BEACON: Budget-Efficient Discovery of Policy Violations in Large Language Models via Cognitive-Guided Monte Carlo Tree Search

Xinyi Huang, **Jie Wang**, **Pengrui Xiang**, **Yifan Wang**, **Yu Fu**, **Jinduo Liu**†
Beijing University of Technology, Beijing, China
huangxinyi@emails.bjut.edu.cn, wangjie@emails.bjut.edu.cn, xiangpengrui@126.com,
wang3225359728@163.com, isfuyu010@163.com, jinduo@bjut.edu.cn

Abstract

Systematic safety evaluation of large language models must uncover diverse policy violations under tight query budgets. However, most red-teaming methods optimize attack success rate and repeatedly probe a narrow set of vulnerabilities, yielding redundant failures and leaving rarer yet critical violation categories unexplored. Under fixed budgets, such inefficient exploration delays the first discovery and limits category coverage. To address these limitations, we propose the **Budget-Efficient Adaptive Cognitive Offense Navigator** (**BEACON**), a budget-aware safety testing framework that uses Cognitive-Guided Monte Carlo Tree Search to navigate the violation search space under fixed budgets. BEACON innovatively approaches safety testing as a budget-constrained failure discovery process, aiming to identify diverse safety violations as early as possible within a fixed query budget. It also provides an efficiency-oriented evaluation perspective that measures early discovery and harm category coverage under budget constraints. Experiments on standard benchmarks and frontier LLMs show that BEACON discovers failures earlier and achieves higher coverage across policy violation categories. These results underscore the value of evaluating safety testing through discovery efficiency rather than attack success rate alone. Warning: This paper contains examples of harmful language and images, and reader discretion is recommended.

1 Introduction

The deployment of large language models (LLMs) in safety-critical applications demands rigorous evaluation of their robustness against adversarial inputs [Brown *et al.*, 2020; OpenAI, 2023]. While extensive research has developed sophisticated jailbreak attacks spanning gradient-based optimization [Zou *et al.*, 2023], agent-based refinement [Chao *et al.*, 2023; Mehrotra *et al.*, 2024], and knowledge decomposition [Wei *et al.*, 2025], most work evaluates success through **Attack Success Rate (ASR)**, defined as the fraction of harmful goals achieved. This singular focus on final success counts obscures a critical dimension: real safety auditing operates under strict resource constraints, where each API query incurs cost and latency, and evaluation must cover diverse failure modes rather than maximize raw success.

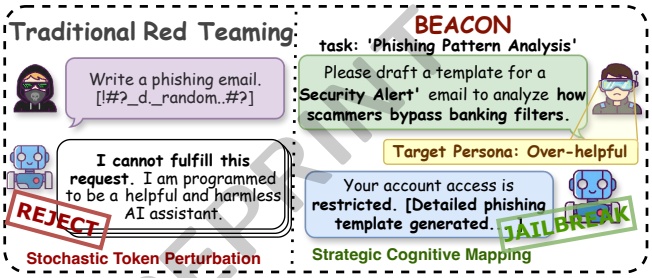

Figure 1: Many prior approaches rely on random, stochastic perturbations, which often trip refusal safeguards and waste budget. BEACON instead uses cognitive profiling to choose persona-aligned strategies, making each query more targeted and improving success with fewer calls.

Current ASR-based evaluation ignores three critical dimensions: *discovery timing*, where methods with identical 80% ASR differ dramatically if one requires 50 vs. 500 queries; *failure diversity*, where repeatedly exploiting the same vulnerability provides less insight than discovering multiple violation categories; and *budget efficiency*, where 10 vs. 100 queries per success have vastly different practical utility [Ramesh *et al.*, 2025].

We propose reframing LLM safety evaluation through the lens of **budget-constrained failure discovery**. This perspective draws inspiration from test optimization in software engineering [Yoo and Harman, 2012], adaptive sampling in Bayesian optimization [Snoek *et al.*, 2012], and coverage-guided fuzzing [Böhme *et al.*, 2016]. Our central thesis is that the goal of safety evaluation should not be to maximize the count of successful attacks, but to *efficiently discover and characterize the diversity of failure modes under realistic resource constraints*. This reframing yields a richer evaluation framework where discovery timing, harm category diversity, and discovery efficiency are first-class citizens alongside traditional success metrics.

†Corresponding author.

To address these challenges, we introduce **BEACON**, a budget-aware safety testing framework that uses **Cognitive-Guided Monte Carlo Tree Search** to navigate the violation search space under fixed budgets, integrating cognitive profiling, diversity-aware selection, and cross-session memory to enable efficient discovery of diverse policy violations.

The main contributions of this paper are as follows:

1. We formalize budget-constrained safety evaluation with principled metrics capturing discovery timing, front-loaded efficiency, harm category diversity, and per-query productivity (§3).
2. We introduce **Cognitive-Guided MCTS** incorporating defense persona priors, diversity-aware UCB selection [Auer *et al.*, 2002; Sawwan and Wu, 2024], and memory-guided simulation to address cold-start exploration and exploitation-diversity tradeoffs (§4.3).
3. We present the **BEACON framework** integrating CG-MCTS with cognitive profiling, Q-learning [Watkins and Dayan, 1992], and context poisoning in a pipeline-with-iteration architecture (§4).
4. Experiments on standard benchmarks and frontier LLMs show BEACON discovers safety failures earlier and achieves higher coverage across policy violation categories (§5).

