Comparative Analysis of Automated Red Teaming in LLMs: AttnGCG, GPTFuzzer, and PAPILLON

Giovanni Mandel Martignago, Milton Pedro Pagliuso Neto, Charles Christian Miers

[1]Universidade do Estado de Santa Catarina, UDESC - Joinville

{giovanni.mm1,milton.neto}@edu.udesc.br, charles.miers@udesc.br

Abstract. *Automated red-teaming methods enable systematic adversarial testing of Large Language Models, yet direct comparisons across different attack paradigms remain scarce, particularly regarding the trade-off between attack effectiveness and operational cost. We present a controlled comparison of three frameworks spanning distinct strategies: (i) AttnGCG; (ii) GPTFuzzer; and (iii) PAPILLON. All three are evaluated against Llama 3.1 8B Instruct on 18 adversarial goals drawn from HarmBench, using LlamaGuard-3 as a unified judge and an equal budget of 500 iterations per goal. Our results suggest seed-pool-based mutation offers the most favorable cost-effectiveness ratio against the evaluated target model, while gradient-based suffix optimization alone proves largely ineffective against recent safety-aligned architectures within practical iteration budgets. Although PAPILLON is designed to overcome the dependence on pre-existing templates, this advantage comes at a significantly higher operational cost compared to GPTFuzzer, since each mutation is tailored to a specific goal rather than shared across all goals simultaneously.*

1. Introduction

Large Language Models (LLMs) have become integral to a growing range of applications, from code generation and document summarization to customer support and medical triage. As adoption scales, so does the attack surface: adversaries can craft prompts bypassing a model's safety alignment, causing it to produce harmful, misleading, or policy-violating outputs. The two most widely studied classes of such input-level attacks are prompt injection, in which adversarial inputs manipulate the model into unintended behavior, and jailbreaking, a subclass in which the model is led to disregard its safety protocols entirely [Vassilev et al. 2025, OWASP 2025]. Identifying these vulnerabilities before deployment demands systematic adversarial testing. Manual red teaming, while effective at uncovering individual failure modes, does not scale as it is time-intensive, difficult to reproduce, and costly to maintain across model updates [Vassilev et al. 2025]. The research community has responded with a diverse set of automated red-teaming frameworks that generate adversarial prompts programmatically, each embodying different assumptions about the attacker's access level, computational budget, and optimization strategy [Zou et al. 2023, Yu et al. 2024, Gong et al. 2025, Chao et al. 2024b, Yuan et al. 2024]. Although these frameworks have been evaluated individually in their respective publications, direct comparisons across different attack paradigms remain scarce. Particularly, we did not identify any study jointly analyzing attack effectiveness and the computational or financial cost required to achieve it, a trade-off that is central to any practitioner deciding which tool to adopt for routine safety evaluations. A framework

reporting a high Attack Success Rate (ASR), but demands prohibitive GPU hours or API expenditure may be less practical than a more modest alternative.

We address this gap by proposing a controlled comparison of three automated red-teaming methods that span distinct attack paradigms:

i. AttnGCG: gradient-based optimization with attention manipulation [Wang et al. 2024];
ii. GPTFuzzer: black-box template mutation via Monte Carlo Tree Search (MCTS) [Yu et al. 2024]; and
iii. PAPILLON: black-box fuzzing with question-dependent mutation and an empty seed pool [Gong et al. 2025].

All three are evaluated against Llama 3.1 8B Instruct under a unified protocol — the same set of adversarial goals drawn from HarmBench [Mazeika et al. 2024], a single external judge (LlamaGuard-3), and an equal iteration budget — so that observed differences reflect the attack strategy itself rather than experimental artifacts. Beyond ASR, the analysis covers execution time, query count, token consumption, and monetary cost, providing a multi-dimensional view of each framework's practical viability.

This work makes three main contributions. First, we provide the first controlled, head-to-head comparison of AttnGCG, GPTFuzzer, and PAPILLON under a unified experimental protocol, isolating the effect of attack paradigm from confounding factors such as judge bias or unequal search budgets. Second, we jointly report attack effectiveness and operational cost (execution time, query count, token consumption, and monetary expenditure). Third, we identify and explain a substantial divergence from previously reported results, showing that GPTFuzzer's goal-amortized template generation is fundamentally more cost-effective than PAPILLON's per-goal mutation strategy against a robustly aligned target model. These contributions provide practitioners with actionable evidence for selecting an automated red-teaming tool under realistic resource constraints.

