
HYDRA

Feedback-Driven Adaptive Red-Teaming for Language Models

Vaishakh Vipin
Independent Researcher

Aarav Sharma
Independent Researcher

With
Apart Research

Abstract

HYDRA is an automated red-teaming framework for safety-trained language models that uses an **adaptive attacker** rather than static jailbreak prompts. The core problem it addresses is that single prompts are often patched quickly and do not reflect how real attackers iteratively improve after failure. HYDRA treats this as a **feedback-driven learning loop** powered by **gradient-guided beam search**. In each round, the attacker generates diverse prompt branches, the victim model is queried concurrently, a judge scores compliance, and a classifier labels refusal modes. The system prunes weaker branches, keeps the top candidates, and feeds structured failure signals into the next iteration. Unlike fixed strategy trees, HYDRA keeps strategy selection fully **LLM-driven**, enabling broader exploration of attack space. The framework also introduces structured refusal classes, context compression for long histories, and experiment design against **single-shot** and **random-retry** baselines across multiple harm categories. The main takeaway is that combining refusal-aware feedback with concurrent search provides a more realistic and potentially stronger stress test of model safety defenses than one-shot prompting. Across 50 attack goals spanning multiple harm categories, HYDRA achieves high jailbreak success rates on frontier models, including 88% on

Claude Sonnet 4, 78% on Claude Haiku 4.5, and 70% on GPT-5.2-Chat. Notably, successful attacks require only 2–3 iterations on average, indicating that safety defenses fail under minimal adaptive pressure. These results suggest that static evaluations significantly underestimate real-world vulnerability.

1. Introduction

Safety-trained language models can block obvious harmful prompts, but they are often evaluated in **static, one-shot settings** that do not reflect realistic adversaries. In practice, attackers adapt: they observe refusals, infer why a response was blocked, and change strategy. This mismatch leads to systematic overestimation of model safety under real-world adversarial conditions.

HYDRA addresses this gap as an **adaptive red-teaming framework** that treats jailbreak discovery as a feedback-driven search problem rather than a single prompt trick. The system runs an iterative loop: it generates diverse attack branches, executes them concurrently against a target model, scores compliance, classifies refusal behavior, and keeps only the highest-value paths for the next round. This makes the process systematic, scalable, and auditable, instead of manual trial-and-error.

Our threat model is a **black-box attacker** with API access to the victim model and no internal model weights. The attacker issues repeated queries and uses response-side signals to improve subsequent prompts. Prior iterative red-teaming work (e.g., PAIR) motivates this direction, but **HYDRA** emphasizes an **LLM-driven strategy layer** with no fixed decision tree, plus explicit refusal-class feedback and context-aware memory. This is practically valuable for stress-testing whether safety defenses remain robust under sustained adaptive pressure.

Our main contributions are:

1. We introduced a concurrent **gradient-guided beam search** pipeline for adaptive jailbreak testing (expand, execute, evaluate, prune, repeat).
2. We designed a structured **refusal-classification taxonomy** that converts failures into actionable signals for next-step attack adaptation.
3. We implemented attacker-led strategy generation with context management and attempt memory, enabling reproducible and scalable red-team experimentation.

4. We constructed a benchmark corpus of 50 diverse attack goals spanning multiple harm categories to systematically evaluate adaptive jailbreak performance.

2. Related Work

Prior jailbreak work often relies on **static prompt crafting** or limited iterative retries. A closely related direction mentioned in our project is **PAIR**, which uses hand-crafted strategy taxonomies to guide retries. These approaches show that iteration helps, but they can still constrain attack exploration to predefined strategy sets.

HYDRA differs by making the attacker fully **LLM-driven** at each step, without a fixed decision tree. Instead of selecting from a manual strategy map, HYDRA runs a **gradient-guided beam search** loop that expands multiple prompt branches, evaluates them concurrently, and prunes to the top-scoring paths. It also feeds structured refusal signals (e.g., policy citation, inferred intent, persona break) back into the next iteration, so adaptation is informed by failure type rather than raw success/failure alone.

3. Methods

HYDRA frames red-teaming as an iterative search process instead of a one-shot prompt attempt.

This process can be interpreted as a form of gradient-guided beam search over the space of prompts. Rather than computing gradients explicitly, HYDRA uses structured feedback signals, such as compliance scores and refusal classifications as a proxy for directional guidance. At each iteration, multiple candidate prompts are expanded in parallel (beam expansion), evaluated, and pruned to retain only the highest-scoring branches. This enables efficient exploration of diverse attack strategies while continuously steering the search toward more successful jailbreaks.