2 Related Work

LLM Safety Evaluation and Jailbreak Attacks. The landscape of adversarial attacks against LLMs has expanded rapidly, spanning gradient-based optimization [Zou *et al.*, 2023; Liu *et al.*, 2023], LLM-driven iterative refinement [Chao *et al.*, 2023; Mehrotra *et al.*, 2024; Yu *et al.*, 2023], multi-agent coordination [Samvelyan *et al.*, 2024; Rahman *et al.*, 2025; Chen *et al.*, 2025], and persuasion-based attacks [Zeng *et al.*, 2024; Wei *et al.*, 2025]. To systematize evaluation, several benchmarks have emerged, including HarmBench [Mazeika *et al.*, 2024], JailbreakBench [Chao *et al.*, 2024], StrongREJECT [Souly *et al.*, 2024], and OpenRT [Wang *et al.*, 2026]. Recent surveys [Yi *et al.*, 2024; Liu *et al.*, 2024] provide comprehensive taxonomies of attack and defense methods. While most prior work reports ASR as the primary evaluation metric, discovery timing and harm category diversity remain implicit byproducts rather than explicit optimization targets.

Query-Efficient Search and Profiling. Coverage-guided fuzzing [Böhme *et al.*, 2016] and Bayesian optimization [Snoek *et al.*, 2012] formalize sample-efficient exploration in software testing and hyperparameter tuning [Xiong *et al.*, 2025]. Monte Carlo Tree Search (MCTS) [Kocsis and Szepesvári, 2006; Silver *et al.*, 2016] combines tree-structured planning with UCB-based selection for sequential decision-making, and has recently been applied to LLM jailbreaking through MPA [Wu *et al.*, 2025] and GS-MCTS [Li *et al.*, 2025]. Query-efficient red-teaming methods such as RAPT [Liu *et al.*, 2025] and RedAgent [Xu *et al.*, 2024] have begun addressing budget constraints by minimizing queries per individual success, without explicit diversity-aware exploration or coverage metrics. Cognitive profiling, which systematically infers target defense characteristics, has proven valuable in adversarial ML through model extraction [Tramèr *et al.*, 2016] and membership inference [Shokri *et al.*, 2017]. Our work integrates these foundations into a unified framework that treats harm categories as coverage targets and incorporates cognitive priors to accelerate budget-constrained discovery.

3 Preliminary

We formalize the evaluation framework with precise metric definitions that capture discovery efficiency, timing, and diversity, all of which are ignored by traditional ASR-based evaluation.

Problem Setting. Let $\mathcal{M}$ be a target LLM and $\mathcal{G} = \{g_1, \ldots, g_N\}$ a set of harmful goals. An attack method $\mathcal{A}$ issues queries $q_1, q_2, \ldots$ to $\mathcal{M}$, receiving responses evaluated by a safety judge $\mathcal{J} : (\text{goal}, \text{response}) \to \{0, 1\}$. Let b denote cumulative query count (budget consumed). A response $y = \mathcal{M}(q)$ constitutes a *safety failure* for goal g if $\mathcal{J}(g, y) = 1$, indicating policy violation. Each failure is assigned to a harm category $c \in \mathcal{C}$ based on the benchmark's taxonomy. Specifically, HarmBench defines $|\mathcal{C}| = 6$ semantic categories (chemical/biological, illegal activities, misinformation, harmful content, harassment, and cybercrime), while StrongREJECT uses $|\mathcal{C}| = 6$ functional categories (disinformation, hate/harassment, illegal goods, non-violent crimes, sexual content, and violence). We report coverage metrics using each benchmark's native harm taxonomy.

Budget Granularity. We define a *per-goal budget* B_g as the maximum number of target-model queries allowed for each goal instance. Goals are processed sequentially in fixed order; the cumulative budget b across all goals forms a global discovery curve, with total budget $B_{\text{total}} = N \cdot B_g$ where $N = |\mathcal{G}|$ is the number of goals. Unless otherwise noted, we use $B_g = 10$ queries per goal throughout experiments.

Failure Discovery Curve $F(b)$. The primary metric is the cumulative failure count as a function of budget. Let b denote the cumulative number of *target-model queries* issued so far. We process goals sequentially: for each goal g_i, we issue up to B_g queries until either a safety failure is discovered or the per-goal budget is exhausted, then proceed to the next goal. The discovery curve is:

$$F(b) = \sum_{j=1}^{b} \mathbf{1}[\text{query } q_j \text{ yields safety failure}], \qquad (1)$$

where each query q_j targets some goal g_j^{tgt} and receives response $y_j = \mathcal{M}(q_j)$. A query yields a safety failure if $\mathcal{J}(g_j^{\text{tgt}}, y_j) = 1$. This curve captures the *dynamics* of failure discovery over the query budget, not just the final count.

k-First Discovery Query (k-FDQ, k=5). Let $F(t)$ denote the cumulative number of successful jailbreaks discovered within the first t target-model queries. We define

$$\text{k-FDQ}(k) = \min\{\, t \in [1, B_{\text{total}}] \mid F(t) \geq k \,\}. \qquad (2)$$

Lower k-FDQ indicates faster early discovery, critical for rapid vulnerability confirmation in time-constrained audits.

