Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Four-Dimensional Empirical Analysis of Security Vulnerabilities in LLM-Based Systems

Ryan Jay Wong

780195841@qq.com

UCSI University

Research Article

Keywords: LLM security; multimodal security; agent security; attack case analysis; jailbreak; prompt injection; memory poisoning; tool-use security

Posted Date: November 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11806818/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

A Four-Dimensional Empirical Analysis of Security Vulnerabilities in LLM-Based Systems

Ryan Jay Wong
UCSI Researcher, Kuala Lumpur, Malaysia

Abstract

The growing integration of large language models (LLMs) with multimodal perception, retrieval, persistent memory, and external tools has expanded the attack surface of modern AI systems. Existing studies report vulnerabilities ranging from prompt-based jailbreaks and visual instruction injection to retrieval poisoning, memory manipulation, and unauthorized tool use. However, these findings are reported under heterogeneous taxonomies, threat assumptions, and evaluation protocols, limiting direct comparison of their underlying mechanisms and security consequences. This paper presents a four-dimensional analytical framework for examining security vulnerabilities in LLM-based systems. The framework characterizes each documented attack in terms of its attack surface (AS), exploitation mechanism (EM), security constraint failure (CF), and security impact (SI). We apply this framework to a structured comparative analysis of representative primary studies and publicly available security benchmarks covering conventional LLMs, multimodal LLMs (MLLMs), and agent-based systems. For quantitative evaluation, we construct the Agent Attack Evaluation Benchmark (AAEB) and evaluate 13 LLMs across five agent attack families. We further examine the publicly available evaluation resources AgentHarm and JailbreakBench to characterize their task coverage and safety evaluation settings. The analysis distinguishes experimental results obtained from AAEB from benchmark statistics reported by existing resources. It identifies substantial variation in attack effectiveness across agent attack families, limitations in automated safety judging, and paradigm-specific security boundaries introduced by multimodal perception, persistent state, retrieval, and tool execution.

Keywords: LLM security; multimodal security; agent security; attack case analysis; jailbreak; prompt injection; memory poisoning; tool-use security

1 Introduction

Large language models (LLMs) are increasingly deployed as components of applications for natural language understanding, content generation, retrieval, programming assistance, and decision support. Their capabilities are shaped by large-scale pretraining and subsequent instruction tuning and alignment procedures [1, 2, 3]. More recently, LLMs have been integrated with visual and auditory encoders, retrieval systems, persistent memory, external tools, and planning loops. These developments have produced multimodal large language models (MLLMs) and agent-based architectures that can interpret heterogeneous inputs and perform multi-step tasks in external environments[4, 5].

The expansion of capabilities also changes the security problem. A conventional chatbot primarily receives user-provided text and returns generated content [6, 7]. An MLLM additionally processes images, audio, video, and structured multimodal representations. An agent may retrieve documents, update memory, call APIs, execute code, and pass information to other agents [8, 9, 10]. Malicious content can therefore enter through prompts, retrieved webpages, images, audio signals, memory records, tool responses, or inter-agent messages. Once incorporated into

a model's context or execution state, such content may influence subsequent interpretation and actions.

Prior work has demonstrated that alignment does not provide a complete security boundary. Prompt-based attacks exploit role-playing, intent concealment, adversarial suffixes, and automated prompt optimization to elicit policy-violating outputs from aligned models [11, 12, 13, 14, 15, 16]. Multimodal attacks can embed malicious instructions in images or exploit interactions between modalities [17, 18, 19, 20, 21]. Agent-based attacks extend these risks to retrieval, memory, reasoning, tool execution, and multi-agent communication [22, 23, 24, 25, 26, 27, 28].

Although the literature contains many attack demonstrations and benchmark results, the reported cases are described using heterogeneous terminology. An attack name may refer to an input channel, an adversarial strategy, a compromised component, or an observed consequence. Furthermore, studies differ in target models, access assumptions, attack budgets, success criteria, and evaluation protocols. These differences make direct numerical comparison inappropriate unless the original settings are preserved. A case-based analysis can nevertheless provide a consistent way to compare the structural characteristics of reported attacks.

This paper therefore investigates publicly documented attack cases using a four-dimensional framework: (i) *attack surface* (AS), the component or interaction channel through which adversarial content enters; (ii) *exploitation mechanism* (EM), the strategy by which the content or component is manipulated; (iii) *security constraint failure* (CF), the security property or control that is violated or weakened; and (iv) *security impact* (SI), the resulting consequence. The four dimensions are treated as coding categories for empirical case analysis, rather than as a claim that every attack follows a single causal sequence.

The study is guided by three research questions (RQs):

- **RQ1:** What attack families and attack-effectiveness patterns are observed in the proposed AAEB benchmark, and what complementary evaluation characteristics are reported by existing security resources?
- **RQ2:** How can reported attack procedures, violated security constraints, and documented impacts be consistently characterized using the four-dimensional framework?
- **RQ3:** What security and evaluation differences emerge across conventional LLMs, MLLMs, and agent-based systems, particularly with respect to multimodal inputs, persistent memory, retrieval, and external actions?

The contributions of this paper are threefold:

1. We develop Four-Dimensional Security Case Analysis (FDSCA), a source-grounded analytical framework that explicitly separates attack surfaces, exploitation mechanisms, security constraint failures, and security impacts. This separation reduces the ambiguity that arises when attack names conflate entry points, adversarial operations, violated properties, and outcomes.
2. We operationalize FDSCA through explicit characterization definitions and evidence-tracing rules, and use it to examine representative documented attacks across conventional LLMs, MLLMs, and agent-based systems. The analysis preserves the original threat assumptions and does not infer undocumented attack properties.
3. We construct AAEB and conduct experiments to evaluate five agent attack families across 13 LLMs. We further examine publicly available evaluation resources, including AgentHarm and JailbreakBench, to characterize benchmark coverage and automated safety evaluation. These findings are connected to a cross-paradigm analysis of security boundaries and evaluation gaps.

The remainder of this paper is organized as follows. Section 2 describes the evidence sources and extraction procedure. Section 3 introduces the system settings and threat model. Section 4 defines the four-dimensional analytical framework, and Section 5 operationalizes its coding rules. Sections 6.1–6.3 analyze documented attacks across the three system paradigms. Section 7 compares the resulting security patterns, and Section 8 presents quantitative results from the proposed AAEB benchmark and examines complementary evidence from existing security evaluation resources. The remaining sections discuss defensive implications, evaluation resources, validity considerations, and conclusions.

2 Evidence Sources and Empirical Study Design

This study adopts a source-grounded empirical synthesis of publicly documented attacks and benchmark evaluations. The evidence unit is a reported attack configuration or benchmark measurement, rather than an independently collected real-world incident. This distinction is important because published demonstrations are produced under different models, system prompts, attacker capabilities, tool configurations, and success criteria. The study therefore retains source-specific evaluation conditions and does not pool attack-success rates across incompatible benchmarks.

The evidence base consists of primary research papers and public benchmark reports covering three system paradigms: conventional LLMs, MLLMs, and agent-based systems. Representative sources are selected when they describe a concrete attack procedure, identify the affected system component, or report a quantitative security evaluation. Survey papers are used to locate relevant primary studies and provide context, but benchmark statistics are attributed to the original benchmark publications rather than counted as independent observations.

For each reported attack or benchmark, the analysis records the source, system paradigm, target model or system, input or interaction surface, exploitation procedure, security property or control at risk, documented impact, benchmark name, success definition, and reported metric. When a source does not provide sufficient evidence for one of the four dimensions, that dimension is described as unspecified rather than inferred. The four-dimensional mapping is used to organize and compare source evidence; it is not presented as the output of a newly conducted independent annotation exercise.

The quantitative component consists of three complementary evaluation resources. First, we construct the Agent Attack Evaluation Benchmark (AAEB) to evaluate five agent attack families across 13 LLMs. The corresponding attack-success results are obtained from our own experiments under the experimental settings defined in this study. Second, AgentHarm provides a documented task set containing 110 explicitly malicious agent tasks, expanded to 440 instances through augmentations, across 11 harm categories. Third, JailbreakBench provides a 300-example judge-comparison dataset containing annotations from three human annotators and four automated judges, namely HarmBench, GPT-4, Llama Guard 2, and Llama-3-70B. The majority vote of the human annotators serves as the reference label for examining automated evaluation consistency.

The analysis proceeds in three stages. First, descriptive comparisons summarize benchmark scale, attack-family coverage, and reported attack-success rates. Second, the four-dimensional framework is applied qualitatively to representative attack procedures to distinguish their entry surfaces, exploitation mechanisms, affected constraints, and documented impacts. Third, the findings are compared across system paradigms to identify additional trust boundaries introduced by multimodal perception, retrieval, persistent memory, tool use, and inter-agent communication.

