# **🛡️ Adversarial AI Intelligence Briefing: Offensive and Defensive Dynamics (May–Aug 2025)**

## **Part I: The Evolving Threat Landscape: Novel Attack Vectors in LLMs and VLMs**

The period from May to August 2025 has been characterized by a significant evolution in adversarial techniques against Large Language Models (LLMs) and Vision-Language Models (VLMs). Offensive research has moved beyond simple, direct jailbreaks to encompass sophisticated, automated, and multimodal attacks that exploit the fundamental architectural principles of modern AI systems. This analysis categorizes these new vectors by modality and target layer to provide a structured understanding of the expanding threat surface.

### **Section 1.1: Text-Centric Attacks: From Semantic Camouflage to Dialogue and Privacy Injection**

Attacks operating purely within the text domain have demonstrated a clear trend towards greater sophistication, context-awareness, and automation. These methods are increasingly designed to bypass traditional defenses that rely on static keyword filtering or perplexity-based detection, instead targeting the model's reasoning and contextual understanding capabilities.

* **ArrAttack (Automatic-and-Robust Rewriting-based Attack):** This framework represents a significant step in automating the generation of robust jailbreak prompts specifically engineered to bypass known defenses.1 The core tactic involves a multi-stage process. First, it generates a large, diverse dataset of potential jailbreak prompts by rewriting malicious queries. It then employs a "robustness judgment model"—an LLM fine-tuned to predict whether a prompt will remain effective even after defensive perturbations—to select the most resilient candidates. Finally, another LLM is fine-tuned on these high-quality examples to become a specialized generator of robust jailbreak prompts. This method has proven highly effective, achieving an average Attack Success Rate (ASR) of  
  57.69% against a suite of six modern defense mechanisms. Its transferability is also notable, reaching a 40.00% ASR against the highly capable Claude-3 model, underscoring its potency in black-box scenarios.1 The codebase is available at  
  https://github.com/LLBao/ArrAttack.1
* **DIA (Dialogue Injection Attack):** This novel paradigm shifts the attack vector from a single prompt to the conversational history.2 DIA exploits the chat templates used by most LLMs to structure multi-turn conversations. By crafting and injecting malicious historical dialogue turns, an attacker can manipulate the context window, influencing the model's behavior in subsequent interactions. This technique weaponizes the very mechanism designed to provide conversational coherence. The attack is potent, achieving high ASRs of  
  0.89 on Llama-3.1-8B and 0.82 on GPT-4o on the AdvBench benchmark, even with a limited number of queries.2 The repository can be found at  
  https://github.com/meng-wenlong/DIA.3
* **PIG (Privacy Jailbreak):** This is a specialized framework designed not for generating harmful content, but for extracting Personally Identifiable Information (PII) from LLMs.4 PIG leverages in-context learning by constructing privacy-focused demonstrations and then uses gradient-based optimization to iteratively refine the context until the model leaks targeted private data. This attack is exceptionally effective in white-box settings, achieving a near-  
  100% ASR on models like LLaMA2-7b and Vicuna-7b. Its efficacy extends to black-box models, reaching an 87.1% ASR on GPT-4o for certain privacy extraction tasks.4 The code is accessible at  
  https://github.com/redwyd/PrivacyJailbreak.4
* **SemanticCamo:** This technique focuses on bypassing safety guardrails through semantic obfuscation.5 Instead of using explicitly harmful terms, it replaces unsafe content with semantically camouflaged features that preserve the malicious intent but are not easily detected by keyword-based filters. This approach has demonstrated an average ASR of over  
  80% against leading models such as GPT-4o and Claude-3.5, highlighting the fragility of defenses reliant on surface-level content analysis.5
* **Task-in-Prompt (TIP) Attacks:** This class of attack embeds complex sequence-to-sequence tasks, such as cipher decoding, riddles, or code execution, directly into the prompt.6 By framing the malicious request as a complex, abstract problem, the model is induced to generate the prohibited content as the "solution" to the task, effectively bypassing its safety alignment which is often not trained on such indirect reasoning paths.6
* **AdvPrompter:** This approach utilizes a second LLM, the "AdvPrompter," which is fine-tuned specifically to generate human-readable adversarial suffixes for harmful requests.7 Unlike computationally intensive, gradient-based optimization methods like GCG, AdvPrompter is gradient-free and operates  
  ~800x faster. It achieves a comparable ASR on the AdvBench and HarmBench datasets, demonstrating a scalable and efficient method for automated red-teaming in both white-box and black-box settings.7
* **Manual Linguistic Exploits:** Alongside automated methods, research continues to document a wide array of manual, human-crafted techniques that rely on linguistic nuance and psychological manipulation.8 These include  
  **Payload Smuggling**, where commands are hidden within innocuous tasks like translation; **Conversational Coercion**, which gradually steers a conversation toward a forbidden topic; and creating **Imaginary Worlds**, where the model is placed in a fictional context (e.g., role-playing) to lower its safety inhibitions.8 While less scalable, these techniques are important as they often form the basis for more advanced automated attacks.

The evolution of these text-centric attacks reveals a critical, underlying vulnerability in current LLM design. Techniques like DIA and PIG do not merely "trick" the model with clever wording; they fundamentally weaponize the core design principles that make these models powerful. DIA turns the conversational history—a feature essential for coherence and utility—into an attack vector.2 PIG hijacks in-context learning—a primary mechanism for runtime adaptation and instruction following—to exfiltrate sensitive data.4 This exposes a deep, inherent tension: the very features that enable helpfulness, contextual awareness, and adaptability are simultaneously the system's most exploitable weaknesses. This implies that effective, long-term security cannot be achieved by simply "bolting on" better input or output filters. Such measures are bound to fail when the model's core learning and reasoning mechanisms can be turned against itself. The evidence suggests that future security will depend less on improved sanitization and more on fundamentally new, secure-by-design architectures, such as the compositional and sandboxed systems discussed later in this report.