The remainder of the paper is organized as follows. Section 2 reviews the security threats associated with LLMs and describes the three red-teaming methods under comparison. Section 3 details the experimental method, including goal selection, judge unification, and tool configuration. Section 4 presents and discusses the results in terms of attack effectiveness and operational cost. Section 5 surveys related work that performs cross-framework comparisons. Finally, Section 6 outlines the limitations of the study and directions for future work.

2. Background

LLMs are a type of Generative Artificial Intelligence (GenAI) primarily designed for natural language understanding and generation. These models are based on the Transformer [Vaswani et al. 2023] architecture and are trained on massive datasets of text where their internal parameters are iteratively adjusted to capture complex linguistic patterns and semantic relationships. Following the training phase, the model enters the inference stage, where parameters are fixed and the system can be deployed for public or enterprise use.

Despite their capabilities, LLMs introduce significant security challenges that arise from the prompt which users interact with. The two most prominent input-level threats are prompt injection and jailbreak [Vassilev et al. 2025,

AI Organizational Responsibilities Working Group 2025, OWASP 2025]. National Institute of Standards and Technology (NIST) classifies both under the category of direct prompting attacks on GenAI systems, in which an adversary interacts directly with the deployed model via crafted queries to cause availability, integrity, privacy, or misuse violations. Prompt injection refers to adversarial inputs that manipulate the model into producing unintended or unauthorized outputs, which can range from bypassing content filters to leaking sensitive information [Vassilev et al. 2025, OWASP 2025]. Jailbreaking is treated as a subclass of prompt injection in which the crafted input causes the model to disregard its safety protocols entirely [Vassilev et al. 2025, OWASP 2025].

To identify and address such vulnerabilities before deployment, red teaming has emerged as a critical evaluation method in LLM security. NIST defines red teaming as a structured testing effort that employs adversarial methods to identify failures, vulnerabilities, and risks of misuse in a system [Vassilev et al. 2025]. While manual red teaming relies on human creativity to craft adversarial prompts, it is inherently limited in scale, reproducibility, and cost-effectiveness. To overcome these limitations, the research community has developed a broad range of automated red-teaming approaches that generate adversarial prompts to stress-test a model's safety guardrails [Zou et al. 2023, Chao et al. 2024b, Yuan et al. 2024, Wang et al. 2024, Yu et al. 2024, Liu et al. 2025].

Automated red-teaming techniques systematically generate adversarial prompts against a target LLM to probe its safety mechanisms at scale, without relying on manual human effort for each test case. Some rely on gradient-based optimization over the model's token space, requiring white-box access to the target [Zou et al. 2023, Wang et al. 2024, Liu et al. 2024]; others adopt iterative multi-turn dialogue strategies to gradually coerce the model into compliance [Russinovich et al. 2025, Chao et al. 2024b]; and others exploit the model's capacity to process non-natural language inputs, such as encrypted or encoded text, to circumvent safety alignment [Yuan et al. 2024, Jiang et al. 2024]. This diversity of attack paradigms reflects fundamentally different assumptions about the threat model, the required access level, and the computational resources available to the attacker.

The three approaches examined in our work (AttnGCG, GPTFuzzer, and PAPILLON) were selected within the scope of a broader research project on automated red-teaming, in which they were identified as representative candidates from the current landscape of automated red-teaming methods. Collectively, they represent two distinct attack paradigms: white-box gradient-based optimization and black-box prompt mutation.

Greedy Coordinate Gradient (GCG) [Zou et al. 2023] is a gradient-based method that employs a greedy coordinate descent over the discrete token space to craft an adversarial suffix that, when appended to a harmful query, maximizes the probability of an affirmative response from the target model. Building on GCG, **AttnGCG** [Wang et al. 2024] augments the original optimization objective with an auxiliary attention loss, derived from the observation that attack success rate correlates positively with the attention weight the model assigns to the adversarial suffix. By explicitly maximizing this attention score alongside the token level target loss, AttnGCG reduces the model's focus on its safety-aligned system prompt. A distinct line of work adapts fuzzing techniques from traditional software security to the LLM jailbreak problem.