Because the source-derived case-level coding records are not uniformly available in a common format, this study does not report the size of a newly constructed literature-derived attack corpus, a duplicate-removal count, Cohen's kappa, a four-dimensional exact-match rate, a chi-square

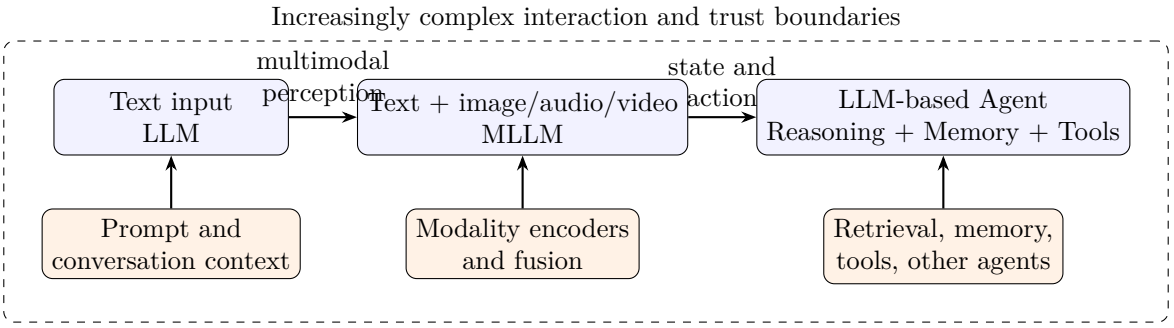


Figure 1: A high-level view of the evolution from text-oriented LLMs to multimodal and agent-based systems. The categories are analytical abstractions: practical systems may combine all three.

statistic, or Cramer's V. Such statistics would require a predefined case-level dataset and, for inter-annotator reliability, independently coded labels. Instead, quantitative claims are restricted to experimental measurements obtained from AAEB, values reported by the cited benchmark sources, and arithmetic summaries that can be directly verified from these results.

3 Background and Threat Model

3.1 Evolution of LLM-based systems

A conventional LLM receives a token sequence and generates a continuation conditioned on its parameters and context. Safety behavior is commonly shaped through instruction tuning, preference optimization, and policy-oriented training. The model may be surrounded by system prompts, moderation filters, and application logic, but the central inference operation remains text-conditioned generation.

An MLLM extends this interface by encoding one or more non-text modalities and integrating their representations with language tokens. Depending on the architecture, modality encoders may be connected to the language model through projection layers, resamplers, cross-attention, or other fusion mechanisms. This introduces additional preprocessing and representation pathways. The security boundary therefore includes modality-specific encoders, preprocessing pipelines, fusion operations, and the language-generation component.

An agent-based system adds an execution loop. A representative abstraction is

$$o_t \rightarrow r_t \rightarrow a_t \rightarrow e_t \rightarrow s_{t+1}, \tag{1}$$

where o_t is an observation, r_t is a reasoning or planning state, a_t is a selected action, e_t is the execution result, and s_{t+1} is the updated state. The loop may include retrieval, memory read/write operations, tool invocation, and communication with other agents. Security consequently depends on both model behavior and the controls governing state transitions and external actions.

3.2 Security alignment and constraint enforcement

Alignment seeks to make model behavior consistent with human preferences, intended use, and safety policies. RLHF is one influential approach in which preference comparisons are used to optimize a model toward desired responses [2]. Related approaches include supervised safety fine-tuning, preference optimization, constitutional or principle-guided training, and inference-time

safeguards. These approaches can improve refusal behavior, but they should not be equated with formal enforcement.

For the purposes of this study, a *security constraint* is a requirement imposed on a system's behavior or operation. Examples include refusing disallowed assistance, protecting confidential information, preserving instruction authority, preventing unauthorized tool execution, and maintaining the integrity of persistent memory. Constraints may be represented in model training, system instructions, application code, access-control policies, or combinations of these mechanisms.

Three distinctions are important. First, a model's learned tendency to refuse is not equivalent to an external authorization check. Second, a system prompt is not a cryptographic or operating-system security boundary. Third, a model's explanation that it followed a policy does not establish that the policy was actually enforced. Robust deployments should therefore combine model-level alignment with independently enforced controls, especially when external actions or sensitive data are involved.

3.3 Threat model

We consider an adversary who can supply or influence one or more inputs processed by an LLM-based system. Depending on the deployment, the adversary may control a user prompt, publish content that may later be retrieved, provide an image or audio sample, influence a memory write, or interact with an agent through a tool or communication interface. We do not assume that the adversary can directly modify model weights or compromise the host operating system unless a particular study explicitly adopts such a capability.

The adversary may seek to elicit disallowed or policy-violating content, extract private or system-level information, cause the system to follow untrusted instructions, induce unauthorized tool calls or external side effects, corrupt persistent memory or retrieved knowledge, disrupt availability, integrity, or task completion, or propagate malicious information across agents and system components.

The adversary's capabilities are deployment-dependent. A direct-prompt attacker may interact repeatedly with the model but have no control over external documents. An indirect-injection attacker may publish content that is later retrieved but may not directly address the agent. A multimodal attacker may provide inputs in one or more modalities. A memory attacker may influence content that is stored and reused. A tool-mediated attacker may exploit overly broad permissions or weak validation of tool arguments. Comparisons of attack success must therefore state the assumed access, model, system configuration, and evaluation setting.

3.4 Security objectives

We use six security objectives as a common analytical vocabulary: confidentiality, integrity, instruction authority, action authorization, availability, and traceability. Confidentiality prevents unauthorized disclosure of private data, credentials, hidden instructions, or protected context. Integrity prevents unauthorized modification of model context, retrieved knowledge, memory, tool arguments, or execution state. Instruction authority preserves the intended precedence between trusted policies and untrusted content, while action authorization ensures that external actions are permitted by the user, application policy, and least-privilege rules. Availability maintains access to the service and prevents excessive resource consumption or persistent task failure. Traceability preserves provenance and audit information sufficient to determine where information originated and why an action was taken.

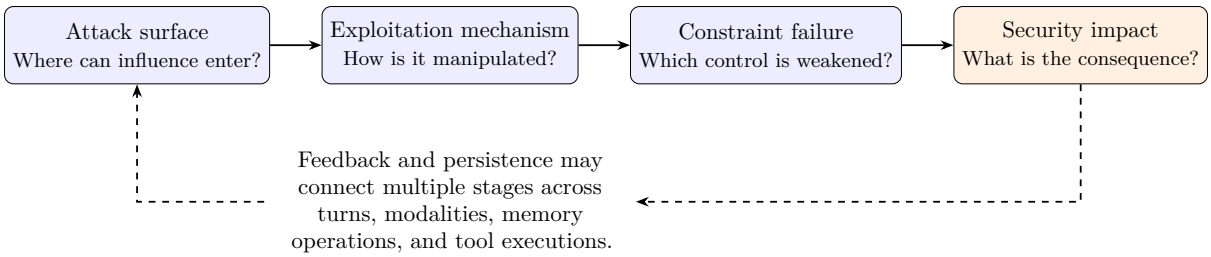


Figure 2: Four-dimensional analytical framework for describing security vulnerabilities. The sequence is a reasoning aid rather than a claim that every attack follows a strictly linear causal path.

4 Four-Dimensional Case-Analysis Framework

4.1 Four dimensions of vulnerability analysis

A vulnerability report often combines several different concepts. For example, "image jailbreak" may refer to the input channel, the adversarial strategy, and the resulting unsafe generation. To avoid category confusion, we describe each vulnerability using four dimensions. **Attack surface (AS).** The attack surface identifies the interface, component, or information channel through which an adversary can influence the system. Examples include prompts, conversation history, image and audio inputs, retrieved documents, memory stores, tool schemas, tool outputs, and inter-agent messages. **Exploitation mechanism (EM).** The exploitation mechanism describes how the adversary uses an attack surface. Examples include intent concealment, instruction injection, adversarial perturbation, semantic conflict, memory poisoning, reasoning manipulation, tool-argument manipulation, and malicious information propagation. **Constraint failure (CF).** The constraint failure identifies the security property or operational control that is violated or weakened. Examples include instruction-authority confusion, failure to preserve data boundaries, memory-integrity loss, insufficient action authorization, or unsafe output generation. **Security impact (SI).** The impact describes the consequence of the failure, such as harmful content generation, information disclosure, unauthorized external action, corrupted state, denial of service, or propagation of compromise. The dimensions are related but not interchangeable. An image is an attack surface; an embedded instruction is an exploitation mechanism; failure to treat that instruction as untrusted is a constraint failure; and an unauthorized action is an impact. One attack may involve multiple surfaces and mechanisms, and one mechanism may cause several constraint failures.

The coding taxonomy is designed to distinguish the role played by each dimension rather than to reproduce attack names used by individual studies. Attack surfaces identify the entry points through which adversarial influence reaches a system, whereas exploitation mechanisms describe the operations used to manipulate those entry points. Constraint failures represent the security properties or enforcement controls that are violated, while security impacts capture the observable consequences documented by the source. The dimensions are not assumed to be mutually exclusive at the attack level. For analytical consistency, representative cases are characterized using one primary category per dimension whenever the available source evidence supports such a distinction. Secondary characteristics are discussed when an attack involves multiple surfaces, mechanisms, constraint failures, or impacts. Cases with insufficient supporting evidence are explicitly identified rather than assigned speculative labels.

4.2 Formal representation

Let a system be represented by

$$\mathcal{S} = (M, X, C, Z, T, \Pi), \tag{2}$$

where M denotes the model or collection of models, X the input modalities, C the contextual information, Z the internal or persistent state, T the available tools and external interfaces, and Π the applicable policies and authorization rules.

