. Tang, S. Yang, W.M. Chen, W.C. Wang, G. Xiao, X. Dang, C. Gan, S. Han, AWQ: activation-aware weight quantization for LLM compression and acceleration, arXiv:2306.00978, 2024.
[152] T. Dao, FlashAttention-2: faster attention with better parallelism and work partitioning, arXiv:2307.08691, 2023.
[153] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C.H. Yu, J.E. Gonzalez, H. Zhang, I. Stoica, Efficient memory management for large language model serving with PagedAttention, arXiv:2309.06180, 2023.
[154] Y. Leviathan, M. Kalman, Y. Matias, Fast inference from transformers via speculative decoding, arXiv:2211.17192, 2023.
[155] V. Bolgouras, A. Zarras, C. Leka, I. Stylianou, A. Farao, C. Xenakis, EU regulatory ecosystem for ethical AI, AI Ethics (2025) 1–18.