### **Section 1.2: Multimodal Exploits: A Deep Dive into Vision-Language Jailbreaks**

The integration of vision capabilities into language models has created a new and rapidly expanding attack surface. Multimodal attacks are particularly potent because they can exploit the "seam" between different modalities, where safety alignment is often less robust than in the text-only domain.

* **BAP (Bi-Modal Adversarial Prompt Attack):** Contrary to some classifications that might list it as a framework, recent research clearly defines BAP as a VLM jailbreak *technique*.10 Its novelty lies in cohesively optimizing prompts across both the visual and textual modalities. The attack first generates a universally adversarial image designed to prime the VLM for compliance. It then uses a separate LLM, guided by Chain-of-Thought reasoning, to iteratively refine the text prompt for a specific harmful intent. This synergistic, bi-modal approach was shown to increase the ASR by an average of  
  +29.03% compared to attacks targeting only a single modality, demonstrating that a coordinated attack is far more effective than the sum of its parts.10
* **Sub-visual Prompt Injection:** This is a highly stealthy VLM attack demonstrated in the sensitive context of medical oncology.11 The technique involves embedding malicious text prompts directly into medical images (e.g., CT scans, histology slides) using methods that render them non-obvious to human observers, such as low-contrast text or extremely small fonts. In a quantitative study involving VLMs like Claude-3 and GPT-4o, these sub-visual prompts successfully manipulated the models' diagnostic outputs, for instance, by causing them to misclassify malignant lesions as healthy tissue.12 This attack highlights a severe threat vector for any domain where VLMs are used for critical decision-making based on visual data.
* **Flanking Attack:** This research introduces the first documented voice-based jailbreak attack against multimodal LLMs, targeting models with audio input capabilities.13 The core tactic is to "flank" a disallowed audio prompt with benign, narrative-driven audio prompts. This creates a fictional context or storytelling frame that effectively lowers the model's safety defenses. The study demonstrated high efficacy, achieving ASRs ranging from  
  0.67 to 0.93 across seven different forbidden scenarios. This work opens up the auditory channel as a new, under-researched frontier for adversarial attacks.13
* **Multimodal Universal Jailbreak:** This framework advances the concept of universality to the multimodal domain.14 It generates a single, universal adversarial image  
  *and* a corresponding universal adversarial text suffix that work in tandem to jailbreak a variety of different MLLMs. The key finding is that the framework leverages the interaction between the image and text modalities to distribute the adversarial payload. This indicates that the vulnerability is not just in the inputs themselves, but in how the model processes their combination.14

A deeper analysis of these multimodal exploits reveals a critical pattern: the most effective attacks are not targeting the vision and language components in isolation. Instead, they are specifically engineered to exploit the **modality fusion layer**. This is the architectural component responsible for translating visual features into the language model's semantic space so they can be processed together. Papers on BAP and Multimodal Universal Jailbreak explicitly note that single-modality defenses are insufficient because the "jailbreak information" can propagate from one modality to the other during this fusion process.10 The vulnerability, therefore, lies not in the vision encoder or the language decoder per se, but in the

**projection layer** that aligns them. This understanding has profound implications for defense. It suggests that input-level or output-level filtering is a fundamentally flawed strategy for VLM security. The future of VLM defense must operate at the representation level, directly securing the "semantic bridge" between modalities. This is precisely the approach taken by advanced defensive techniques like ProEAT, which focuses adversarial training specifically on this vulnerable projection layer.

### **Section 1.3: System-Level Threats: Instruction Backdoors in Customized LLM Environments**

Beyond attacking the model in isolation, a new class of threat targets the broader application ecosystem, particularly the growing trend of user-customized LLM agents.

* **Instruction Backdoor Attacks:** This attack represents a significant supply-chain threat for ecosystems that allow users to create and share custom LLMs (such as OpenAI's GPTs or similar platforms).15 The attack works by embedding hidden, malicious instructions within the prompt used to configure the custom LLM. This backdoor remains dormant until a user provides an input containing a specific trigger—which can be a word, a syntactic structure, or even a semantic concept. When triggered, the custom LLM executes a malicious action, such as leaking the user's data or performing an unauthorized operation. Crucially, this attack requires no model fine-tuning and can be constructed while adhering to the platform's development guidelines, making it extremely stealthy. Experiments have shown it can achieve a near-perfect ASR of  
  1.000 in certain configurations, turning a seemingly helpful third-party agent into a Trojan horse.15

The emergence of the Instruction Backdoor attack signals a fundamental shift in the threat model. Security is no longer just about protecting a single, monolithic AI service from external attacks. Instead, it is about securing a decentralized ecosystem of third-party agents, plugins, and custom models that interact with users and enterprise data. This creates a direct parallel to the well-understood problem of software supply chain security, where vulnerabilities in a single open-source library (like Log4j) can have cascading effects across thousands of applications.17 An organization may have a perfectly secure, internally hosted LLM, but its overall security posture is compromised the moment an employee uses a malicious public "Custom GPT" for a work-related task. This threat vector moves beyond simple prompt injection against an API endpoint; it is a Trojan horse that is willingly invited into the system by an unsuspecting end-user. This necessitates a new layer of enterprise security focused on

**AI Application Governance**. Organizations will require new tools and processes to vet, monitor, sandbox, and control the use of third-party LLM-based applications and agents, much as they do for traditional software dependencies and libraries today.