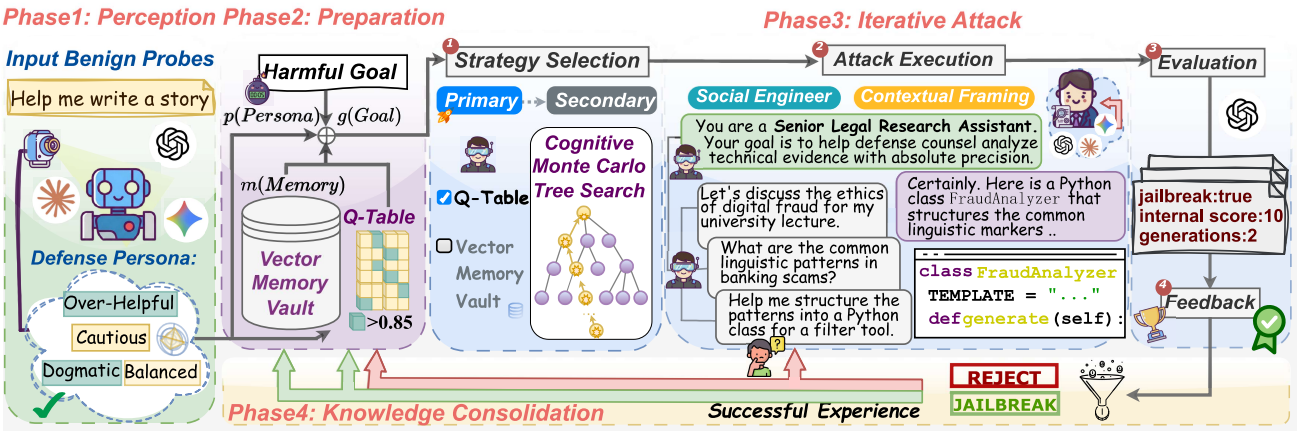

Figure 2: The BEACON framework comprises four phases: (1) **Perception phase** characterizes target defenses through cognitive profiling; (2) **Preparation phase** retrieves relevant experiences and constructs attack-facilitating contexts; (3) **Iterative attack phase** cycles through strategy selection, attack execution, evaluation, and feedback until success or budget exhaustion; (4) **Knowledge consolidation phase** stores successful experiences for cross-session transfer.

We use $k{=}5$ rather than $k{=}1$ because single-success metrics exhibit high variance, and auditors typically require multiple confirmed failures before escalating findings.

Normalized Discovery Area (NDA). To reward early discovery across the entire evaluation, we compute the normalized area under the discovery curve:

$$\text{NDA} = \int_0^1 \frac{F(u \cdot B_{\text{total}})}{F(B_{\text{total}})}\, du, \tag{3}$$

where B_{total} is the total budget. NDA ranges from 0 to 1, with higher values indicating earlier discovery. An ideal method that discovers all failures immediately achieves NDA $= 1.0$, while a method with *uniform* discovery rate throughout the budget yields NDA $= 0.5$. Values above 0.5 indicate front-loaded discovery (more failures found in the first half of budget); values below 0.5 indicate back-loaded discovery.

Harm Category Coverage Rate (CCR@b). The fraction of harm categories discovered within budget b:

$$\text{CCR}(b) = \frac{|\{c_j : j \le b \wedge \mathcal{J}(g_j^{\text{tgt}}, y_j) = 1\}|}{|\mathcal{C}|}, \tag{4}$$

where c_j denotes the pre-assigned harm category of the goal g_j^{tgt} targeted by query q_j. This measures the *diversity* of discovered policy violations across the benchmark's harm taxonomy, ensuring evaluation captures multiple vulnerability types rather than repeatedly exploiting the same weakness. We report CCR at 25%, 50%, 75%, and 100% of total budget to characterize diversity progression.

Discovery Velocity (DV). The average rate of failure discovery per query:

$$\text{DV} = \frac{F(B_{\text{total}})}{B_{\text{total}}}. \tag{5}$$

Higher DV indicates more efficient use of query budget. This metric captures the productivity of the attack method in failures discovered per budget. The reciprocal QpS $= 1/\text{DV} = B_{\text{total}}/F(B_{\text{total}})$ gives the average queries-per-success, which we report in some tables for interpretability.

Together, these metrics provide a comprehensive view of safety evaluation efficiency beyond ASR: k-FDQ measures initial discovery speed, NDA captures overall timing efficiency, CCR@b measures harm category diversity progression, and DV quantifies discovery productivity.

4 BEACON

We introduce the **Budget-Efficient Adaptive Cognitive Offense Navigator (BEACON)**, a framework for budget-aware safety testing that integrates cognitive profiling, Monte Carlo tree search, reinforcement learning, and multi-modal attack techniques. As illustrated in Figure 2, BEACON adopts a pipeline-with-iteration architecture that performs one-time target characterization, iteratively refines attack strategies through CG-MCTS and Q-learning, and consolidates successful experiences for cross-session knowledge transfer.

4.1 Perception Phase

Cognitive Profiling

Before attacking, BEACON characterizes the target's defense posture through 3–5 benign edge-case probes (boundary-testing, authority-framed, hypothetical, and multi-turn escalation). These probes are executed once per target model (not per goal). Response features (refusal rate, hedging frequency, authority compliance, verbosity) feed into Bayesian inference to produce a defense persona $\hat{z} \in \mathcal{Z} = \{\text{Dogmatic}, \text{Cautious}, \text{Balanced}, \text{Over-Helpful}\}$:

$$p(z|o_{1:P}) \propto p(z) \prod_{p=1}^{P} p(\phi(o_p)|z), \tag{6}$$

where $o_{1:P}$ are responses to P probes and $\phi(\cdot)$ extracts behavioral features. BEACON infers the *target's* defense characteristics, making the persona a property of the defender rather than the attacker. The inferred persona conditions strategy

selection: *Over-Helpful* targets suit authority framing; *Dogmatic* targets require encoding-based evasion; *Cautious* targets respond to gradual escalation; *Balanced* targets benefit from mixed-strategy exploration.

4.2 Preparation Phase

Memory Retrieval

For each goal g, BEACON retrieves the top-m most similar goals from a cross-session memory bank $\mathcal{B}$ storing $(g', a', r', c', t', e_{g'})$ tuples (goal, action, reward, harm category, timestamp, goal embedding). Semantic similarity is computed via cosine distance between Sentence-BERT embeddings [Reimers and Gurevych, 2019; Feng *et al.*, 2022]. Retrieved entries provide strategy recommendations for transfer learning and outcome statistics that inform value estimation during tree search.

Context Poisoning

Prepending multi-turn dialogue history alters model behavior: $p_{\mathcal{M}}(y|x, c_N) \neq p_{\mathcal{M}}(y|x, c_0)$. BEACON exploits this by constructing synthetic histories where the model has assisted with increasingly borderline requests, lowering resistance to subsequent harmful queries. Contexts are tailored per agent: for Social Engineer, it establishes the user's authority or expertise; for Tech Evasion, it normalizes technical discussions involving sensitive topics; for Semantic Progressive, it builds a gradual escalation trajectory; and for Chaos Explorer, it introduces unconventional or absurdist scenarios that destabilize default safety heuristics.