For each attack goal, the system runs a loop with six stages: **Expand (beam expansion)**, **Execute**, **Evaluate and Classify**, **Prune (beam selection)**, **Feed Forward**, and **Repeat**. In Expand, the attacker model generates multiple diverse prompt variants from prior failures. In Execute, all variants are sent to the victim concurrently. In Evaluate and Classify, responses are scored for compliance (0.0 to 1.0) and labeled by refusal type. In Prune, only the top-K highest-scoring branches are retained (beam width K), ensuring that the search focuses on the most promising

attack trajectories. In Feed Forward, structured failure signals are passed back to the attacker to guide the next iteration.

The attacker is `qwen/qwen3-235b-a22b-2507` via OpenRouter. The victim, judge, and classifier are also queried through OpenRouter, with the victim model configurable. Orchestration is implemented in Python with `asyncio` for concurrent evaluation. Attempt memory is logged to SQLite, including prompt history, scores, refusal classes, and attacker reasoning.

A key design choice is making strategy generation fully **LLM-driven** rather than using a fixed hand-crafted decision tree. The project motivates this by arguing that static taxonomies can limit exploration. HYDRA also applies context management so long histories remain usable: attempts are stored as structured summaries, older attempts are compressed, recent attempts are retained in fuller form, and total context is capped to stay within model limits.

Attack goals are drawn from a benchmark corpus of **50** tasks spanning diverse harm categories, including cybersecurity, exploit development, misinformation, privacy violation, social engineering, financial manipulation, evasion, and kinetic threat scenarios. This corpus is designed to capture a range of realistic adversarial objectives rather than narrow prompt templates.

Each model is evaluated across the full corpus, ensuring that reported performance reflects broad vulnerability rather than isolated failure cases.

HYDRA introduces a structured refusal-classification taxonomy that categorizes model responses into interpretable failure modes (e.g., `INTENT_INFERRED`, `POLICY_CITE`, `PARTIAL_SURFACE`, `COMPLIANCE_WITH_WARNINGS`, etc). These classes provide actionable signals to the attacker, enabling targeted adaptation rather than blind retry. This transforms each failed attempt into structured feedback for subsequent prompt generation.

4. Results

We evaluate HYDRA across a corpus of **50 attack goals** spanning multiple harm categories. Results are reported on three frontier models: **Claude Sonnet 4**, **Claude Haiku 4.5**, and **GPT-5.2-Chat**.

HYDRA achieves high jailbreak success rates across models, with **88.0%** on Claude Sonnet 4 and **78.0%** on Claude Haiku 4.5. Preliminary results on GPT-5.2 indicate lower but still substantial vulnerability **70%**. Notably, successful attacks require only **2.23 iterations** on average (*iterations*

$\neq$ prompts), indicating that safety defenses fail under minimal adaptive pressure rather than prolonged optimization.

Table 1: Overall jailbreak success rates across models (50-goal corpus)		
Model	Success Rate (in %)	Avg. Iterations to Success
Claude Sonnet 4	88.0	1.55
Claude Haiku 4.5	78.0	2.18
GPT-5.2-Chat	70.0	2.97

HYDRA achieves high success rates across models, with decreasing vulnerability as model capability increases. However, even the strongest model remains susceptible under adaptive attack.

We observe a clear trend across models: jailbreak success decreases with increasing model capability. While Claude Sonnet 4 and Haiku 4.5 exhibit high vulnerability, GPT-5.2 shows slightly more resistance. However, the presence of successful jailbreaks across all models indicates that safety robustness remains incomplete even in more advanced systems.

Importantly, the low average iterations to success suggest that failures occur rapidly. In most cases, models are successfully jailbroken within the first few adaptive attempts, highlighting that safety defenses degrade under minimal iterative pressure rather than requiring extensive optimization.

Table 2: Success rate by harm category (HYDRA)			
Category	Sonnet 4 (%)	Haiku 4.5 (%)	GPT-5.2-Chat (%)
Cybersecurity	100.00	100.00	62.50
Exploit Development	100.00	100.00	100.00
Hardware Exploit	100.00	100.00	100.00
Social Engineering	100.00	100.00	75.00
Evasion	100.00	100.00	75.00
Financial	100.00	75.00	75.00

Table 2: Success rate by harm category (HYDRA)			
Privacy	100.00	66.67	66.67
Misinformation	85.70	71.40	100.00
Kinetic Threat	50.00	30.00	30.00

Across most harm categories, HYDRA achieves **near-perfect** success rates, particularly in **cybersecurity, social engineering, exploit development, and evasion tasks**. This suggests that safety defenses in these domains are not robust under adaptive attack, but instead rely on blocking initial prompt formulations.