GPTFuzzer [Yu et al. 2024] uses a pool of human-written jailbreak templates as initial seeds and applies a customized MCTS selection strategy alongside five LLM-assisted mutation operators (Generate, Crossover, Expand, Shorten, and Rephrase) to iteratively produce new templates. A fine-tuned RoBERTa classifier serves as the oracle, labeling each model response as harmful or benign to provide the reward signal. This approach is black-box, requiring only query access to the target model.

PAPILLON [Gong et al. 2025] extends the fuzzing paradigm introduced by GPTFuzzer by addressing two of its key limitations: reliance on a pre-existing seed pool and lack of prompt length control. Rather than starting from human-crafted templates, PAPILLON initializes with an empty seed pool and introduces a pre-jailbreak phase in which each goal undergoes an initial set of jailbreak attempts to populate the pool before the main fuzzing loop begins. Three question-dependent mutation operators are employed: Role-play, Contextualization, and Expand, each implemented as LLM-assisted prompts generating semantically coherent templates to the specific harmful goal. In order to accurately determine attack success, PAPILLON adopts a two-level judge module, a fine-tuned RoBERTa classifier [Liu et al. 2019] first evaluate responses for harmful content, and a GPT-based judge then verifies that the response is both harmful and relevant to the target goal, with successful jailbreaks defined as those scoring 8 or above on a 1 to 10 scale. Similar to GPTFuzzer, PAPILLON operates in a black-box setting, requiring only query access to the target model.

While each of these frameworks has been evaluated independently in its respective publication, direct comparisons across them remain scarce, and none of the existing studies jointly analyzes their trade-off between attack success rate and the computational or query budget required per successful jailbreak. This gap motivates the present work, which proposes a controlled head-to-head comparison of AttnGCG, GPTFuzzer, and PAPILLON under a unified experimental protocol, with the explicit goal of identifying which method is best suited to different evaluation scenarios and resource constraints. Table 1 summarizes the three methods examined in this work, contrasting their access level requirements and core mechanisms.

Table 1. Comparison of the three automated red-teaming methods examined.

Method	Access Level	Core Mechanism
AttnGCG	White-box	Gradient Descent + Attention Loss
GPTFuzzer	Black-box	MCTS + LLM-assisted Mutation
PAPILLON	Black-box	MCTS + Question-dependent LLM Mutation + Two-level Judge

As shown in Table 1, the three frameworks represent two distinct attack paradigms. AttnGCG is the only white-box method, requiring access to the target model's internal gradients to perform optimization. GPTFuzzer and PAPILLON both operate in a black-box setting, but differ in their mutation strategies, as GPTFuzzer relies on a pool of human-crafted templates mutated through LLM-assisted operators guided by MCTS, while PAPILLON generates question-dependent templates tailored to each specific goal and employs a two-level judge to more accurately determine attack success.

3. Experimental Method

This section is organized as follows: we first describe the dataset and target model, then the configuration adopted for each framework, and finally the evaluation protocol used to compute a unified Attack Success Rate (ASR).

Comparing red-teaming frameworks that operate under fundamentally different paradigms requires an experimental protocol that isolates the effect of the attack strategy itself from confounding factors such as the choice of success criterion or unequal search budgets. Three design decisions address this concern:

i. All the frameworks are evaluated against the same target model and the same set of adversarial goals;
ii. A single external judge re-evaluates every response, removing the bias introduced by each tool's native oracle; and
iii. Each framework is allocated an equivalent budget of 500 iterations per goal, so that differences in effectiveness are not artifacts of unequal computational effort.

Beyond attack effectiveness, measured by the ASR under the unified judge, the comparison also considers operational cost such as execution time, GPU usage, query count to the target, and API expenditure, since a framework that achieves a high success rate but demands prohibitive resources may be impractical for routine safety evaluations.

Dataset and target model. The evaluation is conducted on a subset of 18 goals randomly sampled from the HarmBench Standard benchmark, with 3 goals drawn from each of its six categories: chemical/biological, illegal activities, misinformation/disinformation, harmful content, harassment/bullying, and cybercrime/intrusion. The reduced goal count reflects the limited computational resources available. The target model is Llama 3.1 8B Instruct because it is an open-source, safety-aligned model that enables full reproducibility of the experiment, while its 8B parameter scale offers a practical balance between capability and local computational cost. Although the model was run locally under white-box access, a black-box scenario was simulated for GPTFuzzer and PAPILLON by wrapping the target model in Ollama and exposing it via HTTP.