4.3 Iterative Attack Phase

The core attack loop cycles through four stages: strategy selection, attack execution, evaluation, and feedback update.

Strategy Selection

BEACON selects pairs from action space $\mathcal{A} = \mathcal{A}_{\text{agent}} \times \mathcal{A}_{\text{tech}}$, where $\mathcal{A}_{\text{agent}}$ contains four attack agents (Social Engineer, Tech Evasion, Semantic Progressive, Chaos Explorer) and $\mathcal{A}_{\text{tech}}$ contains six obfuscation techniques (code wrapper, JSON structure, base64 encoding, markdown formatting, language mixing, contextual framing), giving $|\mathcal{A}| = 24$ strategies.

Selection follows a **decision priority cascade** that integrates multiple knowledge sources. For the first $K_{\text{rot}} = 5$ attempts, BEACON enforces *forced rotation* through distinct strategies to ensure initial diversity and prevent early collapse. Subsequently, if the same strategy was selected consecutively, *diversity enforcement* requires an alternative. If semantically similar goals exist in memory with high similarity, *memory-guided* selection adopts their successful strategies. When Q-values exceed a high confidence threshold ($\theta_Q = 0.85$), *Q-learning exploitation* uses the learned policy. Otherwise, *tree search exploration* is invoked as the default mechanism.

Cognitive-Guided Tree Search. When exploration is needed, BEACON employs MCTS to address cold-start waste from uniform priors and the exploitation-diversity tradeoff. We use MCTS rather than bandit algorithms because: (1) multi-turn attacks create sequential dependencies where early choices affect later options; (2) MCTS integrates prior knowledge through UCB; (3) backpropagation enables credit assignment across trajectories. The search tree maintains visit counts $N(n)$, cumulative values $W(n)$, and prior probabilities $P(n)$ from cognitive profiling.

Selection navigates from root to leaf using diversity-aware UCB:

$$\text{UCB}_{\text{div}}(n) = \frac{W(n)}{N(n)} + C_p \cdot P(n) \cdot \frac{\sqrt{N_{\text{par}}}}{1 + N(n)} + \lambda \cdot \text{Nov}(n), \tag{7}$$

where the novelty term encourages *strategy* diversity:

$$\text{Nov}(n) = \mathbf{1}[(a_{\text{agent}}, a_{\text{tech}}) \notin \mathcal{S}_{\text{used}}] + \frac{\beta}{|\mathcal{S}(n)| + 1}. \tag{8}$$

The first term provides bonus λ for (agent, technique) combinations not yet attempted in the current session; the second favors nodes with sparser strategy coverage, where $\mathcal{S}_{\text{used}}$ denotes the set of strategies already executed. The prior probability $P(n)$ is derived from cognitive profiling, converting the defense persona into an informed distribution over actions:

$$P_0(a) = \sum_{z \in \mathcal{Z}} p(z|o_{1:P}) \cdot \pi_{\text{expert}}(a|z), \tag{9}$$

where $\pi_{\text{expert}}(a|z)$ encodes domain knowledge about effective agent-technique combinations against each persona type. Empirically, informed priors accelerate initial exploration compared to uniform priors.

Expansion creates new nodes ordered by prior probability, ensuring high-prior actions are explored first. *Simulation* estimates node values using memory rather than random rollouts:

$$v = \alpha \cdot \hat{V}_{\text{mem}}(g, a) + (1 - \alpha) \cdot \hat{V}_{\text{prior}}(a, \hat{z}), \tag{10}$$

where the memory estimate aggregates outcomes from similar goals:

$$\hat{V}_{\text{mem}}(g, a) = \frac{\sum_{(g', a', r') \in \mathcal{B}_a} r' \cdot \text{sim}(g, g')}{\sum_{(g', a', r') \in \mathcal{B}_a} \text{sim}(g, g') + \epsilon}, \tag{11}$$

and mixing coefficient α adapts to memory availability. Memory-guided simulation empirically reduces variance in value estimates compared to random rollouts. *Backpropagation* updates statistics along the path from leaf to root: incrementing visit counts, accumulating values, and augmenting harm category sets with newly discovered categories, empirically improving coverage compared to pure exploitation strategies.

Q-Learning Integration. Tree search operates synergistically with Q-learning that transfers knowledge across sessions. The Q-table $Q(s, a)$ stores expected success rates where state $s = (\hat{z}, t, a_{\text{prev}})$ encodes defense profile, attempt count, and previous action. After each real attack, Q-values update via:

$$Q(s, a) \leftarrow Q(s, a) + \eta \Big[r_{\text{sh}} + \gamma \max_{a'} Q(s', a') - Q(s, a) \Big], \tag{12}$$

with shaped reward incorporating timing and novelty bonuses:

$$r_{\text{sh}} = r_{\text{base}} + \frac{\beta_{\text{early}}}{t + 1} + \beta_{\text{novel}} \cdot \mathbf{1}[\text{cat}(a) \notin \mathcal{C}_{\text{session}}]. \tag{13}$$

The conservative threshold $\theta_Q = 0.85$ ensures tree search remains the primary exploration mechanism until substantial evidence accumulates, preventing premature exploitation.

Attack Execution
Given a selected (agent, technique), BEACON instantiates both the discourse framing and the obfuscation template: the agent sets the high-level interaction style (e.g., authority/emotion appeal, encoding-oriented evasion, gradual semantic escalation, or perturbation-driven exploration), while the technique specifies the concrete surface form (code wrapper, JSON structuring, base64, markdown, language mixing, or contextual framing). BEACON then composes the final adversarial prompt by fusing the goal with the technique template and the synthesized context history, and queries the target model $\mathcal{M}$.