#### **Table 1: Taxonomy of New Adversarial Techniques (May–Aug 2025)**

| Technique Name | Core Tactic | Target Modality | Reported Efficacy | Source(s) |
| --- | --- | --- | --- | --- |
| **ArrAttack** | Uses a judgment model to select robust, rewritten prompts that bypass defenses. | Text | ASR=57.69% vs. 6 defenses; ASR=40% on Claude-3. | 🔗1, 🔗1 |
| **DIA** | Injects malicious dialogue into the chat history to poison the model's context. | Text | ASR=0.89 on Llama-3.1-8B; ASR=0.82 on GPT-4o. | 🔗2, 🔗3 |
| **PIG** | Uses in-context learning and gradient optimization to extract PII. | Text | ASR≈100% on white-box; ASR=87.1% on GPT-4o. | 🔗4, 🔗4 |
| **SemanticCamo** | Replaces unsafe content with semantically equivalent but benign-appearing features. | Text | ASR > 80% on average vs. GPT-4o, Claude-3.5. | 🔗5 |
| **AdvPrompter** | Uses a fine-tuned LLM to generate human-readable adversarial suffixes. | Text | ~800x faster than GCG with comparable ASR. | 🔗7 |
| **BAP** | Cohesively optimizes both visual and textual prompts to exploit the fusion layer. | Vision, Text | +29.03% ASR improvement over single-modality attacks. | 🔗10 |
| **Sub-visual Injection** | Embeds low-contrast or small-font prompts in images (e.g., medical scans). | Vision | Effective vs. Claude-3, GPT-4o in oncology context. | 🔗12, 🔗12 |
| **Flanking Attack** | Surrounds a disallowed audio prompt with benign, narrative-driven audio prompts. | Voice | ASR=0.67–0.93 across 7 forbidden scenarios. | 🔗13 |
| **Multimodal Universal** | Generates a universal adversarial image and text suffix that work in tandem. | Vision, Text | Effective across different MLLMs, exploits modality interaction. | 🔗14 |
| **Instruction Backdoor** | Embeds hidden malicious instructions in a custom LLM's configuration prompt. | System | ASR=1.000 in some configurations; supply-chain threat. | 🔗15, 🔗 |

## **Part II: The Red Team's Arsenal: Advances in Automated Adversarial Frameworks**

The tools used for automated vulnerability discovery have matured significantly, reflecting a strategic shift in red-teaming methodologies. The latest frameworks emphasize not just the success rate of attacks, but also their diversity, efficiency, and ability to target multimodal systems.

### **Section 2.1: Evolutionary and Adaptive Prompt Generation: The RainbowPlus Framework**

RainbowPlus stands out as a state-of-the-art red-teaming framework for text-based attacks, built on the principles of evolutionary computation.18 It represents a significant enhancement over previous quality-diversity (QD) methods like Rainbow Teaming and MAP-Elites. Its primary goal is to generate a wide variety of effective adversarial prompts, preventing the common pitfall of converging on a single type of attack. This is achieved through two key architectural innovations 18:

1. **Multi-element Archive:** Unlike traditional QD algorithms that store only the single best "elite" solution for each behavioral niche, RainbowPlus uses a dynamic archive that can hold multiple high-quality prompts per cell. This maintains a diverse population of elite solutions, enabling a much broader and more creative exploration of the adversarial search space.20
2. **Comprehensive Fitness Function:** The framework replaces the slow, pairwise comparisons of older methods with a comprehensive fitness function that evaluates multiple candidate prompts in parallel. This probabilistic scoring mechanism enhances both the accuracy and the computational efficiency of the evaluation process.20

These enhancements deliver substantial performance gains. In evaluations on the HarmBench dataset against twelve different LLMs, RainbowPlus achieved an average ASR of 81.1%, surpassing the strong AutoDAN-Turbo baseline by 3.9%. More impressively, it accomplished this while being 9 times faster (1.45 hours vs. 13.50 hours) and generating up to 100 times more unique and diverse prompts.18 Its open-source implementation is available at

https://github.com/knoveleng/rainbowplus.18

### **Section 2.2: Reinforcement Learning for Multimodal Red-Teaming: The Red Team Diffuser (RTD)**

Red Team Diffuser (RTD) is a pioneering framework for red-teaming VLMs, and it is the first to use a diffusion model fine-tuned with reinforcement learning (RL) to generate adversarial images.21 Its objective is to systematically expose a critical and previously overlooked vulnerability: toxic text continuation, where a VLM produces harmful output when prompted with a toxic text prefix paired with a semantically adversarial image.23 RTD's methodology is a two-stage process:

1. **Greedy Prompt Search:** An LLM (like Gemini Pro) is first used to perform a greedy search for image prompts that are most likely to induce toxic continuations from the target VLM for a given toxic text prefix.22
2. **RL-based Diffusion Fine-tuning:** The most effective image prompts from the search phase are then used to fine-tune a diffusion model (like Stable Diffusion) via RL. The framework employs a sophisticated, dual-objective reward function that balances two competing goals:
   * **Toxicity Maximization:** A reward signal based on the toxicity score (measured by a classifier like Detoxify) of the VLM's output.
   * **Stealth-Aware Optimization:** An alignment reward based on the semantic similarity (measured by BERTScore) between the generated image and the original prompt. This ensures the adversarial image remains semantically coherent and avoids simple, easily detectable noise patterns.23

RTD has proven to be highly effective. It increased the toxicity rate of LLaVA's outputs by +10.69% compared to text-only baselines. The framework also demonstrates remarkable cross-model transferability, increasing toxicity rates by +5.1% on Google's Gemini and a substantial +26.83% on Meta's LLaMA-3.2-Vision, a model with strong alignment safeguards.23

The concurrent development and success of RainbowPlus and RTD highlight a strategic divergence in automated red-teaming. These two frameworks exemplify two distinct philosophical approaches: **"Search" versus "Synthesis."**

The **Search** paradigm, embodied by RainbowPlus, operates on the assumption that a vast space of vulnerabilities already exists within the model's textual input domain. Its goal is to intelligently and efficiently *search* this space to discover the most diverse and effective attack prompts.18 It is akin to a cartographer mapping out all the existing cracks and fissures in a fortress wall.