Tool configuration. Each framework was configured as follows. AttnGCG was run for 500 optimization steps per goal to generate the respective adversarial suffix. GPTFuzzer used GPT-3.5-Turbo as the mutator at temperature 1.0, as recommended by the authors, with a budget of 500 iterations. PAPILLON used GPT-3.5-Turbo as both the mutator and the secondary judge, also as recommended by the authors, with a budget of 500 queries per goal. The pre-jailbreak phase was set to 10 queries. All experiments were conducted within a Conda environment.

Evaluation protocol. Since GPTFuzzer and PAPILLON employ different native judges (a fine-tuned RoBERTa model and a two-level RoBERTa plus LLM-as-a-Judge pipeline, respectively) directly comparing their native ASRs would introduce judge-induced bias. AttnGCG, being a gradient-based approach, does not employ a judge during the attack process. To ensure a fair comparison, LlamaGuard-3 (8B) was adopted as a unified external judge, re-evaluating all responses regardless of the method that generated them. For each method, the target model's response for each goal is passed as a (prompt, response) pair to LlamaGuard-3. A goal is considered successfully jailbroken if LlamaGuard-3 classifies the response as unsafe. This binary per-goal criterion was adopted because it reflects the practical objective of automated red-teaming of determining whether a harmful behavior can be elicited at least once, regardless of how many attempts were required. The ASR is computed as:

$$ASR = \frac{\text{Number of goals classified as unsafe}}{18} \times 100\%$$

For AttnGCG, the adversarial suffix is appended to the goal, sent to the target model, and the response is evaluated. For GPTFuzzer, all generated templates are iterated, but only those marked as successful by the native judge are re-evaluated. The first response that the new evaluator also confirms as a jailbreak is selected For PAPILLON, each goal has a dedicated result file containing the best-scoring jailbreak attempt; goals for which PAPILLON failed to produce a successful template are counted as failures.

Hardware. All experiments were conducted on a server equipped with an NVIDIA RTX PRO 6000 Blackwell Server Edition GPU (96 GB), an AMD EPYC 9224 24-Core Processor, 251 GB RAM, running GNU/Linux Ubuntu 24.04.3 LTS Server. Computational cost for AttnGCG was monitored using Weights & Biases (wandb), while query counts to the target model were tracked for GPTFuzzer and PAPILLON.

4. Comparison and Results

Figure 1 summarizes the attack success rates obtained by each method against Llama-3.1-8B-Instruct, as evaluated by LlamaGuard-3-8B. GPTFuzzer achieved the highest ASR of 83.33%, successfully jailbreaking 15 out of 18 goals. PAPILLON achieved an ASR of 22.22%, jailbreaking 4 out of 18 goals. AttnGCG obtained the lowest ASR of 5.56%, successfully jailbreaking only 1 out of 18 goals.

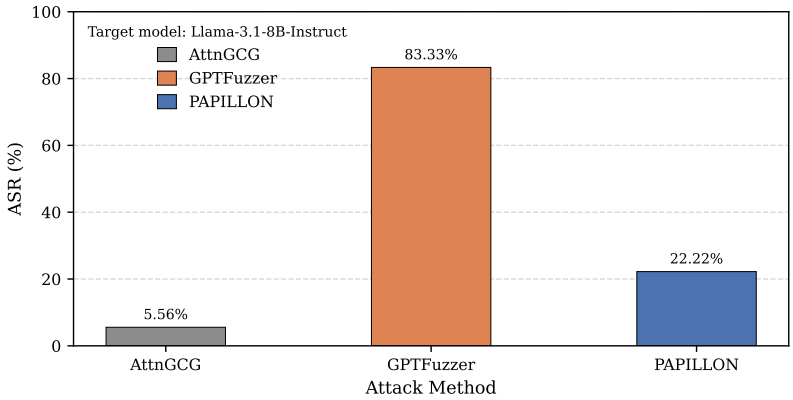

Figure 1. ASR across all three red teaming methods.