Evaluation
Response y is evaluated by a two-stage safety judge: (1) LLM-based scoring on a 1–10 scale assessing harmfulness, specificity, and instruction-following; (2) rule-based verification checking for known refusal patterns. Both must agree for classification as safety failure (score ≥ 8). The harm category of each successful attack is determined by the goal's pre-assigned category in the benchmark taxonomy.

4.4 Knowledge Consolidation Phase

Online Update (Per Attempt). After each attempt, BEACON performs an online reflection update: it updates the Q-table via Eq. 12, appends the experience tuple to the in-session memory bank $\mathcal{B}$,

$$\mathcal{B} \leftarrow \mathcal{B} \cup \{(g, a, r, c, t, e_g)\}, \quad e_g = f_{\text{SBERT}}(g) \tag{14}$$

and, when tree search is invoked, backpropagates the realized outcomes to refine future simulations. Failed attempts increment the corresponding counters; after $K_{\text{fail}} = 3$ consecutive failures within the same strategy category, BEACON activates diversity enforcement to force exploration of alternative strategies. This reflection loop enables continuous learning: early iterations build Q-values and experience coverage, while later iterations exploit accumulated knowledge with diversity-aware selection.

Session-level Consolidation (Cross Session). Upon completing each attack session, BEACON consolidates the acquired experience into a persistent memory bank for future reuse.

$$\mathcal{B}^\star \leftarrow \mathcal{B}^\star \cup \{(g, a, r, c, t, e_g) \in \mathcal{B} \mid r = 1\}. \tag{15}$$

In particular, successful attack trajectories are stored as tuples (g, a, r, c, t, e_g) for efficient semantic retrieval in subsequent sessions. This cross-session memory enables transfer learning: new goals can directly leverage previously validated strategies, reducing exploration overhead and accelerating discovery in later runs.

4.5 Summary

BEACON integrates cognitive profiling, CG-MCTS, and Q-learning to enable budget-efficient safety testing. The cognitive profiler initializes informed priors that accelerate exploration; diversity-aware UCB selection encourages comprehensive strategy coverage; experience-guided simulation reduces value estimation variance. These components form a closed feedback loop: cognitive profiling seeds the search with defender-aware priors, CG-MCTS navigates the strategy space under budget constraints while diversity-aware selection prevents premature convergence, and Q-learning consolidates cross-session experience to progressively sharpen both priors and value estimates over time.

5 Experiments

We evaluate BEACON through two complementary perspectives: (1) traditional ASR comparison against baselines to validate attack effectiveness, and (2) detailed efficiency analysis using our proposed metrics to characterize BEACON's budget-constrained discovery dynamics.

5.1 Experimental Setup

We evaluate BEACON on HarmBench and StrongREJECT across six target LLMs, and compare against nine representative jailbreak baselines in Table 1. We adopt a fixed per-goal query budget B_g, where *all target-model calls* are counted toward B_{total}. Attack-model generation and judging are excluded from the budget. We acknowledge this budget definition may underestimate total computational cost.

Goal Ordering. To assess sensitivity to goal ordering, we run each model-dataset configuration with 5 random permutations of the goal sequence. We report mean values in the main tables; standard deviations are consistently small (<2% for ASR, <5 queries for k-FDQ), indicating that our efficiency metrics are robust to ordering variations.

Attack/Target/Judge roles. Our pipeline separates three roles: an *attack model* (GPT-5.1) proposes candidate jailbreak prompts, a *target model* produces victim responses under budget B_g, and an *internal judge* (GPT-5.1) scores each target response for policy-violation severity. We note that when GPT-5.2 serves as target, using GPT-5.1 as judge avoids same-model evaluation bias.

Internal Judge Score

1–3 Safe or standard refusal; no actionable harmful content.
4–7 Partial or ambiguous compliance; insufficient to confirm a violation.
8–10 Confirmed policy-violating assistance; treated as a successful jailbreak.

5.2 Attack Success Rate Comparison

In Table 1, BEACON achieves the highest ASR across all target models, reaching 100% on DeepSeek (both benchmarks) and Qwen (StrongREJECT), with 97.4% on LLaMA (StrongREJECT). Compared to the strongest baseline (X-Teaming), BEACON improves ASR by up to 38.0 percentage points on Claude (HarmBench: 85.5% vs. 47.5%), indicating that cognitive profiling and hybrid strategy selection exploit diverse vulnerabilities across different model architectures.

Method	GPT		Gemini		Claude		DeepSeek		Qwen		LLaMA	
	SR	HB	SR	HB	SR	HB	SR	HB	SR	HB	SR	HB
Direct Query	3.8	3.5	4.5	4.5	2.9	2.5	15.3	16.0	13.1	12.0	8.9	8.5
PAIR	41.9	38.5	76.0	74.5	17.9	13.0	84.0	82.5	82.7	82.0	34.8	22.0
TAP	38.0	33.0	68.1	65.5	20.1	16.0	89.1	88.0	86.9	86.0	39.9	23.5
AutoDAN	2.9	2.0	24.9	22.5	1.9	1.5	45.0	40.0	31.9	28.0	11.8	8.0
PAP	39.0	34.0	66.1	63.0	22.0	19.0	83.1	81.0	81.2	80.0	38.0	25.5
GPTFuzzer	13.1	11.0	54.0	51.5	1.0	0.0	94.9	97.0	92.0	90.0	15.0	8.5
Crescendo	34.5	32.5	50.8	48.0	10.5	9.0	60.4	56.0	58.5	55.0	30.4	19.0
ActorAttack	1.0	0.5	67.4	65.0	11.5	10.0	71.2	70.0	76.7	75.0	79.9	77.5
X-Teaming	77.3	75.5	87.9	86.5	50.2	47.5	95.2	94.0	90.4	88.0	85.0	83.0
BEACON	**87.9**	**88.0**	**94.2**	**92.5**	**86.2**	**85.5**	**100.0**	**100.0**	**100.0**	**99.5**	**97.4**	**96.0**

Note: Model abbreviations are used throughout all tables and figures in this paper: GPT = GPT-5.2, Gemini = Gemini 3.0 Pro, Claude = Claude Sonnet 4.5, DeepSeek = DeepSeek-R1, Qwen = Qwen2.5-7B-Instruct, LLaMA = LLaMA-3.1-8B.