A vulnerability instance can be represented as

$$v = (A, E, F, I, \Gamma), \tag{3}$$

where A is a set of attack surfaces, E a set of exploitation mechanisms, F a set of affected constraints, I a set of impacts, and Γ the assumptions describing attacker capabilities and deployment conditions. This representation is intentionally set-valued: an attack can cross several components and affect multiple properties.

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ denote the security requirements applicable to a deployment. A system execution trace is $\tau = (o_1, a_1, \ldots, o_T, a_T)$, containing observations and actions over time. A constraint violation occurs when a trace or its observable outcome fails a required property:

$$\exists P_i \in \mathcal{P} \quad \text{such that} \quad P_i(\tau) = \text{false}. \tag{4}$$

This abstraction does not imply that all safety policies can be reduced to binary predicates. Some properties are contextual, probabilistic, or dependent on human judgment. The notation simply makes explicit that a security claim should identify the property being evaluated and the execution conditions under which it is expected to hold.

4.3 Mapping system paradigms to the framework

Table 1 summarizes representative mappings. The entries are illustrative categories, not an exhaustive list of attacks or a claim that each mechanism is unique to one paradigm.

4.4 Research Scope and Positioning

Existing surveys have investigated the security of LLM-based systems from different perspectives. LLM-oriented surveys cover a broad range of security and privacy concerns, including prompt manipulation, training-time threats, model misuse, and risks associated with autonomous systems. Multimodal surveys primarily examine modality-specific and cross-modal attacks involving textual, visual, and auditory inputs. Meanwhile, surveys on agent and agentic-system security investigate threats and defenses associated with agent architectures, execution processes, interaction boundaries, and software lifecycles. Although these studies provide valuable insights into different aspects of LLM-based system security, their scopes and analytical emphases vary considerably. The major distinctions between representative surveys and the present study are summarized in Table 2.

Building on the security concerns investigated in these studies, the present work adopts a broader system-level perspective to examine vulnerabilities across LLMs, MLLMs, and agent-based systems. Rather than focusing exclusively on a particular attack category, modality, or system architecture, it analyzes security vulnerabilities through four complementary dimensions: Attack Surface (AS), Exploitation Mechanism (EM), Constraint Failure (CF), and Security Impact (SI). This perspective connects the components exposed to adversarial threats, the mechanisms through which vulnerabilities are exploited, the security constraints that fail, and the resulting consequences. It also provides a consistent basis for examining defensive strategies and identifying unresolved evaluation challenges across different LLM-based paradigms.

Table 1: Illustrative mapping of vulnerabilities to the four analytical dimensions.

System	Attack surface	Exploitation mechanism	Constraint at risk	Potential impact
LLM	User prompt, conversation context	Intent concealment, role-play, adversarial suffix, prompt optimization	Instruction authority; safe output policy	Harmful or disallowed generation
LLM with retrieval	Retrieved document, web page, knowledge base	Indirect prompt injection, poisoned source content	Trust separation; information provenance	Data disclosure; task redirection
MLLM	Image, audio, video, modality encoder	Embedded instruction, perturbation, modality conflict	Cross-modal policy consistency; trusted-input handling	Unsafe generation; misinterpretation
MLLM	Fusion and shared representation	Cross-modal semantic manipulation, conflicting cues	Consistent interpretation of policy-relevant evidence	Safety bypass or incorrect decision
Agent	Memory read/write interface	Memory poisoning, stale or adversarial state	Memory integrity; provenance preservation	Persistent behavioral manipulation
Agent	Planning and reasoning context	Goal manipulation, task-boundary reframing	Goal integrity; policy adherence	Unsafe plan or task deviation
Agent	Tool schema, arguments, execution result	Tool misuse, argument manipulation, excessive permissions	Action authorization; least privilege	Unauthorized external side effect
Multi-agent system	Messages and shared state	Malicious information propagation, trust abuse	Inter-agent trust; provenance and accountability	Cascading compromise

5 Proposed Four-Dimensional Case Analysis Method

This section operationalizes the four-dimensional framework, referred to as Four-Dimensional Security Case Analysis (FDSCA), by defining systematic characterization criteria and evidence-tracing rules. FDSCA provides a structured approach to examining documented vulnerabilities across heterogeneous LLM-based systems. Rather than introducing a new attack algorithm, trained classifier, or defense model, it establishes a consistent analytical procedure for identifying and distinguishing attack surfaces, exploitation mechanisms, security constraint failures, and security impacts. The framework is intended to facilitate transparent case interpretation and support future quantitative investigations based on systematically collected and independently annotated vulnerability records.

5.1 Method input and output

The input to FDSCA is a set of eligible attack reports selected according to the criteria in Section 2. Each report is decomposed into one or more distinct cases. The output for case i is a

Table 2: Scope-oriented comparison of representative security surveys and the present study. The descriptions summarize the primary focus and analytical emphasis of each work.

Existing work	**Primary scope**	**Organizing emphasis**	**Relationship to the present study**
Li and Fung [29]	LLM security and privacy	Inference-time and training-time threats, model misuse, privacy concerns, and agent-related risks	Provides a broad overview of LLM security; the present study further examines vulnerabilities across LLMs, MLLMs, and agent-based systems through a unified analytical perspective.
Wang et al. [19]	Multimodal jailbreaks	Modality-related jailbreak techniques and their effects on model behavior	Focuses primarily on jailbreak vulnerabilities in MLLMs; the present study additionally considers broader security threats, constraint failures, and their downstream impacts.
Kapoor et al. [30]	Text, image, video, and audio attacks	Practitioner-oriented coverage of adversarial threats across multiple modalities	Provides broad modality coverage; the present study further connects modality-related attack surfaces and exploitation mechanisms with security constraints and potential impacts.
Tang et al.; Tamuka et al.; Kovalev and Guan; Gan et al. [31, 32, 33]	Agent and agentic-system security	Agent attacks, defensive strategies, execution boundaries, and software-lifecycle risks	Provides agent-focused security analyses; the present study places agent vulnerabilities within a broader analytical scope encompassing LLMs and MLLMs.
Present study	**LLMs, MLLMs, and agent-based systems**	**Attack surfaces, exploitation mechanisms, constraint failures, and security impacts**	**Establishes a unified analytical perspective for examining vulnerabilities, defensive controls, and evaluation gaps across different LLM-based paradigms.**

record

$$z_i = (p_i, a_i, e_i, c_i, s_i, r_i), \tag{5}$$

where p_i denotes the system paradigm, a_i the attack-surface label, e_i the exploitation-mechanism label, c_i the security-constraint-failure label, s_i the security-impact label, and r_i the source evidence and evaluation metadata. The four primary labels correspond to AS, EM, CF, and SI. The system paradigm and evidence metadata are retained as contextual fields rather than additional analytical dimensions.

5.2 Analysis pipeline

FDSCA follows five sequential stages. The pipeline is applied to representative documented attack cases selected for qualitative analysis, rather than to an exhaustive or statistically sampled corpus of all published attacks. During **case identification**, concrete attack scenarios are extracted from eligible primary studies and benchmark reports; a case must describe an attack procedure or demonstrated security failure, rather than a general security concern. During **case normalization**, duplicate descriptions of the same attack instance are merged, while materially different configurations from one paper are separated only when their entry surfaces, exploitation procedures, or demonstrated failures differ. During **four-dimensional characterization**, each case is examined in terms of AS, EM, CF, and SI. A primary category is identified for each dimension when supported by the source evidence, while additional characteristics are retained in the accompanying discussion when necessary. During **evidence verification**, each

characterization is linked to a source location, such as a paper section, figure, table, or benchmark task; an unreported dimension is marked as unspecified rather than inferred. Finally, during **cross-case synthesis**, the characterized cases are compared to identify recurring vulnerability patterns, differences across system paradigms, and potential relationships among the four dimensions. This stage supports qualitative comparative analysis rather than statistically validated frequency or association estimates.

5.3 Coding decision rules

To improve analytical consistency, the following decision rules are applied when characterizing documented attack cases. First, AS is assigned to the component or channel through which the adversarial influence initially enters the system; downstream components are recorded in the evidence notes. Second, EM describes the operation performed by the attacker, not the vulnerability's consequence. Third, CF identifies the security property or control that fails, such as instruction authority, confidentiality, memory integrity, or action authorization. Fourth, SI records the most direct documented consequence; speculative downstream harms are not coded as observed impacts. When a case spans multiple stages, the primary label is selected according to the central mechanism emphasized by the original attack demonstration, and secondary labels are preserved separately.

5.4 Evidence quality and analytical scope

The quantitative evidence is obtained from three complementary evaluation resources. For AAEB, we conduct our own experiments and calculate attack success rates for five attack families across 13 LLMs under the experimental settings defined in this study. For JailbreakBench and AgentHarm, we examine the task definitions, evaluation settings, and statistics documented in their original publications and publicly available resources. These resources measure different properties and are therefore analyzed separately. Specifically, AAEB measures attack effectiveness under our experimental settings, JailbreakBench provides evidence concerning the consistency of automated safety judges against human reference labels, and AgentHarm describes the scale and category coverage of harmful agent tasks.