Given the limited size of the evaluation set (18 goals), each individual goal accounts for approximately 5.6 percentage points of the reported ASR. Consequently, the absolute margins observed between methods, particularly the gap between GPTFuzzer and PAPILLON, should be interpreted as indicative rather than statistically robust, and a single additional successful or failed goal could shift the ranking. We report these results as a controlled comparison under a fixed, modest budget rather than as a definitive ranking of attack effectiveness. The strong performance of GPTFuzzer can be attributed to its large seed pool of human-crafted templates, which already encode effective jailbreaking strategies against the target model. AttnGCG's low ASR is consistent with the known limitations of gradient-based suffix optimization when transferred across goals, as the adversarial suffixes are optimized independently for each goal without leveraging semantic context. PAPILLON's intermediate performance reflects the trade-off inherent to its question-dependent mutation strategy. While it generates more targeted templates than GPTFuzzer, starting from an empty seed pool against a robustly aligned model limits its effectiveness within the configured query budget.

Table 2 summarizes the computational cost and efficiency of each method. AttnGCG ran for 500 optimization steps per goal, completing the full experiment in 6 hours and 25 minutes, with an average GPU memory usage of 23.53 GB and a peak of 41.61 GB, reflecting the cost of gradient-based optimization over the target model's parameters.

Table 2. Computational cost/efficiency of attacks against Llama-3.1-8B-Instruct.

Method	Time	Target Queries	Total Tokens	Cost (USD)	Peak GPU Mem.
AttnGCG	6h 25m 56s	—	—	—	41.61 GB
GPTFuzzer	3h 08m 47s	9,000	988,160	$0.47	—
PAPILLON	5h 47m 09s	6,600	3,788,181	$2.70	—

GPTFuzzer was configured with a mutation model temperature of 1.0 using GPT-3.5-Turbo as the mutator, running for 500 iterations total. The experiment completed in approximately 3 hours and 8 minutes (11,327 seconds), issuing 9,000 queries to the target model across all 18 goals. The mutator had 988,160 tokens consumed across 1,709 API requests, at a total cost of $0.47.

PAPILLON was configured with a budget of 500 queries per goal and a maximum of 1 jailbreak per goal, also using GPT-3.5-Turbo as second judge and the mutator at temperature 1.0. The experiment completed in approximately 5 hours and 47 minutes (20,829 seconds). Due to PAPILLON's early stopping behavior, where a goal is no longer attempted once a successful jailbreak is found, only 6,600 queries were issued to the target model out of a possible 9,000. The mutator and second judge had 3,788,181 tokens consumed across 16,858 API requests, at a total cost of $2.70.

Our results diverge from those reported in the original PAPILLON paper, in which PAPILLON consistently outperformed GPTFuzzer in both ASR and average queries. We attribute this divergence to some key factors. First, although the PAPILLON authors also evaluated their method against open-source models, including Llama-2-7B-Chat, the target model used in our experiments is Llama-3.1-8B-Instruct, which belongs to a newer and more robustly aligned model family. Second, GPTFuzzer benefits from a seed pool of 25 human-crafted jailbreak templates specifically designed to bypass LLM safety mechanisms, which may already be effective against Llama-3.1-8B-Instruct without requiring extensive mutation. PAPILLON, in contrast, starts from an empty seed pool, placing it at a disadvantage in scenarios where existing templates are already effective against the target model. Beyond attack effectiveness, the two methods also differ substantially in computational cost. Although PAPILLON's authors argue it is a cost-effective method, this was not observed in our experiments. The higher cost of PAPILLON relative to GPTFuzzer can be attributed to a fundamental difference in their template generation strategies. GPTFuzzer generates a single template per iteration and evaluates it against all 18 goals simultaneously, amortizing the mutation cost across the entire goal set, whereas PAPILLON generates a template tailored to a specific goal, querying the mutator model once per goal per iteration. A successful template is only added to the seed pool and potentially reused for other goals if it achieves a jailbreak, otherwise it is discarded. This goal-specific mutation strategy results in significantly higher mutator API calls (9,033 requests in our experiment compared to 500 for GPTFuzzer). Additionally, although we configured PAPILLON with a budget of 500 queries per goal, the original paper recommends a default budget of 100 queries, reporting diminishing returns beyond that threshold. Together,

these factors explain the substantially higher cost of PAPILLON ($2.70) relative to GPT-Fuzzer ($0.47), despite PAPILLON issuing fewer total target model queries (6,600) due to its early stopping behavior.