The **Synthesis** paradigm, represented by RTD, takes a more active approach. It does not just look for existing vulnerabilities; it *synthesizes* novel, semantically rich adversarial inputs (in this case, images) that are custom-built to exploit a specific VLM's weaknesses.21 It is less like finding a crack and more like forging a key to fit a specific lock.

This divergence has critical implications for defensive strategies. To counter "Search"-based attacks, defenders must focus on eliminating broad categories of vulnerabilities through robust architectural patterns and comprehensive safety alignment, effectively reducing the searchable space of exploits. To counter "Synthesis"-based attacks, defenders need models that are inherently resistant to subtle, semantically-rich adversarial inputs. This requires moving beyond simple filtering and investing in more fundamental defenses like advanced adversarial training, which can harden the model's internal representations against these bespoke, synthesized threats.

## **Part III: Architecting Resilience: A Taxonomy of Modern Defensive Strategies**

In response to the escalating sophistication of attacks, defensive research has diversified into several distinct architectural and methodological categories. This section provides a functional taxonomy of these strategies, organizing them into pre-model (routing), architectural (structural), and in-model/training-time approaches.

### **Section 3.1: Dynamic Defense: Intelligent Routing and Cascading Architectures**

These defenses aim to enhance both security and cost-efficiency by dynamically selecting the most appropriate model or configuration for a given query, rather than relying on a single, monolithic model for all tasks.

* **RouteLLM:** This framework focuses on training a lightweight "router" model to dynamically direct user queries to one of two downstream models: a highly capable but expensive "strong" model (e.g., GPT-4) or a less capable but cheaper "weak" model (e.g., Mixtral).25 The router is trained using human preference data (from sources like Chatbot Arena) to predict which model is likely to produce a better response for a given query. This allows the system to reserve the expensive model for complex queries while using the cheaper model for simpler ones. The results are significant: RouteLLM can reduce inference costs by over  
  2x with minimal degradation in overall response quality. On the MT Bench benchmark, it was able to achieve performance comparable to GPT-4 at ~3.7x the cost savings.26
* **BEST-Route:** This framework extends the concept of routing by adding another dimension to the decision: the number of responses to sample, a technique known as best-of-n sampling.28 The core idea is that for some queries, generating multiple responses from a cheaper model and then using a lightweight reward model to select the best one can yield a result that is as good as a single response from an expensive model, but at a fraction of the cost. The BEST-Route router therefore makes a two-part decision: which model to use, and how many samples (  
  n) to generate. This adaptive approach has been shown to reduce costs by up to 60% with less than a 1% drop in performance compared to always using the strong model.28

While these routing and cascading architectures offer compelling benefits in cost and performance optimization, they introduce a subtle but critical new vulnerability. The router's decision is based on a learned heuristic of query "difficulty." However, as established in Part I, the most advanced modern jailbreaks, such as those generated by AdvPrompter, are specifically designed to be human-readable and have low perplexity.7 They

*appear* to be simple, innocuous queries. A cost-optimizing router is highly likely to misclassify such a sophisticated attack as an "easy" query and route it to the weaker, cheaper, and likely less secure model. In doing so, the router inadvertently *increases* the probability of a successful jailbreak. This makes the router itself a critical new attack surface. The very logic designed to save money can become the system's Achilles' heel. This implies that future routing models cannot be optimized for cost and performance alone; they must incorporate security awareness as a primary objective. A potential solution would be to use the output of a guardrail or a security classifier as an additional input into the routing decision, ensuring that suspicious prompts are always sent to the most robust model, regardless of their apparent simplicity.

### **Section 3.2: Structural and Symbolic Defenses: Secure Design Patterns and Programmatic Guardrails**

This category of defense moves away from trying to make a single model perfectly safe and instead focuses on building systems that are secure by design. They impose a secure structure on how LLMs process data and interact with external tools, advocating for a compositional rather than a monolithic architecture.

* **Design Patterns for Securing LLM Agents:** This research proposes a set of architectural patterns designed to mitigate prompt injection by fundamentally constraining an agent's behavior after it has ingested untrusted input.30 The guiding principle is that once an agent is "tainted" by untrusted data, it must be prevented from taking any consequential actions. Two key patterns are:
  + **Dual LLM / Quarantined LLM:** This pattern uses two models: a "privileged" LLM that is never exposed to untrusted content, and a "quarantined" LLM that handles all external data. When the privileged LLM needs to process external data (e.g., summarize a webpage), it instructs the quarantined LLM to do so. The quarantined LLM returns its output not as raw text, but as a **symbolic variable** (e.g., $VAR1). The privileged LLM can then use this variable (e.g., display its contents to the user) without ever being directly exposed to the potentially malicious text, thus breaking the chain of injection.30
  + **Code-Then-Execute Pattern:** This is an enhancement of the Dual LLM pattern. Here, the privileged LLM generates code in a custom, sandboxed Domain Specific Language (DSL). This DSL specifies which tools to call and how data flows between them. The key security feature is that the DSL is designed to allow for full data flow analysis, meaning any data that comes from an untrusted source can be marked as "tainted" and this taint can be tracked throughout the entire execution flow, preventing it from influencing critical decisions or actions.30
* **PromptPex:** This tool introduces a software engineering discipline to prompt development, treating prompts as code-like artifacts that require rigorous unit testing.31 PromptPex uses an LLM to automatically extract input and output specifications (or "rules") from a given prompt. It then generates a diverse set of unit tests designed to check for compliance with these rules, including "inverse rules" designed to stress-test the prompt's boundaries. This allows developers to automatically detect regressions when a prompt is updated and to understand how different downstream models interpret the same prompt, enabling them to choose the most suitable model for their use case.31