FDSCA itself is an analytical characterization framework. Its four dimensions are defined to support transparent source interpretation, but the present study does not claim measured inter-annotator reliability or predictive accuracy for the framework. A future replication with a released case-level coding sheet and independent annotators could evaluate coverage and Cohen's kappa for each dimension. Those statistics are not estimated here because the required independently coded records are not part of the evidence examined in this manuscript.

6 Case Analysis of Attacks

6.1 Case Analysis of Conventional LLM Attacks

Intent-concealment attacks obscure the adversary's actual objective by embedding a harmful request in a seemingly benign task, fictional scenario, translation request, or multi-step narrative. The attack exploits the model's tendency to interpret the full context rather than evaluate the request solely through isolated keywords. Role-playing and nested prompt structures can further reframe the task and encourage the model to treat policy-relevant content as hypothetical [34, 16].

The underlying weakness is not simply the presence of a particular phrase. Rather, the model must infer the relationship between the user's stated task, the requested content, and applicable

safety constraints. If the model overweights the local narrative or follows an untrusted instruction that conflicts with a higher-priority policy, safety behavior may degrade. Context reframing is therefore best described as a mechanism operating on the prompt and conversation-context surfaces, with instruction-authority and policy-interpretation constraints at risk.

Manual jailbreak construction can be supplemented by automated search. Greedy Coordinate Gradient (GCG) searches for adversarial suffixes by optimizing token sequences against a target objective [11]. AutoDAN generates more natural-looking jailbreak prompts, while COLD-Attack explores controllable and stealthy prompt generation [12, 15]. Black-box methods such as PAIR and Tree of Attacks through Inference (TAP) use model responses to iteratively refine candidate attacks [13, 14].

These methods differ in access assumptions and optimization procedures. White-box or gradient-informed attacks may require access to model gradients or a surrogate, whereas black-box methods rely on query feedback. Their common analytical feature is the search for inputs that shift model behavior toward an attacker-selected outcome. Reported success rates should not be compared without accounting for target model, policy, attack budget, refusal definition, and evaluator.

Prompt injection occurs when untrusted content is interpreted as an instruction that changes the model's behavior. Direct prompt injection is supplied through the user-facing interface; indirect prompt injection is embedded in external content that the model is asked to process, such as retrieved documents, web pages, or emails. The security challenge is to preserve the distinction between instructions that define the system's intended behavior and content that should be treated only as data.

A robust design should not rely solely on natural-language wording to establish this distinction. It should preserve provenance, isolate untrusted content where possible, restrict the actions that can be triggered by retrieved material, and validate tool arguments outside the model. Prompt injection is particularly consequential in agentic systems, but its conceptual roots are already present in context-conditioned LLM applications.

Inference-time attacks are not the only relevant threat. Training data poisoning can introduce malicious or biased associations into model behavior, while backdoor attacks may cause a model to behave unexpectedly when a trigger is present. These attacks differ from prompt-level jailbreaks because they target the data or training process rather than only the inference context. Their feasibility depends on access to training data, fine-tuning pipelines, model supply chains, or third-party components.

The distinction matters for defense. Input filtering may reduce some inference-time risks but cannot reliably remove a backdoor embedded in model parameters. Conversely, training-data auditing does not prevent a deployed model from following a malicious instruction retrieved at inference time. A complete threat analysis should therefore state whether the attack targets training, inference, or the surrounding application.

Prompt filters and refusal classifiers can detect known patterns but may be bypassed by paraphrasing, obfuscation, multilingual inputs, or novel compositions. Adversarial training can improve robustness against represented attack families, but generalization to unseen strategies is not guaranteed. Self-checking approaches can add a second reasoning pass, yet the checking process may share the same contextual vulnerabilities as the primary model. Goal-prioritization methods seek to preserve higher-level objectives under adversarial instructions, but their effectiveness depends on the training distribution and evaluation setting [35, 36].

The broader lesson is that model-level defenses are useful but should not be treated as the only security boundary. Applications that handle confidential data or external actions require independent access control, provenance tracking, and output validation.

6.2 Case Analysis of Multimodal LLM Attacks

MLLMs process inputs that may include text, images, audio, and video. Each modality introduces its own preprocessing and encoding pipeline. Image attacks may exploit visual perturbations, embedded text, layout, or image regions. Audio attacks may use spoken instructions, acoustic perturbations, or transcription ambiguities. Video introduces temporal sequences and synchronization between frames and audio. These modalities can also be combined, producing attacks that are difficult to characterize through a single-channel threat model.

The existence of a modality-specific attack surface does not automatically imply a successful jailbreak. An attack must survive preprocessing, be represented in a form the model can interpret, influence the relevant reasoning process, and lead to a policy-violating output or action. Evaluation should therefore distinguish perceptual success from security impact.

Visual inputs can contain text or semantic cues that influence model behavior. An image may include instructions that are not visible in the accompanying user prompt, or it may be constructed to elicit a particular interpretation. Research has demonstrated that visual adversarial examples can jailbreak aligned language models and that images can hijack generative models at runtime [17, 18]. Work on the visual vulnerabilities of MLLMs further illustrates the importance of image content and the interaction between visual and language pathways [20].

The security issue can be decomposed into three questions: whether the visual content is correctly perceived, whether its provenance and authority are interpreted correctly, and whether the resulting instruction is allowed to influence the system's behavior. A model may accurately read text in an image while still making an unsafe decision about whether that text should be followed. Thus, improving optical recognition alone is not a complete defense.

Cross-modal attacks exploit differences or conflicts between modalities. A benign textual instruction may be paired with an image or audio stream containing a conflicting instruction. Alternatively, one modality may distract the model while another carries the adversarial objective. Hybrid attacks can combine perturbations and semantic manipulation across channels.

A useful analytical distinction is between *perceptual inconsistency* and *authority inconsistency*. Perceptual inconsistency occurs when modalities provide conflicting descriptions of the same scene or event. Authority inconsistency occurs when one modality contains instructions that should not override trusted policies but are nevertheless treated as authoritative. The former is a multimodal understanding problem; the latter is a security and instruction-governance problem. They can co-occur but require different evaluations.

Audio introduces risks involving speech recognition, speaker attribution, acoustic perturbation, and the distinction between quoted speech and executable instruction. A system may transcribe a malicious utterance accurately but incorrectly treat it as an instruction from the user. Audio inputs also create an attribution problem: a spoken command may originate from the authenticated user, a recording, a background speaker, or a synthetic source. The security decision should therefore depend on source authentication and task authority, not solely on whether speech recognition succeeds. A modality-wide review of adversarial attacks identifies audio as one of the channels that must be considered alongside text, image, and video [30].

Video adds temporal structure: an instruction may appear briefly in a frame, be distributed across frames, or conflict with audio narration. Frame sampling can omit short-lived content, while temporal aggregation can combine individually innocuous segments into a policy-relevant instruction. A video-capable system should preserve timestamps and source provenance when extracting frame-level text, speech, or scene descriptions. Evaluation should report modality, temporal alignment, preprocessing, transcription or captioning behavior, and whether the model is expected to follow or merely describe embedded content. The available literature summarized here is more developed for visual jailbreaks than for unified security evaluation of audio and video; these modalities should therefore be treated as important but less consistently benchmarked

Table 3: Representative MLLM attack surfaces and the security questions they raise.

Modality or stage	**Representative attack pattern**	**Security question**	**Potential defensive control**
Text	Prompt injection, adversarial wording	Is the instruction trusted and within policy?	Instruction hierarchy, input screening, policy checks
Image	Embedded text, visual perturbation, image hijacking	Is visual content treated as data or as an authoritative instruction?	OCR provenance, image analysis, instruction isolation
Audio	Spoken injection, acoustic manipulation	Who is speaking, and should the utterance be executed?	Speaker/source checks, transcription provenance, confirmation
Video	Frame-level instruction, temporal conflict	Does the system preserve temporal and source context?	Frame sampling audits, temporal consistency checks
Fusion	Cross-modal semantic conflict	How are conflicting cues resolved under policy?	Cross-modal consistency checks, conservative conflict handling
Action interface	Unsafe response or action selection	Is the final response or action permitted?	Output moderation, action authorization, audit logging

attack surfaces, rather than as equally mature research areas.

Defenses should be applied at multiple points in the multimodal pipeline. Input-level checks can detect suspicious or malformed content, but may miss semantic instructions. Encoder-level defenses can improve robustness to perturbations, but do not necessarily resolve authority conflicts. Fusion-level mechanisms can compare modality-specific evidence and flag conflicts, but should not blindly privilege one modality. Output-level moderation can identify unsafe responses, yet it may occur after sensitive information has already been exposed or an action has been selected.

A stronger design should maintain provenance and instruction status for each modality-derived element. Table 3 summarizes representative modality-specific attack patterns and corresponding security questions. For example, text extracted from an image should remain marked as untrusted visual content rather than being silently converted into an ordinary instruction. Cross-modal safety evaluation should test both agreement and disagreement conditions and should separately measure perception, instruction interpretation, refusal behavior, and downstream action.

6.3 Case Analysis of Agent-based System Attacks

Agents often use short-term context, long-term memory, vector stores, or external knowledge bases to preserve information across interactions. These mechanisms improve continuity and task performance but create persistent attack surfaces. AgentPoison studies attacks that target agent memory or knowledge bases, demonstrating how poisoned information can influence subsequent agent behavior [4].