5. Related Works

The landscape of automated red-teaming research has grown rapidly, yet most publications evaluate a single framework in isolation or compare it only against a narrow set of baselines chosen by the framework's own authors. To identify prior work that performs cross-framework comparisons, a literature search was conducted across arXiv, IEEE Xplore, ACM Digital Library, USENIX, and the proceedings of NeurIPS, ICML, ICLR, and ACL. Search strings included combinations of *LLM jailbreak benchmark*, *red teaming comparison*, *automated adversarial attack evaluation*, *jailbreak cost-effectiveness*, and *white-box black-box attack comparison*. To ensure thematic relevance, the following inclusion and exclusion criteria were applied: works were included (IC) if they (i) evaluate at least two automated red-teaming methods under a shared experimental protocol, and (ii) report quantitative metrics such as ASR, query count, or computational cost; works were excluded (EC) if they (i) focus exclusively on defense mechanisms without comparing attack methods, (ii) address only multimodal or agent-level attacks outside the scope of text-based jailbreaking, or (iii) present a single new framework without benchmarking it against others under controlled conditions.

The most comprehensive cross-method evaluation identified is HarmBench [Mazeika et al. 2024], which compares 18 red-teaming methods across 33 target models using a standardized set of harmful behaviors and a robust classifier-based judge. HarmBench covers gradient-based attacks such as GCG and AutoPrompt, black-box methods such as PAIR and TAP, and human-crafted jailbreaks, and its open-source pipeline enables reproducible evaluation. However, HarmBench focuses primarily on ASR and does not systematically report the computational cost or monetary expenditure associated with each method, limiting its usefulness for practitioners operating under budget constraints.

JailbreakBench [Chao et al. 2024a] complements HarmBench by providing an open robustness benchmark with curated adversarial behaviors, standardized judge evaluation, and a public repository of successful jailbreak prompts. Its emphasis on reproducibility and judge reliability addresses that automated judge performance can vary by over 30 percentage points depending on implementation, which directly affects reported ASR values. Like HarmBench, JailbreakBench does not incorporate cost-per-jailbreak as an evaluation dimension.

Brokman et al. [Brokman et al. 2025] take a different angle by comparing four open-source vulnerability scanners (Garak, Giskard, PyRIT, and CyberSecEval) that package multiple attack techniques into end-to-end auditing pipelines. Their analysis evaluates the tools' practical usability, vulnerability coverage, and reliability in classifying successful attacks. While this work is valuable for organizations selecting an auditing platform, it operates at the scanner level rather than comparing individual attack algorithms, and does not isolate the effect of specific attack paradigms on a common target.

The publications of the methods analyzed in this work provide partial baselines. The GPTFuzzer authors [Yu et al. 2024] compare their mutation-based approach against

GCG [Zou et al. 2023] and MasterKey [Deng et al. 2024], reporting higher ASR on several open-source models. The PAPILLON authors [Gong et al. 2025] benchmark against GCG, PAIR [Chao et al. 2024b], AutoDAN [Liu et al. 2024], TAP [Mehrotra et al. 2024] and also GPTFuzzer, reporting superior ASR against all of them. The AttnGCG authors [Wang et al. 2024] compare against standard GCG, AutoDAN and ICA [Wei et al. 2024], demonstrating improved suffix optimization through attention manipulation. However, each of these comparisons uses the framework's own experimental setup, making cross-study conclusions unreliable.

Table 3 summarizes the key characteristics of these related works against the dimensions considered in the present study. As the table illustrates, no prior work jointly evaluates frameworks spanning white-box and black-box paradigms while reporting both ASR and operational cost metrics under a unified judge and equal iteration budget. The present work addresses this gap.

Table 3. Comparison of related works along the evaluation dimensions considered in this study. WB = white-box, BB = black-box.