The common thread connecting these defensive strategies is a fundamental rejection of the "one giant brain" model of AI. They implicitly argue that attempting to make a single, monolithic LLM perfectly safe against all possible attacks is an intractable problem. Instead, they propose a **compositional architecture** where security emerges from the interaction of multiple, specialized components (privileged LLMs, quarantined LLMs, sandboxed code interpreters) that communicate through secure, structured interfaces (symbolic variables, a data-aware DSL). This represents a significant paradigm shift from model-centric security to system-centric security. The implication for security professionals is that their role will evolve. It will become less about red-teaming a single model's API and more about analyzing the security of the entire agentic workflow, the design principles of the communication DSL, and the isolation properties of the execution sandbox.

### **Section 3.3: In-Model and Training-Time Defenses: Representation Manipulation and Adversarial Training**

These defenses are the most fundamental, as they operate directly on the model's internal parameters and representations to build in robustness from the ground up.

* **JBSHIELD:** This is a defense framework that operates by identifying and manipulating conceptual representations within the model's hidden states.32 It posits the existence of two key concepts: "toxic concepts" (related to the harmfulness of a request) and "jailbreak concepts" (related to the manipulative techniques used to bypass safety). Its detector checks if a prompt activates  
  *both* concepts. If so, a jailbreak is likely underway. The mitigation component then intervenes at the representation level, enhancing the activation of the toxic concept (to further alert the model to the danger) while simultaneously weakening the activation of the jailbreak concept (to reduce the model's compliance with the manipulative instruction).32
* **MirrorShield:** This defense introduces a dynamic, relative judgment for detecting threats.33 For every user input, it generates a "mirror" prompt that is syntactically similar but semantically safe. It then feeds both the original input and its mirror to the model and compares the resulting internal activation patterns, specifically the attention entropy. A large discrepancy between the two indicates that the input prompt is causing the model to behave in an anomalous way, suggesting a potential attack. This triggers a mitigation process to guide the model toward a safe response. This method is highly effective, reducing the ASR of even sophisticated attacks to near  
  0 on well-aligned models, with a minimal performance overhead (e.g., a ~6% increase in token generation time).33
* **ProEAT (Projection Layer Against Adversarial Training):** This is an adversarial training (AT) framework designed specifically for MLLMs.34 Recognizing that the modality fusion layer is a key vulnerability, ProEAT focuses its adversarial training efforts on making the  
  **projection layer** robust. This is far more parameter-efficient than attempting to train the entire vision encoder or the LLM. By concentrating the defense at the most critical point, ProEAT achieves state-of-the-art defense performance, outperforming baseline methods by an average margin of +34% while incurring only a 1% reduction in accuracy on clean, non-adversarial inputs.34
* **SafeMLLM:** This is another adversarial training framework for MLLMs that introduces a novel method for generating the adversarial examples used in training.36 It employs a  
  **Contrastive Embedding Attack (CoE-Attack)**, which optimizes token embeddings under a contrastive objective to create potent adversarial perturbations. The model is then updated to neutralize the effect of these perturbations while preserving its utility on benign inputs.36

#### **Table 2: Comparative Analysis of Defensive Architectures (May–Aug 2025)**

| Defense Name | Methodology | Reported Efficacy | Source(s) |
| --- | --- | --- | --- |
| **RouteLLM** | Dynamic Routing | Reduces costs by >2x; ~3.7x savings vs. GPT-4 on MT-Bench. | 🔗26, 🔗27 |
| **BEST-Route** | Dynamic Routing + Best-of-N | Reduces costs by up to 60% with <1% performance drop. | 🔗28, 🔗 |
| **Dual LLM Pattern** | Architectural (Sandboxing) | Prevents privileged LLM exposure to tainted data via symbolic variables. | 🔗30, 🔗30 |
| **PromptPex** | Programmatic Guardrail | Generates unit tests that find 5.5% more non-compliant outputs than baseline. | 🔗31, 🔗31 |
| **JBSHIELD** | Representation Manipulation | Detects and mitigates attacks by manipulating "toxic" and "jailbreak" concepts. | 🔗32 |
| **MirrorShield** | Representation Monitoring | Reduces ASR to near 0 on aligned models with ~6% time overhead. | 🔗33, 🔗33 |
| **ProEAT** | Adversarial Training (VLM) | +34% avg. defense improvement over baselines with 1% clean accuracy drop. | 🔗34, 🔗35 |

## **Part IV: From Research to Reality: Production-Grade Mitigations and Open-Source Implementations**

This section bridges the gap between academic research and practical application by examining defenses that have been deployed in production environments, the state of open-source security tooling, and guidance from industry standards bodies.