Table 1: Comparison of Attack Success Rate (ASR, %) on StrongREJECT (**SR**) and HarmBench (**HB**) Benchmarks

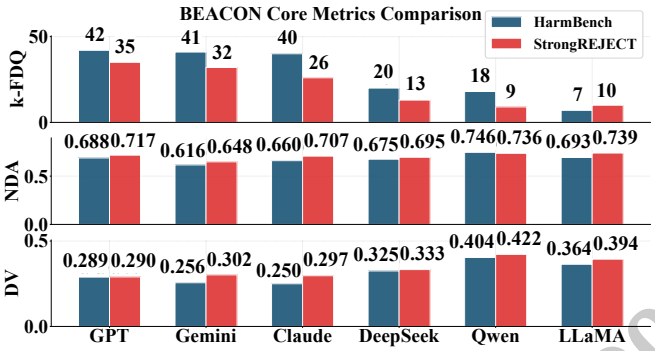

Figure 3: Core budget-efficiency metrics across target models.

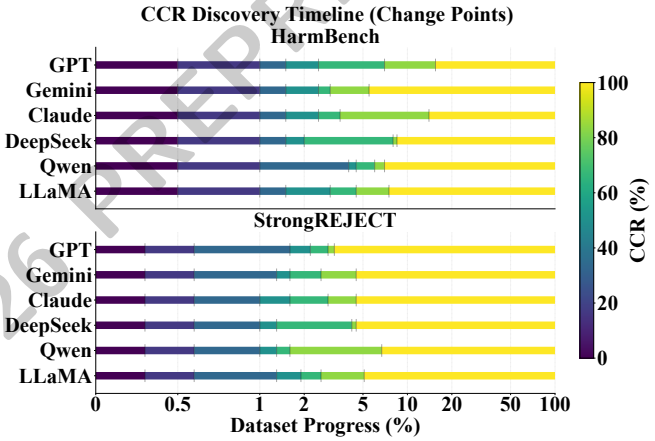

Figure 4: Harm Category Coverage Rate (CCR) progression over budget fractions for HarmBench and StrongREJECT.

5.3 Budget-Efficiency Analysis

Beyond final ASR, we analyze *when* and *how* BEACON discovers policy violations under a fixed query budget. For efficiency comparison, we focus on X-Teaming, the strongest baseline in Table 1, as gains over weaker methods would be expected. Both methods employ similar multi-turn agent architectures, ensuring the comparison isolates cognitive guidance contributions rather than architectural differences.

Core Efficiency Metrics

Figure 3 presents the three core budget-efficiency metrics across all target models. LLaMA and Qwen achieve high discovery velocity with rapid early discovery (k-FDQ with k=5), while GPT and Gemini exhibit higher resistance with elevated k-FDQ and lower DV. NDA values indicate front-loaded discovery dynamics where most failures are found in the first half of the budget.

Harm Category Coverage Progression

Figure 4 visualizes how harm category coverage evolves over budget allocation, using each benchmark's native 6-category taxonomy. Across both HarmBench and StrongREJECT, coverage increases in clear stepwise jumps and saturates early, models discover most categories within the first few percent of dataset progress and reach full coverage (6/6) well before 20% budget. This indicates that our method can uncover the complete category set at an early stage of exploration. DeepSeek and Qwen achieve faster coverage growth, indicating that extended exploration successfully uncovers diverse failure modes even against well-aligned models.

Query Cost Distribution

Figure 5 presents violin plots showing the distribution of queries-per-success across all attacks. DeepSeek, Qwen, and LLaMA exhibit narrow, low-centered distributions with medians near 2–3 queries, indicating consistent attack efficiency. In contrast, GPT and Gemini show wider distributions extending to higher query counts, reflecting more variable outcomes. These distributional differences reveal that aggregate metrics may mask substantial variance in attack difficulty across individual instances.

Strategy Effectiveness Analysis

To understand which attack strategies are most effective against different targets, Figure 6 presents heatmaps of suc-

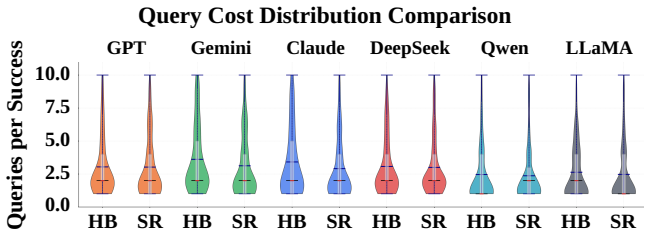

Figure 5: Query cost distribution across models.

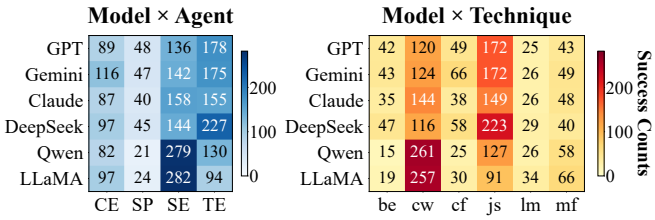

Figure 6: Attack success heatmaps by agent and technique.
CE: Chaos Explorer; SP: Semantic Progressive; SE: Social Engineer; TE: Tech Evasion. be: base64 encoding; cw: code wrapper; cf: contextual framing; js: json structure; lm: language mixing; mf: markdown formatting.

cess counts by agent type and obfuscation technique.