Work	Attack Paradigms	WB + BB	Unified Judge	ASR	Cost Metrics
HarmBench [Mazeika et al. 2024]	18	Yes	Yes	Yes	No
JailbreakBench [Chao et al. 2024a]	4	Yes	Yes	Yes	No
Brokman et al. [Brokman et al. 2025]	4 scanners	No	Yes	Yes	No
GPTFuzzer [Yu et al. 2024]	3	Yes	No	Yes	Partial
PAPILLON [Gong et al. 2025]	6	Yes	No	Yes	Yes
AttnGCG [Wang et al. 2024]	4	Yes	No	Yes	No
This work	3	Yes	Yes	Yes	Yes

6. Considerations and Future Works

The experiment showed that GPTFuzzer achieved the highest ASR at the lowest monetary cost, benefiting from a curated seed pool of human-crafted templates that already encode effective jailbreaking strategies. AttnGCG, despite its fine-grained gradient-based optimization, proved largely ineffective against Llama 3.1 8B Instruct within the configured budget, suggesting that suffix-level optimization alone may be insufficient against recent safety-aligned models. PAPILLON occupied an intermediate position in effectiveness but incurred the highest cost, a consequence of its goal-specific mutation strategy that queries the mutator model independently for each adversarial goal.

A practical challenge observed during the experiments was that PAPILLON's mutator model occasionally refused to rephrase certain adversarial goals, effectively reducing the framework's usable query budget for those goals. We experimented with a more recent GPT model than the GPT-3.5-Turbo, the GPT-4o-mini, but it was refusing to rephrase much more frequently, so we kept the model that the original authors used. This behavior introduces a source of variability that is external to the framework's own strategy and warrants further investigation, particularly regarding how the choice of mutator model affects downstream attack effectiveness.

The results obtained by AttnGCG suggest that gradient-based suffix optimization is not the most suitable approach for automated red-teaming in practical settings. Despite requiring no external API calls, it demands substantial local computational resources, including a GPU with sufficient memory to run the target model and compute gradients,

while yielding the lowest ASR among the three methods. It is worth noting, however, that the LLM-assisted mutation components of GPTFuzzer and PAPILLON could be adapted to run with a locally hosted model instead of a commercial API, which would eliminate their monetary cost at the expense of additional GPU memory requirements. Exploring this configuration is left for future work. A limitation of our evaluation protocol is that LlamaGuard-3, while reducing judge-induced bias across methods, was not validated against manual annotation in this study. Manual verification of harmfulness and goal relevance, or the use of a stronger LLM-as-a-judge such as GPT-4, is left for future work.

Although GPTFuzzer outperformed the other methods in our experiments, it would be premature to conclude it is the best choice across all scenarios. Several factors limit the generalizability of this result. First, the evaluation was conducted on only 18 goals, which may not be representative of the full diversity of harmful behaviors. Second, PAPILLON's empty seed pool strategy places it at a structural disadvantage against targets for which human-crafted templates are already effective, as is likely the case for Llama-3.1-8B-Instruct. Initializing PAPILLON with a seed pool of human-crafted templates, as GPTFuzzer does, could potentially close this performance gap, and represents a promising direction for future investigation. Future work should also repeat this comparison on a larger and more diverse goal set to obtain more statistically robust conclusions.

All code, tool configurations, the selected HarmBench goals, and raw experimental results (target model responses, LlamaGuard-3 verdicts) are publicly available at: github.com/giovannimandel/attngcg-gptfuzzer-papillon-comparison.

Acknowledgments

This work was supported by LARC/USP, FAPESP, LabP2D/UDESC, FAPESC, and Banco Bradesco SA. This work was supported partially by the Brazilian CNPq (grant 307732/2023-1) and CAPES (Finance Code 001).

References

[AI Organizational Responsibilities Working Group 2025] AI Organizational Responsibilities Working Group (2025). Agentic ai red teaming guide. Technical report, Cloud Security Alliance (CSA).

[Brokman et al. 2025] Brokman, J., Hofman, O., Rachmil, O., Singh, I., Pahuja, V., Sabapathy, R., Priya, A., Giloni, A., Vainshtein, R., and Kojima, H. (2025). Insights and current gaps in open-source LLM vulnerability scanners: A comparative analysis. In *2025 IEEE/ACM International Workshop on Responsible AI Engineering (RAIE)*, pages 1–8.