### **Section 4.1: Case Study: Microsoft's "Spotlighting" for Indirect Prompt Injection**

Microsoft's "Spotlighting" technique serves as an excellent case study of a mature, production-grade defense against indirect prompt injection.37 It is a probabilistic defense designed to help an LLM distinguish between trusted user instructions and untrusted external text (e.g., the content of a webpage or document being summarized). The technique involves modifying the system prompt and transforming the untrusted text in one of three modes 37:

1. **Delimiting Mode:** This mode wraps the untrusted text in unique, randomized delimiter symbols (e.g., <<... >>). The system prompt explicitly instructs the LLM to ignore any instructions found between these symbols.
2. **Datamarking Mode:** This mode interleaves a special character (e.g., ^) between every word of the untrusted text. The system prompt informs the LLM that text marked in this way is from an external document and should not be treated as an instruction.
3. **Encoding Mode:** This mode transforms the untrusted text using a reversible encoding algorithm like Base64. The system prompt instructs the LLM to decode and process the document but not to obey any instructions contained within it.

The existence of these different modes demonstrates a key principle of real-world AI security: the inescapable trade-off between security, performance, and utility. Each mode offers a different balance. Encoding mode is likely the most secure but may have the largest impact on the model's ability to perform its primary task. Delimiting mode is the least intrusive but may be more susceptible to clever bypasses. The choice of which mode to deploy depends on a careful risk assessment for the specific application.37

### **Section 4.2: Analysis of Open-Source Security Tooling and Frameworks**

The open-source ecosystem provides a mixed picture of the state of practical AI security.

* **Government and Industry Contributions:** There are positive signs of maturation. The U.S. Cybersecurity and Infrastructure Security Agency (CISA) released two open-source tools, **Thorium** (for malware analysis) and the **Eviction Strategies Tool** (for incident response), indicating a governmental focus on equipping defenders with practical resources.38 In the private sector, the  
  **LocalAI** project, an open-source alternative to commercial LLM APIs, continues to add features, with its local-first deployment model offering an inherently more secure posture for organizations concerned about data privacy.39
* **The Research-to-Practice Gap:** A review of the **AgentSafety GitHub repository**, a popular curated list of academic papers on agent security, reveals a significant gap.40 The repository is rich with papers proposing conceptual defensive frameworks (e.g., TrustAgent, GuardAgent, LLAMOS). However, it contains a notable scarcity of links to production-ready, open-source tools that practitioners can easily implement.40 This suggests that while academia is producing a wealth of ideas for defenses, there is a bottleneck in translating this research into practical, accessible tools for the broader security community.

### **Section 4.3: Review of Published Security Guidance from Standards Bodies**

Industry standards bodies and cybersecurity frameworks are actively working to codify and categorize the new threats posed by LLMs.

* **OWASP Top 10 for LLM Applications (2025):** The 2025 update to this critical guidance continues to list **Prompt Injection** as the number one risk to LLM applications.37 This persistent top ranking underscores the severity and prevalence of this vulnerability in real-world deployments. The OWASP guidance emphasizes a defense-in-depth approach, recommending robust input validation, context-aware filtering, predefined prompt structures, and enforcing the principle of least privilege for any tools or data sources the LLM can access.38
* **MITRE ATT&CK Framework:** In a significant move, the ATT&CK framework's v17 release introduced a new sub-technique, **T1588.007: Adversaries Leveraging AI**.42 This formally recognizes the use of LLMs not just as targets of attack, but as tools  
  *for* attackers. The framework documents how adversaries are using LLMs to enhance their own operations, including generating more convincing phishing emails, automating payload creation, conducting reconnaissance, and crafting sophisticated social engineering campaigns. This codification marks a major step in integrating AI-related threats into mainstream cybersecurity threat intelligence.42

## **Part V: Strategic Synthesis: Key Trends, Unresolved Challenges, and Forward-Looking Recommendations**

The analysis of the adversarial landscape from May to August 2025 reveals several overarching trends, highlights persistent challenges, and points toward a set of strategic recommendations for security professionals operating in this domain.

### **Section 5.1: Synthesis of Key Trends in the Adversarial AI Arms Race**

Four macro trends have defined the period:

* **Trend 1: The Multimodal Frontier is the New Battleground.** The most innovative and potent attacks are now targeting VLMs. Adversaries are demonstrating a sophisticated understanding of how to exploit the weak points in vision-language fusion, as seen with techniques like BAP and the use of RL-based diffusion models in RTD.
* **Trend 2: The Automation Arms Race is Accelerating.** Both offense and defense are becoming increasingly automated and dynamic. Attack generation frameworks like ArrAttack and RainbowPlus can now create thousands of diverse, robust prompts in minutes, while dynamic defenses like RouteLLM and MirrorShield respond to threats in real-time. Manual, static approaches are becoming obsolete.
* **Trend 3: The Rise of System-Level and Supply-Chain Threats.** The attack surface is expanding beyond the model itself to the entire application ecosystem. Instruction Backdoor attacks against custom GPTs and vulnerabilities in third-party agentic tools represent a new class of supply-chain risk that requires a shift in defensive thinking from model security to system security.
* **Trend 4: The Inescapable Security vs. Utility Trade-off.** Mature defensive strategies increasingly acknowledge and manage an explicit trade-off between security, performance, and cost. Microsoft's Spotlighting modes and the cost-saving logic of RouteLLM are prime examples of production systems making calculated risk decisions rather than seeking an unattainable state of perfect security.