Memory poisoning differs from a one-time prompt attack in persistence and reuse. Malicious content may be stored during one interaction and retrieved during a later task, potentially after the original source is no longer visible. A memory system should therefore record provenance, source trust, timestamp, scope, and the conditions under which a memory may be used. Read and write permissions should be separated, and sensitive or instruction-like memories should not automatically acquire authority merely because they have been stored.

Tools turn model outputs into external effects. A tool call may read private files, query databases, send messages, modify records, execute code, or initiate transactions. If the model can choose tools and arguments based on untrusted context, a malicious instruction may induce an action that the user did not authorize. Tool descriptions and outputs can themselves become injection channels.

A secure tool interface should follow least privilege. Each tool should expose only the minimum operations and data required for the task. Arguments should be validated against schemas and authorization rules outside the model. High-impact actions should require explicit confirmation or independent policy checks. Read-only and write-capable tools should be separated where feasible, and execution logs should preserve the initiating user, agent, input provenance, arguments, and result.

Retrieval-augmented generation (RAG) connects a model to external documents or databases. This can improve factual grounding, but retrieved content is not necessarily trustworthy. An attacker may place malicious instructions in a document that is later retrieved for an otherwise benign query. The model may then confuse document content with instructions governing its behavior. Related work has explored RAG injection and data extraction risks in retrieval-augmented applications [26, 23].

The key control is to treat retrieved documents as evidence, not authority. Systems should preserve source attribution, apply access controls before retrieval, constrain the operations that retrieved text can trigger, and distinguish quoted or retrieved instructions from system and user instructions. Retrieval ranking and document sanitization can help, but neither guarantees that all malicious content will be detected.

An attack against an agent-based system may span several components rather than remain confined to the interface through which it first enters. For example, adversarial content may be introduced through a retrieved document, interpreted as task-relevant context during planning, written into persistent memory, retrieved in a later session, and eventually used to justify a tool call. The initial attack surface, the mechanism that carries the influence forward, and the final impact are therefore not necessarily located in the same component.

This propagation pattern can be described as a sequence of trust-boundary transitions: external content enters the observation or retrieval layer; the agent incorporates it into a reasoning or planning context; state-management logic preserves some part of it; a later retrieval or inter-agent message reintroduces it; and an action interface converts the resulting plan into an external side effect. At each transition, provenance, authority, and permission scope can be lost or incorrectly inferred. The security analysis should consequently record both the initial entry point and the downstream components affected. This is also why a component-level detector alone may be insufficient: the harmful behavior may emerge only after multiple individually plausible operations are composed.

Multi-agent systems distribute tasks across specialized agents. This architecture may improve modularity and parallelism, but it also introduces communication channels and delegated authority. A compromised or manipulated agent may transmit false information, malicious instructions, or unsafe plans to other agents. Research on the "wolf within" and the spread of manipulated knowledge in LLM-based multi-agent communities illustrates this risk [8, 28].

The central issue is that agents may treat messages from peer agents as more trustworthy than arbitrary external content. Yet a message's origin does not guarantee its integrity or authorization. Secure multi-agent systems should authenticate agent identities, preserve message provenance, define explicit trust levels, restrict delegated permissions, and require independent validation for high-impact actions. Trust should be attached to the content and its evidence, not inferred solely from the identity of the transmitting agent.

7 Cross-paradigm Comparative Analysis

7.1 From contextual manipulation to stateful compromise

The evolution from LLMs to MLLMs and agents should not be interpreted as a strict replacement of one threat class by another. Prompt injection remains relevant in multimodal and agentic

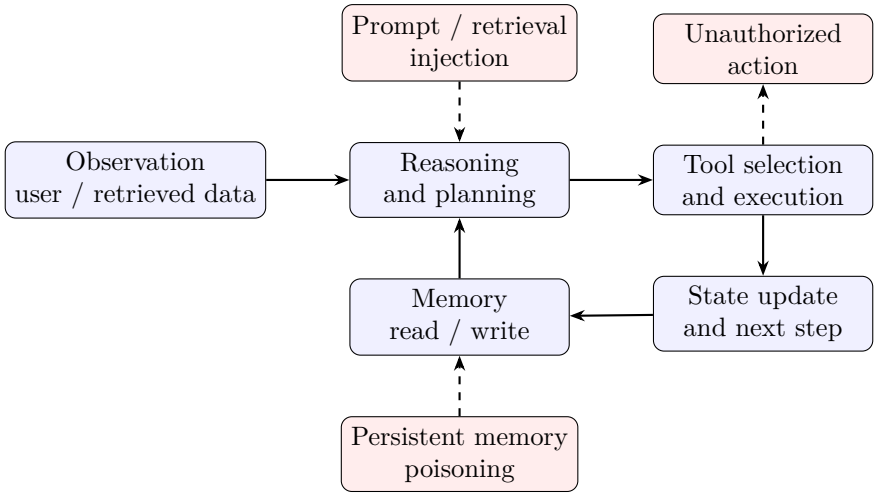


Figure 3: A simplified agent execution loop and representative security intervention points. Adversarial content may enter at observation, memory, planning, or tool interfaces and may persist across iterations.

systems; multimodal models may be embedded within agents; and agents may use conventional LLMs as their reasoning engines. What changes is the number of components and trust boundaries through which an adversarial influence can travel.

In a text-only interaction, an attack may affect the current response. In an MLLM, the attack may enter through a non-text modality and influence a shared semantic interpretation. In an agent, the same influence may be stored, retrieved, passed to another component, or converted into an external action. The security impact therefore depends not only on whether the model produces an unsafe sentence but also on whether the system persists, propagates, or executes the adversarial content.

7.2 Comparative characteristics

Table 4 compares the three paradigms at a conceptual level. The table describes typical tendencies, not universal properties: individual systems may have additional controls or different architectures.

7.3 Compositional attacks and propagation

A compositional attack combines multiple mechanisms or surfaces. For example, malicious text may be embedded in an image, transcribed or extracted by an MLLM, summarized into memory, retrieved in a later task, and then used to influence a tool call. Each component may appear to operate normally in isolation, while the overall system fails because provenance and authority are lost across component boundaries.

This observation motivates a distinction between *local robustness* and *compositional robustness*. Local robustness measures whether a component resists an attack under a specified input condition. Compositional robustness measures whether the end-to-end system preserves its security properties when multiple components interact and information crosses trust boundaries. A system may perform well on isolated prompt-injection tests yet remain vulnerable when retrieved content is stored in memory and later used to justify an action.

Table 4: Cross-paradigm comparison of security characteristics.

Dimension	Conventional LLM	MLLM	Agent-based system
Primary interaction	Text prompt and conversation	Text plus image, audio, video, or other modalities	User, external data, tools, memory, and peer agents
Typical attack surfaces	Prompt, context, training data	Prompt, modality encoders, fusion, output	Prompt, retrieval, memory, tools, state, inter-agent messages
Representative mechanisms	Intent concealment, suffix optimization, prompt injection	Visual/audio injection, perturbation, cross-modal conflict	Memory poisoning, tool misuse, propagation
State persistence	Often limited to conversation context	Conversation and multimodal context	Explicit short- and long-term state, memory, external records
External side effects	Usually indirect through generated content	Usually indirect unless connected to tools	Direct through tool calls and delegated execution
Key security concerns	Policy adherence and confidentiality	Modality provenance and consistent policy interpretation	Integrity, authorization, provenance, and safe execution
Useful controls	Alignment, input/output checks, policy separation	Modality-aware inspection, provenance, fusion checks	Least privilege, sandboxing, memory controls, action approval
Evaluation emphasis	Refusal, harmfulness, leakage, robustness	Perception, cross-modal consistency, harmfulness	Task completion, unauthorized actions, persistence, side effects

7.4 A mechanism-to-control mapping

The mapping in Table 5 links the principal attack mechanisms discussed above to layered defensive strategies and their unresolved limitations.

The mapping suggests that no single defense category covers all failure modes. Alignment is relevant to model behavior but cannot replace access control. Provenance supports trust decisions but does not by itself determine whether an action is safe. Input checks can detect known patterns but may fail on semantic or compositional attacks. State protection is essential for persistent memory, while action controls are necessary whenever model outputs can trigger external side effects. Effective protection therefore requires layered controls whose responsibilities are explicit.

8 Statistical Results and Comparative Evaluation

This section presents the experimental results obtained from the proposed Agent Attack Evaluation Benchmark (AAEB). We evaluate 13 LLMs across five agent attack families: direct prompt injection (DPI), indirect prompt injection (IPI), memory poisoning, mixed attacks, and Plan-of-Thought (PoT) backdoors. Attack effectiveness is measured using the attack success rate (ASR), calculated as the proportion of successfully completed attack attempts among all valid attempts under each experimental setting. For presentation, ASR values are expressed on a normalized scale ranging from 0 to 1, where higher values indicate greater attack effectiveness. This evaluation enables comparisons across attack families and LLMs under the same benchmark protocol.

The evaluation applies the same set of attack test cases to all 13 models to maintain consistency across model-level comparisons. Each attack case is evaluated according to its

Table 5: Mapping representative vulnerability mechanisms to defensive strategies and their remaining challenges.