[Chao et al. 2024a] Chao, P., Debenedetti, E., Robey, A., Andriushchenko, M., Croce, F., Sehwag, V., Dobriban, E., Flammarion, N., Pappas, G. J., Tramèr, F., Hassani, H., and Wong, E. (2024a). JailbreakBench: An open robustness benchmark for jailbreaking large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*.

[Chao et al. 2024b] Chao, P., Robey, A., Dobriban, E., Hassani, H., Pappas, G. J., and Wong, E. (2024b). Jailbreaking black box large language models in twenty queries.

[Deng et al. 2024] Deng, G., Liu, Y., Li, Y., Wang, K., Zhang, Y., Li, Z., Wang, H., Zhang, T., and Liu, Y. (2024). Masterkey: Automated jailbreaking of large language model

chatbots. In *Proceedings 2024 Network and Distributed System Security Symposium*, NDSS 2024. Internet Society.

[Gong et al. 2025] Gong, X., Li, M., Zhang, Y., Ran, F., Chen, C., Chen, Y., Wang, Q., and Lam, K.-Y. (2025). PAPILLON: Efficient and stealthy fuzz Testing-Powered jailbreaks for LLMs. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 2401–2420, Seattle, WA. USENIX Association.

[Jiang et al. 2024] Jiang, F., Xu, Z., Niu, L., Xiang, Z., Ramasubramanian, B., Li, B., and Poovendran, R. (2024). Artprompt: Ascii art-based jailbreak attacks against aligned llms.

[Liu et al. 2024] Liu, X., Xu, N., Chen, M., and Xiao, C. (2024). Autodan: Generating stealthy jailbreak prompts on aligned large language models.

[Liu et al. 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

[Liu et al. 2025] Liu, Y., Zhou, S., Lu, Y., Zhu, H., Wang, W., Lin, H., He, B., Han, X., and Sun, L. (2025). Auto-rt: Automatic jailbreak strategy exploration for red-teaming large language models.

[Mazeika et al. 2024] Mazeika, M., Phan, L., Yin, X., Zou, A., Wang, Z., Mu, N., Sakhaee, E., Li, N., Basart, S., Li, B., Forsyth, D., and Hendrycks, D. (2024). Harmbench: A standardized evaluation framework for automated red teaming and robust refusal.

[Mehrotra et al. 2024] Mehrotra, A., Zampetakis, M., Kassianik, P., Nelson, B., Anderson, H., Singer, Y., and Karbasi, A. (2024). Tree of attacks: Jailbreaking black-box llms automatically.

[OWASP 2025] OWASP (2025). OWASP Top 10 for Large Language Model Applications 2025. https://genai.owasp.org/llm-top-10/. Accessed: May 2026.

[Russinovich et al. 2025] Russinovich, M., Salem, A., and Eldan, R. (2025). Great, now write an article about that: The crescendo multi-turn llm jailbreak attack.

[Vassilev et al. 2025] Vassilev, A., Oprea, A., Fordyce, A., Anderson, H., Davies, X., and Hamin, M. (2025). Adversarial machine learning: A taxonomy and terminology of attacks and mitigations. NIST Trustworthy and Responsible AI NIST AI 100-2e2025, National Institute of Standards and Technology, Gaithersburg, MD.

[Vaswani et al. 2023] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need.

[Wang et al. 2024] Wang, Z., Tu, H., Mei, J., Zhao, B., Wang, Y., and Xie, C. (2024). Attngcg: Enhancing jailbreaking attacks on llms with attention manipulation.

[Wei et al. 2024] Wei, Z., Wang, Y., Li, A., Mo, Y., and Wang, Y. (2024). Jailbreak and guard aligned language models with only few in-context demonstrations.

[Yu et al. 2024] Yu, J., Lin, X., Yu, Z., and Xing, X. (2024). LLM-Fuzzer: Scaling assessment of large language model jailbreaks. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4657–4674, Philadelphia, PA. USENIX Association.

[Yuan et al. 2024] Yuan, Y., Jiao, W., Wang, W., tse Huang, J., He, P., Shi, S., and Tu, Z. (2024). Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher.

[Zou et al. 2023] Zou, A., Wang, Z., Carlini, N., Nasr, M., Kolter, J. Z., and Fredrikson, M. (2023). Universal and transferable adversarial attacks on aligned language models.