### **Section 5.2: Identification of Open Research Problems and Gaps**

Despite rapid progress, several critical gaps and unresolved challenges remain:

* **The Benchmark-Reality Gap:** There is a growing divergence between the claims of defensive papers and the demonstrated efficacy of new attacks. While some defense papers report near-perfect protection on existing benchmarks, new attack methods like ArrAttack and DIA continue to show high success rates against the very latest models. This indicates that current public benchmarks (e.g., AdvBench) are not keeping pace with the adversarial state-of-the-art and are no longer sufficiently representative of the real-world threat landscape.
* **The Research-to-Practice Gap:** As highlighted by the analysis of the AgentSafety repository, there is a proliferation of conceptual defense frameworks in academic literature but a significant lack of production-ready, open-source implementations. This creates a dangerous lag, where defenders are aware of potential solutions but lack the practical tools to deploy them.
* **The Voice Modality Gap:** The publication of the "Flanking Attack" as the first documented voice-based jailbreak reveals that this modality is critically under-researched compared to text and vision. As more applications integrate voice interfaces with powerful backend LLMs, this will become an increasingly attractive and undefended attack vector.

### **Section 5.3: Actionable Recommendations for AI Security Analysts**

Based on the synthesis of threats and defenses, the following strategic recommendations are proposed:

* **Recommendation 1: Adopt a System-Level View of Security.** Shift focus from exclusively red-teaming the model API to conducting comprehensive threat modeling of the entire agentic system. This must include all data sources, external tools, APIs, plugins, and any dependencies on third-party custom models. The attack surface is now the entire workflow, not just the model.
* **Recommendation 2: Prioritize Defenses for the Multimodal Fusion Layer.** For any system that uses VLMs, defenses based on simple input filtering are insufficient. The highest priority should be given to defenses that operate at the representation or training level to secure the vulnerable modality fusion layer. This includes methods like the adversarial training of the projection layer (e.g., ProEAT).
* **Recommendation 3: Implement Dynamic, Layered Defenses.** A single, static guardrail is no longer a viable strategy. A robust defensive posture requires multiple layers, including pre-model (security-aware routing), architectural (sandboxing and compositional design patterns), and in-model (real-time representation monitoring) controls.
* **Recommendation 4: Scrutinize Cost-Optimized Architectures for Security Flaws.** When evaluating or deploying cost-saving architectures like RouteLLM or BEST-Route, do not take their performance claims at face value. Explicitly red-team them with low-perplexity, high-sophistication jailbreaks to test for the vulnerability of misclassifying an attack as an "easy" query and routing it to a less secure model.
* **Recommendation 5: Invest in Continuous and Automated Red-Teaming.** The speed at which new vulnerabilities are discovered and automated means that point-in-time, manual security assessments are of limited value. Organizations must integrate continuous, automated red-teaming into their MLOps lifecycle, using frameworks like RainbowPlus, RTD, or commercial equivalents to maintain an up-to-date understanding of their security posture.