Vulnerability mechanism	**Primary defensive strategies**	**Complementary defensive strategies**	**Remaining challenges**
Intent concealment	Safety alignment and context-aware policy interpretation	Adversarial training and input-level detection	Unseen contextual framing and semantically disguised malicious intent
Adversarial suffix	Adversarial training and robustness optimization	Input screening and inference-time safety verification	Transferability across models and adaptive optimization of adversarial inputs
Indirect prompt injection	Instruction hierarchy enforcement and untrusted-content isolation	Source provenance tracking, input screening, and tool permission restrictions	Semantic injection hidden in retrieved content and cross-component instruction propagation
Cross-modal injection	Modality-aware safety alignment and cross-modal consistency verification	Input inspection, source provenance tracking, and output safety filtering	Conflicting modality-specific instructions and vulnerabilities in multimodal fusion
Memory poisoning	Memory integrity verification and access-controlled memory management	Source provenance tracking, memory validation, and retrieval filtering	Persistent malicious information and delayed activation across interactions
Tool misuse	Least-privilege access control and independent action authorization	Tool argument validation, execution sandboxing, and audit logging	Over-privileged tools and unauthorized external side effects
Inter-agent propagation	Inter-agent authentication and trust management	Message provenance tracking, permission isolation, and independent validation	Trust propagation, compromised agents, and cascading failures across agent interactions

corresponding attack-success criterion, and only valid attack attempts are included in the ASR calculation. The resulting ASRs are calculated separately for each model and attack family. This protocol enables a consistent comparison of attack effectiveness while preserving the distinctions among different attack scenarios.

The results reveal clear differences in attack effectiveness across the five attack families. Mixed attacks achieve the highest average ASR of 0.536, followed by direct prompt injection at 0.449. PoT backdoors and indirect prompt injection achieve average ASRs of 0.235 and 0.208, respectively, while memory poisoning exhibits the lowest average ASR of 0.155. These findings suggest that attack effectiveness varies substantially across different exploitation strategies. In particular, mixed attacks combine multiple attack surfaces or exploitation mechanisms, which may increase the likelihood of compromising agent-based systems under the evaluated settings. Nevertheless, these results should be interpreted within the scope of AAEB rather than as universal indicators of security vulnerability.

The model-level comparison also reveals considerable variation across models. Mistral-7B-Instruct achieves the highest average ASR of 0.519, followed by Qwen2.5-7B at 0.487 and Llama-3.1-8B at 0.458. In contrast, Gemini-2.5-Pro achieves the lowest average ASR of 0.159, followed by GPT-4.1 at 0.173 and Claude-3.7-Sonnet at 0.191. These differences suggest that vulnerability tendencies may vary across model architectures, parameter scales, and alignment strategies. However, the scores should not be interpreted as definitive security rankings, since

Table 6: Attack success rates (ASRs) of 13 LLMs across five attack families in the proposed AAEB benchmark.

LLM	DPI	IPI	Memory poisoning	Mixed attack	PoT backdoor	Average ASR
Qwen2.5-7B	0.723	0.314	0.247	0.786	0.365	0.487
Qwen2.5-72B	0.496	0.227	0.176	0.593	0.258	0.350
Llama-3.1-8B	0.687	0.296	0.224	0.748	0.337	0.458
Llama-3.1-70B	0.438	0.186	0.137	0.517	0.206	0.297
Mistral-7B-Instruct	0.756	0.347	0.276	0.817	0.398	0.519
Mixtral-8x22B	0.487	0.218	0.168	0.586	0.246	0.341
DeepSeek-V3	0.317	0.157	0.116	0.427	0.178	0.239
Claude-3.7-Sonnet	0.248	0.126	0.087	0.357	0.138	0.191
GPT-4.1	0.228	0.117	0.076	0.318	0.127	0.173
Gemini-2.5-Pro	0.207	0.106	0.067	0.297	0.116	0.159
Phi-4	0.563	0.273	0.193	0.657	0.306	0.398
GLM-4-Plus	0.397	0.196	0.146	0.478	0.217	0.287
InternLM2.5-20B-Chat	0.287	0.137	0.097	0.387	0.157	0.213
Average	**0.449**	**0.208**	**0.155**	**0.536**	**0.235**	**0.316**

actual attack outcomes are also influenced by system prompts, tool permissions, memory configurations, retrieval mechanisms, and deployment environments.

Overall, the comparative analysis emphasizes that security vulnerabilities in LLM-based agents cannot be adequately characterized by a single attack category or model-level indicator. Different attack families exhibit distinct vulnerability patterns, while the same attack mechanism may manifest differently across LLMs. These observations reinforce the value of the FDSCA framework in separating attack surfaces, exploitation mechanisms, security constraint failures, and security impacts, thereby supporting a more structured analysis of security risks in LLM-based systems.

8.1 Benchmark coverage and system paradigms

The selected benchmarks cover different parts of the security problem. AgentHarm contains 110 explicitly malicious agent tasks, or 440 instances when augmentations are included, and spans 11 harm categories. Its evaluation requires an agent to maintain the capabilities needed to complete a multi-step harmful task after a jailbreak, rather than merely produce a single unsafe sentence [22]. AAEB instead evaluates agent-level compromise across scenarios, tools, attack methods, and LLMs. JailbreakBench focuses on standardized jailbreak evaluation and the reliability of safety judges. These resources are complementary, but their task counts and metrics should not be added together or interpreted as a common denominator.

The benchmark evidence also illustrates why system paradigm matters. Conventional LLM jailbreak benchmarks primarily assess unsafe response generation. MLLM attacks add modality-specific input and representation pathways, such as image, audio, and video encoders, and may involve failures in perception, cross-modal fusion, or policy interpretation. Agent benchmarks extend the outcome space to memory modification, retrieval influence, tool invocation, and external side effects. Consequently, a single ASR value cannot adequately characterize all four FDSCA dimensions across these paradigms.

8.2 Cross-dimensional evidence synthesis

The source-grounded mappings reveal recurring relationships between the four dimensions. Prompt injection typically enters through user prompts, retrieved documents, or tool outputs; its exploitation mechanism is the placement or reframing of instructions in content that the model

processes; the constraint at risk is the separation between trusted instructions and untrusted data; and the potential impact includes task redirection, disclosure, or unauthorized action. In multimodal settings, the attack surface includes image, audio, and video inputs, while the exploitation mechanism may embed or encode instructions in a modality that is not treated as an authoritative instruction channel. In agent settings, memory poisoning and tool manipulation can affect persistent state and external actions, placing integrity and action-authorization constraints at risk.

These mappings are analytical interpretations of documented attack procedures, not frequency estimates. The present evidence does not provide a unified case-level contingency table from which statistically valid associations between EM, CF, and SI could be calculated. We therefore do not report chi-square tests, Cramer's V, or claims that a particular mechanism is statistically associated with a specific impact. Such analyses would require a deduplicated case-level dataset with explicit primary labels and documented coding decisions.

9 Security Implications and Defensive Controls

9.1 Training-based alignment

Training-based defenses attempt to shape model behavior through supervised examples, preference optimization, adversarial training, and explicit safety objectives. RLHF is a widely used approach for aligning generated responses with human preferences [2]. Goal-prioritization methods aim to preserve higher-level objectives when instructions conflict [35]. Such approaches can improve refusal behavior and reduce susceptibility to known attacks.

Their limitations include incomplete coverage of the attack space, dependence on training data and preference labels, and possible trade-offs with utility. An attacker can also adapt prompts to exploit contexts that were poorly represented during training. Most importantly, model alignment does not enforce application-level permissions. A model may be trained not to disclose secrets, but the application should still prevent unauthorized access to those secrets.

9.2 Detection and filtering

Detection-based defenses inspect prompts, retrieved content, or model outputs for suspicious patterns. Systems such as JailGuard explore detection of prompt-based attacks through input analysis [37]. Detection can be useful as one layer of defense, particularly for known attack families and high-risk interfaces.

However, detection faces a coverage problem: legitimate and malicious content can be semantically similar, and attacks can be paraphrased or distributed across modalities and multiple steps. False positives may block benign tasks, while false negatives allow attacks through. A detector should therefore be evaluated under adaptive attacks and reported with precision, recall, false-positive rate, latency, and the specific attack distribution used.

9.3 Architectural and inference-time safeguards

Self-defense methods ask models to inspect inputs or their own candidate responses before producing an answer. SelfDefend is an example of work exploring model-internal defense against jailbreaks [36]. Other approaches use separate guard models, policy classifiers, constrained decoding, or structured output validation.

These mechanisms can improve resilience, but a safeguard that relies on the same model and context may share the original vulnerability. A second model may also inherit similar biases or fail on the same adversarial composition. Independent enforcement is preferable for high-impact

operations: authorization should be checked by deterministic application logic or a trusted policy engine rather than inferred only from generated natural language.

9.4 Multimodal defenses

MLLM defenses should preserve the source and status of modality-derived information. Image OCR, speech transcription, and video captioning should retain provenance metadata. A system should distinguish an instruction spoken by an authenticated user from text appearing in a background recording or an image supplied for analysis. Cross-modal consistency checks can identify conflicts, but a conflict should trigger careful handling rather than an arbitrary preference for one modality.