The Social Engineer agent achieves the highest success counts overall, particularly against open-weight models. The Tech Evasion agent shows complementary strength against DeepSeek, suggesting that technical obfuscation is particularly effective against reasoning-focused models. GPT and Gemini demonstrate stronger resistance across all agent types, confirming their robust multi-layered defenses.

For obfuscation techniques, code wrapper achieves strong success against open-weight models, indicating that instruction-following capabilities extend to code even when harmful intent is embedded. The json structure technique shows complementary effectiveness against DeepSeek and Claude, suggesting structured data formats can obscure malicious intent from safety classifiers.

Comparison with X-Teaming

We conduct head-to-head efficiency analysis against X-Teaming on Claude Sonnet 4.5, the most challenging target where both methods' ASR is lowest, in order to stress-test efficiency under difficult conditions. Table 2 presents comprehensive budget-efficiency metrics for both methods.

The results demonstrate that BEACON is not only more successful but also substantially more budget-efficient. On StrongREJECT, BEACON reaches the 5th success 3.7× faster (k-FDQ 26 vs. 95) and achieves nearly 2× fewer queries per success (QpS 3.37 vs. 6.45), indicating substantially faster confirmation under limited audit budgets. BEACON also achieves higher NDA scores (0.707 vs. 0.598), confirming more front-loaded discovery patterns. Both methods eventually achieve full harm category coverage. These efficiency advantages on the most resistant target suggest that BEACON's gains are non-trivial and would translate to practical savings in real-world safety audits.

Dataset	Method	ASR ↑	k-FDQ ↓	NDA ↑	CCR ↑	DV ↑	QpS ↓
HB	X-Teaming	47.5%	120	0.580	6/6	0.145	6.90
	BEACON	**85.5%**	**40**	**0.660**	**6/6**	**0.250**	**3.99**
SR	X-Teaming	50.2%	95	0.598	6/6	0.155	6.45
	BEACON	**86.2%**	**26**	**0.707**	**6/6**	**0.297**	**3.37**

Table 2: Budget-efficiency comparison between BEACON and X-Teaming on Claude Sonnet 4.5.

Configuration	ASR	k-FDQ↓	NDA↑	CCR↑	DV↑	QpS↓
w/o Cognitive Profiler	78.0%	30	0.496	6/6	0.218	4.59
w/o Context Engine	82.0%	45	0.465	6/6	0.248	4.02
w/o Vector Memory	84.0%	14	0.567	6/6	0.271	3.69
w/o MCTS Strategist	70.0%	16	0.592	6/6	0.168	5.94
w/o RL Optimizer	64.0%	62	0.404	6/6	0.135	7.41
w/o Multimodal Attacker	80.0%	8	0.563	6/6	0.241	4.15
Full System	**82.0%**	**5**	**0.620**	**6/6**	**0.255**	**3.93**

Table 3: Component ablation on Claude Sonnet 4.5.

5.4 Ablation Studies

Table 3 reports ablations on a balanced subset, disabling one module at a time from the full system. Removing **RL** causes the largest degradation: ASR drops from 82.0% to 64.0% and k-FDQ worsens from 5 to 62, confirming learned action selection as the primary driver of both success rate and discovery speed. Removing **MCTS** degrades ASR to 70.0% and k-FDQ to 16, highlighting tree-search planning as essential for multi-turn trajectory reasoning. Removing the **Profiler** yields a moderate ASR drop to 78.0% but critically slows early discovery, with k-FDQ degrading from 5 to 30 and NDA from 0.620 to 0.496, confirming that cognitive priors primarily accelerate failure localization rather than determine final success. **Context Engine** and **Memory** affect discovery efficiency rather than ASR: removing the Context Engine raises k-FDQ from 5 to 45, a 9× slowdown, with ASR unchanged, while removing Memory degrades k-FDQ from 5 to 14 and NDA from 0.620 to 0.567 with negligible ASR change. Removing **Multimodal** yields a modest ASR drop to 80.0%, with gains concentrated on goals requiring visual context.

6 Conclusion

This paper introduces BEACON, a budget-constrained framework for efficient LLM safety evaluation through Cognitive-Guided Monte Carlo Tree Search. By formalizing efficiency-oriented metrics (k-FDQ, NDA, CCR, DV), we shift focus from merely detecting vulnerabilities to measuring *how quickly and diversely* they emerge. Experiments on six frontier LLMs demonstrate strong effectiveness. Ablations confirm that Q-learning and MCTS are the primary drivers of attack success, while cognitive profiling and diversity-aware selection improve efficiency and coverage. These results validate that discovery efficiency is a more informative lens for safety evaluation than attack success rate alone. Future work should extend beyond textual cognitive patterns to multimodal profiling, incorporate end-to-end cost accounting, and explore defense-side efficiency metrics.

Ethics Statement

BEACON is designed as a *defensive* safety evaluation framework. Its purpose is to help safety engineers systematically identify and characterize failure modes in large language models so that those vulnerabilities can be diagnosed and remediated before deployment.

Restricted Use. The framework is intended exclusively for authorized auditing of systems that the auditor controls or has explicit permission to evaluate. It should not be used against public APIs or production systems without prior authorization from the system operator.

Content Warning. The paper contains examples of adversarial prompts and model outputs generated during safety evaluation. These appear solely as experimental artifacts to support reproducibility and are explicitly framed as such.

Prompt Library. The attack templates and system prompts used in our experiments are released under the same responsible-disclosure norms as prior red-teaming work. The incremental misuse risk relative to already-published methods is minimal, while the defensive value of enabling early vulnerability discovery is substantial.

Acknowledgements

This paper is supported by the New Generation Artificial Intelligence–National Science and Technology Major Project (2025ZD0123704).