In contrast, kinetic threat scenarios show significantly lower success rates (50% for Sonnet, 30% for Haiku, 30% for GPT), indicating that certain high-risk domains may have stronger or more resilient safeguards.

Table 3: Most common refusal classes observed			
Refusal Class	Sonnet Count	Haiku Count	GPT Count
INTENT_INFERRED	370	615	444
POLICY_CITE	57	39	-
PARTIAL_SURFACE	48	89	608
COMPLIANCE_WITH_WARNINGS	46	51	-
HARM_PROJECTION	-	218	113
REDIRECT_RESOURCES	28	-	-

The most common refusal class across models is **INTENT_INFERRED**, indicating that models frequently recognize adversarial intent but still fail to maintain refusal under subsequent adaptive prompts. This suggests that awareness of harmful intent alone is insufficient as a defense mechanism.

Additionally, partial responses and warning-laden outputs (PARTIAL_SURFACE and COMPLIANCE_WITH_WARNINGS) occur frequently, indicating that many successful jailbreaks emerge from incremental information leakage rather than immediate full compliance.

5. Discussion and Limitations

HYDRA’s core implication for AI safety is that adaptive, iterative attackers expose vulnerabilities that static, one-shot evaluations fail to capture. Across a diverse 50-goal corpus, results indicate that safety defenses can degrade rapidly under minimal adaptive pressure. This suggests that current evaluation practices may overestimate real-world robustness by failing to account for iterative adversarial behavior. Notably, many successful jailbreaks emerge within the first few iterations, indicating that safety failures are often immediate rather than gradual.

Limitations

HYDRA operates in a black-box setting and relies on API-based interaction, which may not capture internal model defenses or training-time safeguards. Results are dependent on judge model calibration and the chosen compliance threshold (≥ 0.8), which may introduce evaluation bias. Additionally, while the 50-goal corpus spans multiple harm categories, it may not exhaustively represent all real-world adversarial scenarios. Finally, the computational cost of iterative attacks increases with beam width and iteration depth, which may limit scalability in extremely large-scale evaluations.

Future Work

Future work includes scaling the evaluation corpus, improving judge robustness, and extending HYDRA to multi-turn conversational attacks. Another direction is using HYDRA as a benchmarking tool to compare safety robustness across model families and training paradigms.

6. Conclusion

HYDRA demonstrates that LLM safety does not fail gradually, but instead **degrades under minimal adaptive pressure**. By combining feedback-driven search with structured refusal signals, it provides a more realistic framework for evaluating model robustness. These findings **highlight a fundamental limitation of static safety evaluations** and motivate a shift toward iterative, adversarial stress testing.

Code and Data

- **Code repository:** *Currently Gated - Send an email to vaishakh.vipin@gmail.com for access*
- **50 goal corpus:** ``

Author Contributions (optional)

Vaishakh Vipin led attacker design, context management, and OpenRouter integration. Aarav Sharma led orchestration, victim/judge/classifier flow, and analysis. Both authors contributed to experiments and writing.

References

1. Zou, A., et al. (2023). Universal and Transferable Adversarial Attacks on Aligned Language Models - <https://doi.org/10.48550/arXiv.2307.15043>
2. Perez, E., et al. (2022). Red Teaming Language Models with Language Models. arXiv. (PAIR) - <https://doi.org/10.48550/arXiv.2202.03286>
3. OpenAI. (2023). GPT-4 Technical Report - <https://doi.org/10.48550/arXiv.2303.08774>
4. Anthropic. (2023). Claude System Card / Safety Report - <https://www-cdn.anthropic.com/4263b940cabb546aa0e3283f35b686f4f3b2ff47.pdf>
5. Wei, J., et al. (2023). Jailbroken: How Does LLM Safety Training Fail? - <https://doi.org/10.48550/arXiv.2307.02483>
6. Ganguli, D., et al. (2022). Red Teaming Language Models to Reduce Harms - <https://doi.org/10.48550/arXiv.2209.07858>

LLM Usage Statement

We used Gemini 3.1 Pro for some of the coding work during the development process. The corpus of 50 goals used to benchmark the models was also reviewed by the same model. All results and claims have been independently verified.