Defenses should also be evaluated at multiple stages: input perception, representation, fusion, policy interpretation, and output. A model that correctly recognizes an embedded instruction but refuses to execute it is different from a model that fails to perceive it. Both outcomes matter, but they measure different properties.

9.5 Memory and retrieval protection

Memory and retrieval defenses should address the full data lifecycle: ingestion, validation, storage, retrieval, update, and deletion. Useful controls include source authentication, provenance records, content classification, access control, temporal validity, integrity checks, and restrictions on which memories may influence which tasks. Retrieved content should be treated as untrusted evidence unless its authority has been explicitly established.

A further challenge is delayed activation. A malicious memory may appear harmless when stored but become dangerous when retrieved in a different context. Evaluations should therefore test multi-turn and cross-session behavior, not only immediate responses to poisoned content.

9.6 Tool authorization and system-level isolation

When agents can invoke tools, the system should enforce least privilege and separate planning from execution. The model may propose an action, but an independent component should verify the user's authority, the action's scope, the target resource, and the arguments. High-impact operations should require confirmation or policy approval. Sandboxing, network restrictions, rate limits, and audit logs can reduce the consequences of model errors.

System-level isolation is particularly important because natural-language policies are probabilistic. A model can misunderstand an instruction; an access-control mechanism should not. The safest architecture assumes that model outputs may be manipulated and constrains what those outputs are allowed to do.

10 Evaluation Context and Benchmark Characteristics

10.1 Why evaluation must be system-aware

Security evaluation should measure the property that a defense claims to protect. Refusal rate alone is insufficient for an agent that can access files or execute tools. Similarly, a multimodal benchmark should distinguish whether the model perceived an adversarial instruction, whether it interpreted the instruction as authoritative, and whether it generated or executed a harmful outcome.

Evaluation results are conditional on the model version, system prompt, decoding settings, tool configuration, attack budget, and evaluator. Comparisons across papers are unreliable

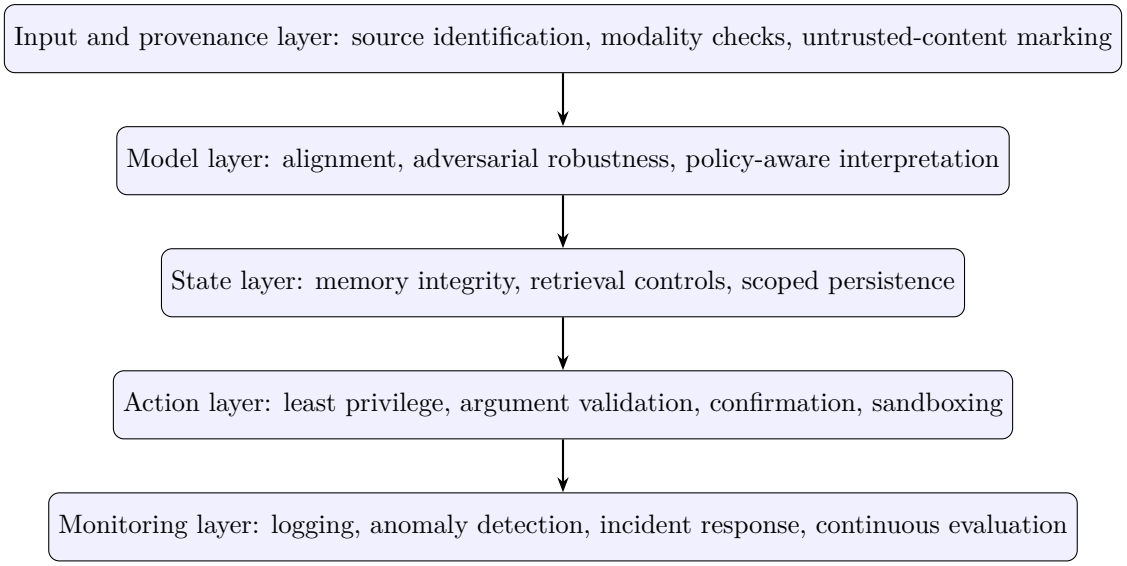


Figure 4: Defense-in-depth architecture for LLM-based systems. The layers are complementary; no single layer should be assumed to provide complete protection.

when these details are missing or inconsistent. Reproducible reports should specify the target system, attacker access, number of queries, input modalities, success definition, evaluator, and safety-utility trade-off.

10.2 Benchmark characteristics and evaluation gaps

Text-based jailbreak benchmarks provide controlled settings for evaluating whether language models resist adversarial prompts and generate policy-compliant responses. However, their evaluation outcomes are generally tied to specific target models, attack configurations, and safety judges. The judge-level evaluation results reported by JailbreakBench further highlight differences in automated evaluator reliability. On its 300-example evaluation set, Llama-3-70B and GPT-4 achieved agreement rates of 90.7% and 90.3% with majority-vote human annotations, respectively, while Llama Guard 2 and HarmBench achieved 87.7% and 78.3%. Notably, HarmBench exhibited a false positive rate (FPR) of 26.8%, whereas Llama-3-70B achieved the lowest false negative rate (FNR) of 5.5%. These differences indicate that automated evaluators may produce substantially different assessments of jailbreak effectiveness [38]. Moreover, text-based benchmarks cannot fully capture vulnerabilities arising from multimodal inputs, persistent memory, or external tool execution.

Agent-oriented benchmarks extend security evaluation beyond single-turn response generation by considering multi-step task execution, tool interactions, and potentially harmful external actions. Nevertheless, their coverage remains dependent on the selected task categories, tool environments, permission configurations, and execution settings. Benchmark performance may not fully reflect delayed memory poisoning, cross-session attacks, or compositional vulnerabilities involving multiple components.

Several evaluation gaps therefore remain. First, compositional attacks are underrepresented when benchmarks test one attack surface at a time. Second, multimodal benchmarks may conflate perception failure with policy failure. Third, agent evaluations often require longer interaction horizons to reveal delayed memory poisoning or attack propagation. Finally, security and utility should be evaluated jointly: a defense that blocks all tool use may prevent some attacks but also eliminate the intended functionality of the agent. These limitations motivate evaluations that incorporate realistic authorization boundaries, longer interaction horizons, and

end-to-end security outcomes.

11 Discussion and Threats to Validity

11.1 Future Research Directions

Several directions deserve further investigation to improve the security of LLM-based systems. First, provenance-aware instruction handling is essential for distinguishing trusted instructions from untrusted content. Retrieved documents, OCR results, tool responses, and peer-agent messages should not acquire instruction authority merely by being incorporated into the model context. Future research should explore structured instruction channels, provenance-aware representations, and policy enforcement mechanisms that operate independently of natural-language interpretation.

Second, cross-modal safety consistency remains a critical challenge for MLLMs. Security policies should be preserved regardless of whether instructions are expressed through text, images, audio, or video, including adversarial combinations across modalities. This requires modality-aware provenance tracking, conflict detection, and calibrated uncertainty estimation, while accounting for the distinct noise, ambiguity, and spoofing characteristics of each modality. Similarly, persistent memory introduces security risks that extend beyond retrieval relevance. Memory systems should support provenance tracking, integrity verification, scoped access, expiration, correction, and deletion, while distinguishing ordinary user preferences and factual records from instructions capable of altering system behavior. Evaluation should further consider delayed activation, cross-session persistence, and interactions between benign and malicious memory entries.

Third, tool-using agents require stronger execution-time security guarantees. Rather than relying solely on model confidence, tool invocation should be governed by explicit authorization mechanisms, such as typed interfaces, capability-based access control, policy engines, execution sandboxes, transaction confirmation, and verifiable audit trails. In this setting, models may propose actions, but independent enforcement mechanisms should determine whether those actions are authorized. For multi-agent systems, corresponding trust mechanisms are needed to verify agent identities, preserve message provenance, constrain delegated authority, and protect shared memory. In particular, consensus among agents should not be regarded as sufficient evidence of correctness when agents share compromised information sources or when malicious agents can impersonate trusted components.

Finally, security evaluation should move beyond static benchmarks toward adaptive and compositional assessment. Future evaluations should incorporate adaptive attackers, unseen attack compositions, cross-model transfer, long-horizon tasks, and realistic permission configurations. Reporting standards should explicitly specify threat assumptions, attack success criteria, reproducible configurations, and artifact availability. Formal methods can further strengthen these evaluations by specifying and enforcing system-level security invariants, such as authorization requirements for sensitive actions, restrictions on information flows to untrusted destinations, and protections against unauthorized modifications to persistent state. Although formally verifying the behavior of generative models remains challenging, deterministic enforcement at security-critical boundaries offers a practical way to complement probabilistic model behavior.

11.2 Threats to Validity

Several limitations should be considered when interpreting the findings. First, the evidence base is restricted to publicly documented studies and benchmarks and may overrepresent attacks that are easier to demonstrate, publish, or reproduce. Second, the quantitative evidence comes

from different sources and evaluation protocols. AAEB results are obtained from our own experiments, whereas the AgentHarm and JailbreakBench statistics are examined based on their original publications and publicly available resources. These results are therefore analyzed separately rather than pooled into a single cross-benchmark score. Third, the four-dimensional mapping involves interpretive judgments when a reported attack spans multiple components or consequences. We mitigate this limitation by defining the dimensions separately and retaining only source-supported claims; nevertheless, the present manuscript does not report independent inter-annotator agreement. Fourth, benchmark coverage is not equivalent to the prevalence of vulnerabilities in deployed systems. Finally, the study is a structured synthesis of available evidence rather than an exhaustive census or a new controlled evaluation of all LLM, MLLM, and agent systems. A future release of a deduplicated case-level dataset with independent annotations would enable formal reliability testing and cross-dimensional statistical analysis.