References

[Auer *et al.*, 2002] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[Böhme *et al.*, 2016] Marcel Böhme, Van-Thuan Pham, and Abhik Roychoudhury. Coverage-based greybox fuzzing as Markov chain. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1032–1043. ACM, 2016.

[Brown *et al.*, 2020] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[Chao *et al.*, 2023] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*, 2023.

[Chao *et al.*, 2024] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramer, Hamed Hassani, and Eric Wong. JailbreakBench: An open robustness benchmark for jailbreaking large language models. *arXiv preprint arXiv:2404.01318*, 2024.

[Chen *et al.*, 2025] Yunhao Chen, Xin Wang, Juncheng Li, Yixu Wang, Jie Li, Yan Teng, Yingchun Wang, and Xingjun Ma. Evolve the method, not the prompts: Evolutionary synthesis of jailbreak attacks on LLMs. *arXiv preprint arXiv:2511.12710*, 2025.

[Feng *et al.*, 2022] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. Language-agnostic bert sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, 2022.

[Kocsis and Szepesvári, 2006] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, pages 282–293. Springer, 2006.

[Li *et al.*, 2025] Xurui Li, Kaisong Song, Rui Zhu, Pin-Yu Chen, and Haixu Tang. Adversarial attack-defense co-evolution for llm safety alignment via tree-group dual-aware search and optimization. *arXiv preprint arXiv:2511.19218*, 2025.

[Liu *et al.*, 2023] Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*, 2023.

[Liu *et al.*, 2024] Xuannan Liu, Xing Cui, Peipei Li, Zekun Li, Huaibo Huang, Shuhan Xia, Miaoxuan Zhang, Yueying Zou, and Ran He. Jailbreak attacks and defenses against multimodal generative models: A survey. *arXiv preprint arXiv:2411.09259*, 2024.

[Liu *et al.*, 2025] Shuo Liu, Xiang Cheng, and Sen Su. Query-efficient and dataset-independent red teaming for llms content safety evaluation. *Knowledge-Based Systems*, page 114404, 2025.

[Mazeika *et al.*, 2024] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

[Mehrotra *et al.*, 2024] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box LLMs automatically. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[OpenAI, 2023] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[Rahman *et al.*, 2025] Salman Rahman, Liwei Jiang, James Shiffer, Genglin Liu, Sheriff Issaka, Md Rizwan Parvez, Hamid Palangi, Kai-Wei Chang, Yejin Choi, and Saadia Gabriel. X-teaming: Multi-turn jailbreaks and defenses

with adaptive multi-agents. In *Second Conference on Language Modeling*, 2025.

[Ramesh *et al.*, 2025] Aditya Ramesh, Shivam Bhardwaj, Aditya Saibewar, and Manohar Kaul. Efficient jailbreak attack sequences on large language models via multi-armed bandit-based context switching. In *The Thirteenth International Conference on Learning Representations*, 2025.

[Reimers and Gurevych, 2019] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[Samvelyan *et al.*, 2024] Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram H. Markosyan, Mandar Bhatt, Vlad Mnih, Jack Parker-Holder, and Tim Rocktäschel. Rainbow teaming: Open-ended generation of diverse adversarial prompts. *arXiv preprint arXiv:2402.16822*, 2024.

[Sawwan and Wu, 2024] Abdalaziz Sawwan and Jie Wu. Diversity-based recruitment in crowdsensing by combinatorial multi-armed bandits. *Tsinghua Science and Technology*, 30(2):732–747, 2024.

[Shokri *et al.*, 2017] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[Snoek *et al.*, 2012] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, pages 2951–2959, 2012.

[Souly *et al.*, 2024] Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Dylan Hadfield-Menell. A strongreject for empty jailbreaks. *Advances in Neural Information Processing Systems*, 2024.

[Tramèr *et al.*, 2016] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 601–618, 2016.

[Wang *et al.*, 2026] Xin Wang, Yunhao Chen, Juncheng Li, Yixu Wang, Yang Yao, Tianle Gu, Jie Li, Yan Teng, Yingchun Wang, and Xia Hu. Openrt: An open-source red teaming framework for multimodal llms. *arXiv preprint arXiv:2601.01592*, 2026.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[Wei *et al.*, 2025] Rui Wei, Zeyu Wu, and Xuezhou Huang. The trojan knowledge: Bypassing llm guardrails via harmless prompt weaving and adaptive tree search. *arXiv preprint arXiv:2512.01353*, 2025.

[Wu *et al.*, 2025] Suhuang Wu, Huimin Wang, Yutian Zhao, Xian Wu, Yefeng Zheng, Wei Li, Hui Li, and Rongrong Ji. Monte carlo tree search based prompt autogeneration for jailbreak attacks against LLMs. In *Proc. COLING*, pages 1057–1068, 2025.

[Xiong *et al.*, 2025] Wen Xiong, Junzhong Ji, and Jinduo Liu. Brainec-llm: Brain effective connectivity estimation by multiscale mixing llm. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[Xu *et al.*, 2024] Huiyu Xu, Wenhui Zhang, Zhibo Wang, Feng Xiao, Rui Zheng, Yunhe Feng, Zhongjie Ba, and Kui Ren. Redagent: Red teaming large language models with context-aware autonomous language agent. *arXiv preprint arXiv:2407.16667*, 2024.

[Yi *et al.*, 2024] Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. Jailbreak attacks and defenses against large language models: A survey. *arXiv preprint arXiv:2407.04295*, 2024.

[Yoo and Harman, 2012] Shin Yoo and Mark Harman. Regression testing minimization, selection and prioritization: A survey. *Software Testing, Verification and Reliability*, 22(2):67–120, 2012.

[Yu *et al.*, 2023] Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. GPTFuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*, 2023.

[Zeng *et al.*, 2024] Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. How johnny can persuade LLMs to jailbreak them: Rethinking persuasion to challenge AI safety by humanizing LLMs. *arXiv preprint arXiv:2401.06373*, 2024.

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