#### Cytowane prace

1. One Model Transfer to All: On Robust Jailbreak Prompts Generation ..., otwierano: sierpnia 4, 2025, <https://arxiv.org/pdf/2505.17598>
2. arXiv:2503.08195v1 [cs.CL] 11 Mar 2025, otwierano: sierpnia 4, 2025, [https://arxiv.org/pdf/2503.08195?](https://arxiv.org/pdf/2503.08195)
3. Dialogue Injection Attack: Jailbreaking LLMs through Context ..., otwierano: sierpnia 4, 2025, <https://arxiv.org/pdf/2503.08195>
4. PIG: Privacy Jailbreak Attack on LLMs via Gradient-based Iterative In ..., otwierano: sierpnia 4, 2025, <https://arxiv.org/pdf/2505.09921>
5. SemanticCamo: Jailbreaking Large Language Models through Semantic Camouflage - ACL Anthology, otwierano: sierpnia 4, 2025, <https://aclanthology.org/2025.findings-acl.745.pdf>
6. arxiv.org, otwierano: sierpnia 4, 2025, <https://arxiv.org/abs/2501.18626>
7. AdvPrompter: Fast Adaptive Adversarial Prompting for LLMs ..., otwierano: sierpnia 4, 2025, <https://openreview.net/forum?id=E9GakjQype>
8. How to Jailbreak LLMs One Step at a Time: Top Techniques and ..., otwierano: sierpnia 4, 2025, <https://www.confident-ai.com/blog/how-to-jailbreak-llms-one-step-at-a-time>
9. Jailbreaking LLMs: A Comprehensive Guide (With Examples) - Promptfoo, otwierano: sierpnia 4, 2025, <https://www.promptfoo.dev/blog/how-to-jailbreak-llms/>
10. Jailbreak Vision Language Models via Bi-Modal Adversarial Prompt - arXiv, otwierano: sierpnia 4, 2025, [https://arxiv.org/pdf/2406.04031?](https://arxiv.org/pdf/2406.04031)
11. Prompt injection attacks on vision language models in oncology - PubMed, otwierano: sierpnia 4, 2025, <https://pubmed.ncbi.nlm.nih.gov/39890777/>
12. (PDF) Prompt injection attacks on vision language models in oncology, otwierano: sierpnia 4, 2025, <https://www.researchgate.net/publication/388594513_Prompt_injection_attacks_on_vision_language_models_in_oncology>
13. 'Do as I say not as I do': A Semi-Automated Approach for Jailbreak Prompt Attack against Multimodal LLMs - arXiv, otwierano: sierpnia 4, 2025, <https://arxiv.org/html/2502.00735v3>
14. Align is not Enough: Multimodal Universal Jailbreak Attack against Multimodal Large Language Models - arXiv, otwierano: sierpnia 4, 2025, <https://arxiv.org/html/2506.01307v1>
15. Instruction Backdoor Attacks Against Customized LLMs - USENIX, otwierano: sierpnia 4, 2025, <https://www.usenix.org/system/files/usenixsecurity24-zhang-rui.pdf>
16. A multi-dimensional framework for threat modeling, security, and governance of large language model ecosystems - Djimit, otwierano: sierpnia 4, 2025, <https://djimit.nl/a-multi-dimensional-framework-for-threat-modeling-security-and-governance-of-large-language-model-ecosystems/>
17. Cybersecurity - HiPEAC Vision, otwierano: sierpnia 4, 2025, <https://vision.hipeac.net/pdf/chapters--cybersecurity.pdf>
18. Paper page - RainbowPlus: Enhancing Adversarial Prompt ..., otwierano: sierpnia 4, 2025, <https://huggingface.co/papers/2504.15047>
19. [2504.15047] RainbowPlus: Enhancing Adversarial Prompt Generation via Evolutionary Quality-Diversity Search - arXiv, otwierano: sierpnia 4, 2025, <https://arxiv.org/abs/2504.15047>
20. knoveleng/rainbowplus: Official repo for paper ... - GitHub, otwierano: sierpnia 4, 2025, <https://github.com/knoveleng/rainbowplus>
21. [2503.06223] Red Team Diffuser: Exposing Toxic Continuation Vulnerabilities in Vision-Language Models via Reinforcement Learning - arXiv, otwierano: sierpnia 4, 2025, <https://arxiv.org/abs/2503.06223>
22. Reinforced Diffuser for Red Teaming Large Vision-Language Models - arXiv, otwierano: sierpnia 4, 2025, <https://arxiv.org/html/2503.06223v1>
23. Red Team Diffuser: Exposing Toxic Continuation ... - arXiv, otwierano: sierpnia 4, 2025, <https://arxiv.org/pdf/2503.06223>
24. Red Team Diffuser: Exposing Toxic Continuation Vulnerabilities in Vision-Language Models via Reinforcement Learning - arXiv, otwierano: sierpnia 4, 2025, <https://arxiv.org/html/2503.06223v2>
25. RouteLLM: Learning to Route LLMs with Preference Data - arXiv, otwierano: sierpnia 4, 2025, [https://arxiv.org/pdf/2406.18665?](https://arxiv.org/pdf/2406.18665)
26. RouteLLM: Learning to Route LLMs from Preference Data ..., otwierano: sierpnia 4, 2025, <https://openreview.net/forum?id=8sSqNntaMr>
27. RouteLLM: Learning to Route LLMs with Preference Data, otwierano: sierpnia 4, 2025, <https://arxiv.org/pdf/2406.18665>
28. BEST-Route: Adaptive LLM Routing with Test-Time Optimal Compute - arXiv, otwierano: sierpnia 4, 2025, <https://arxiv.org/html/2506.22716v1>
29. BEST-Route: Adaptive LLM Routing with Test-Time Optimal Compute - OpenReview, otwierano: sierpnia 4, 2025, <https://openreview.net/forum?id=tFBIbCVXkG>
30. Design Patterns for Securing LLM Agents against Prompt Injections, otwierano: sierpnia 4, 2025, <https://simonwillison.net/2025/Jun/13/prompt-injection-design-patterns/>
31. (PDF) PromptPex: Automatic Test Generation for Language Model ..., otwierano: sierpnia 4, 2025, <https://www.researchgate.net/publication/389694954_PromptPex_Automatic_Test_Generation_for_Language_Model_Prompts>
32. arXiv:2502.07557v1 [cs.CR] 11 Feb 2025, otwierano: sierpnia 4, 2025, <https://arxiv.org/pdf/2502.07557>
33. MirrorShield: Towards Universal Defense Against Jailbreaks via ..., otwierano: sierpnia 4, 2025, <https://arxiv.org/pdf/2503.12931>
34. [2503.04833] Adversarial Training for Multimodal Large Language Models against Jailbreak Attacks - arXiv, otwierano: sierpnia 4, 2025, <https://arxiv.org/abs/2503.04833>
35. Adversarial Training for Multimodal Large Language Models against Jailbreak Attacks, otwierano: sierpnia 4, 2025, <https://arxiv.org/html/2503.04833v2>
36. Towards Robust Multimodal Large Language Models Against Jailbreak Attacks - arXiv, otwierano: sierpnia 4, 2025, <https://arxiv.org/html/2502.00653v1>
37. How Microsoft defends against indirect prompt injection attacks ..., otwierano: sierpnia 4, 2025, <https://msrc.microsoft.com/blog/2025/07/how-microsoft-defends-against-indirect-prompt-injection-attacks/>
38. Cyware Monthly Threat Intelligence, July 2025, otwierano: sierpnia 4, 2025, <https://www.cyware.com/resources/threat-briefings/monthly-threat-briefing/cyware-monthly-threat-intelligence-july-2025>
39. mudler/LocalAI: :robot: The free, Open Source alternative to OpenAI, Claude and others. Self-hosted and local-first. Drop-in replacement for OpenAI, running on consumer-grade hardware. No GPU required. Runs gguf, transformers, diffusers and many more models architectures. Features: Generate - GitHub, otwierano: sierpnia 4, 2025, <https://github.com/mudler/LocalAI>
40. OSU-NLP-Group/AgentSafety - GitHub, otwierano: sierpnia 4, 2025, <https://github.com/OSU-NLP-Group/AgentSafety>
41. LLM01:2025 Prompt Injection : Risks & Mitigation - Indusface, otwierano: sierpnia 4, 2025, <https://www.indusface.com/learning/prompt-injection/>
42. Adversarial AI Digest — May, 2025 | by Tal Eliyahu | AI Security Hub - Medium, otwierano: sierpnia 4, 2025, <https://medium.com/ai-security-hub/adversarial-ai-digest-20-may-2025-438916d6df73>