12 Conclusion

This paper presents a four-dimensional analytical framework for examining security vulnerabilities in LLM-based systems, complemented by an experimental evaluation of agent attacks using the proposed AAEB benchmark. The proposed FDSCA framework distinguishes attack surface, exploitation mechanism, security constraint failure, and security impact, providing a consistent vocabulary for examining documented attacks across conventional LLMs, MLLMs, and agent-based systems. It emphasizes the distinction between where adversarial influence enters, how it is exercised, which security property is weakened, and what consequences are documented. Through experiments conducted on the proposed AAEB benchmark, this study identifies substantial variations in attack effectiveness across different agent attack families and LLMs. The examination of existing resources, including AgentHarm and JailbreakBench, highlights the importance of multi-step agent evaluation and reliable automated safety assessment, particularly given the differences in automated evaluator agreement and classification error profiles reported by JailbreakBench. Meanwhile, AgentHarm highlights the importance of evaluating multi-step harmful agent behavior rather than relying solely on single-turn response refusal.

The findings support three broader conclusions. First, attack effectiveness varies substantially across attack families, models, and system configurations. Second, multimodal perception, persistent memory, retrieval, and tool execution introduce additional security boundaries that are not adequately captured by text-only refusal metrics. Third, the reliability of evaluation methods is essential for drawing sound conclusions about the security of LLM-based systems. This study does not claim to provide a newly collected case corpus, independently measured annotation agreement, or statistical associations that cannot be established from the published source data. Its limitations include public-reporting bias, heterogeneous benchmark protocols, and the interpretive nature of cross-dimensional mapping. Future work should develop a deduplicated case-level dataset with source-linked evidence, independent annotations, and a preregistered coding guide. Such a resource would facilitate reproducible frequency analysis, inter-annotator reliability assessment, and statistically grounded investigations into the relationships among exploitation mechanisms, security constraint failures, and security impacts.

Acknowledgments

The authors have no additional acknowledgments to declare.

Conflict of Interest

The authors declare that they have no competing interests.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] Long Ouyang, Jeffrey Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[3] J Achiam. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[4] Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37:130185–130213, 2024.

[5] Yanxu Mao, Tiehan Cui, et al. From llms to mllms to agents: A survey of emerging paradigms in jailbreak attacks and defenses within llm ecosystem. *ACM Transactions on Knowledge Discovery from Data*, 2026.

[6] Jinyang Wang, Shuai Zhang, Chang Liu, et al. Pidrcmpp: Rapid multi-strategy hierarchical jailbreak attacks on llms. In *International Conference on Intelligent Computing*, pages 89–100. Springer, 2025.

[7] Xuannan Liu, Xing Cui, Peipei Li, et al. Jailbreak attacks and defenses against multimodal generative models: A survey. *arXiv preprint arXiv:2411.09259*, 2024.

[8] Zhen Tan, Chengshuai Zhao, Raha Moraffah, Yifan Li, Yu Kong, Tianlong Chen, and Huan Liu. The wolf within: Covert injection of malice into mllm societies via an mllm operative. *arXiv preprint arXiv:2402.14859*, 2024.

[9] Congying Liu, Xingyuan Wei, et al. Biomedsearch: A multi-source biomedical retrieval framework based on llms. In *2025 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2516–2521. IEEE, 2025.

[10] Congying Liu, Taihao Li, Ming Huang, Xingyuan Wei, et al. Protrlsearch: A multi-round multimodal protein search agent with large language models trained via reinforcement learning. *arXiv preprint arXiv:2603.01464*, 2026.

[11] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

[12] Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. In *International Conference on Learning Representations*, volume 2024, pages 56174–56194, 2024.

[13] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 23–42. IEEE, 2025.

[14] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically. *Advances in Neural Information Processing Systems*, 37:61065–61105, 2024.

[15] Xingang Guo, Fangxu Yu, Huan Zhang, Lianhui Qin, and Bin Hu. Cold-attack: Jailbreaking llms with stealthiness and controllability. *arXiv preprint arXiv:2402.08679*, 2024.

[16] Peng Ding, Jun Kuang, Dan Ma, et al. A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2136–2153, 2024.

[17] Xiangyu Qi, Kaixuan Huang, et al. Visual adversarial examples jailbreak aligned large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 21527–21536, 2024.

[18] Luke Bailey, Euan Ong, Stuart Russell, and Scott Emmons. Image hijacks: Adversarial images can control generative models at runtime. *arXiv preprint arXiv:2309.00236*, 2023.

[19] Siyuan Wang, Zhuohan Long, Zhihao Fan, and Zhongyu Wei. From llms to mllms: Exploring the landscape of multimodal jailbreaking. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17568–17582, 2024.

[20] Yifan Li, Hangyu Guo, et al. Images are achilles' heel of alignment: Exploiting visual vulnerabilities for jailbreaking multimodal large language models. In *European Conference on Computer Vision*, pages 174–189. Springer, 2024.

[21] Yanxu Mao, Peipei Liu, et al. Divide and conquer: A hybrid strategy defeats multimodal large language models. *arXiv preprint arXiv:2412.16555*, 2024.

[22] Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, et al. Agentharm: A benchmark for measuring harmfulness of llm agents. In *International Conference on Learning Representations*, volume 2025, pages 79185–79220, 2025.

[23] Changyue Jiang, Xudong Pan, Geng Hong, Chenfu Bao, Yang Chen, and Min Yang. Feedback-guided extraction of knowledge base from retrieval-augmented llm applications. *arXiv preprint arXiv:2411.14110*, 2024.

[24] Yanxu Mao, Tiehan Cui, et al. Exploring the security of short-term and long-term contextual memory in industrial smart customer agent. *Automated Software Engineering*, 33(3):85, 2026.

[25] Yanxu Mao, Peipei Liu, et al. Stop fixating on prompts: Reasoning hijacking and constraint tightening for red-teaming llm agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26068–26085, 2026.

[26] Shunli Zhang, Ying Ding, et al. You are an expert: Rag injection and guided error expert activation for jailbreaking large language models. *Expert Systems*, 43(9):e70395, 2026.

[27] Tiehan Cui, Peipei Liu, et al. Nexus: Continual learning of symbolic constraints for safe and robust embodied planning. *arXiv preprint arXiv:2605.09387*, 2026.

[28] Tianjie Ju, Yiting Wang, Yi Hua, Xinbei Ma, Pengzhou Cheng, Haodong Zhao, Yulong Wang, Lifeng Liu, Jian Xie, Zhuosheng Zhang, et al. Flooding spread of manipulated knowledge in llm-based multi-agent communities. *Science China Information Sciences*, 69(7):172103, 2026.

[29] Miles Q Li and Benjamin CM Fung. Security concerns for large language models: A survey. *Journal of Information Security and Applications*, 95:104284, 2025.

[30] Shashank Kapoor, Sanjay Surendranath Girija, et al. Adversarial attacks in multimodal systems: A practitioner's survey. In *2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1643–1650. IEEE, 2025.

[31] Yaxin Tang, Yijia Liu, Jiahe Lan, Zheng Yan, and Erol Gelenbe. Security of llm-based agents regarding attacks, defenses, and applications: A comprehensive survey. *Information Fusion*, page 103941, 2025.

[32] Nyashadzashe Tamuka, Topside Ehleketani Mathonsi, et al. Securing llm-based agents against cyberattacks: a comprehensive survey on attack techniques and defense strategies. *Journal of Computer Virology and Hacking Techniques*, 22(1):38, 2026.

[33] Yuyou Gan, Yong Yang, Zhe Ma, et al. Navigating the risks: A survey of security and privacy threats in llm-based agents. *ACM Transactions on Software Engineering and Methodology*, 2026.

[34] Tiehan Cui, Yanxu Mao, et al. Exploring jailbreak attacks on llms through intent concealment and diversion. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20754–20768, 2025.

[35] Zhexin Zhang, Junxiao Yang, Pei Ke, Fei Mi, Hongning Wang, and Minlie Huang. Defending large language models against jailbreaking attacks through goal prioritization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8865–8887, 2024.

[36] Xunguang Wang, Daoyuan Wu, Zhenlan Ji, Zongjie Li, Pingchuan Ma, Shuai Wang, Yingjiu Li, Yang Liu, Ning Liu, and Juergen Rahmel. {SelfDefend}:{LLMs} can defend themselves against jailbreaking in a practical manner. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 2441–2460, 2025.

[37] Xiaoyu Zhang, Cen Zhang, Tianlin Li, Yihao Huang, Xiaojun Jia, Ming Hu, Jie Zhang, Yang Liu, Shiqing Ma, and Chao Shen. Jailguard: A universal detection framework for prompt-based attacks on llm systems. *ACM Transactions on Software Engineering and Methodology*, 35(1):1–40, 2025.

[38] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, et al. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Advances in Neural Information Processing Systems*, 37:55005–55029, 2024.