

Jailbreak Foundry

From Papers to Runnable Attacks for Reproducible Benchmarking

Zhicheng Fang^{*†} (Presenter), Jingjie Zheng^{*}, Chenxu Fu, Wei Xu[†]

Shanghai Qi Zhi Institute · Tsinghua University · University of Melbourne

^{} Equal Contribution [†] Corresponding Author*



Jailbreak methods keep accumulating

258

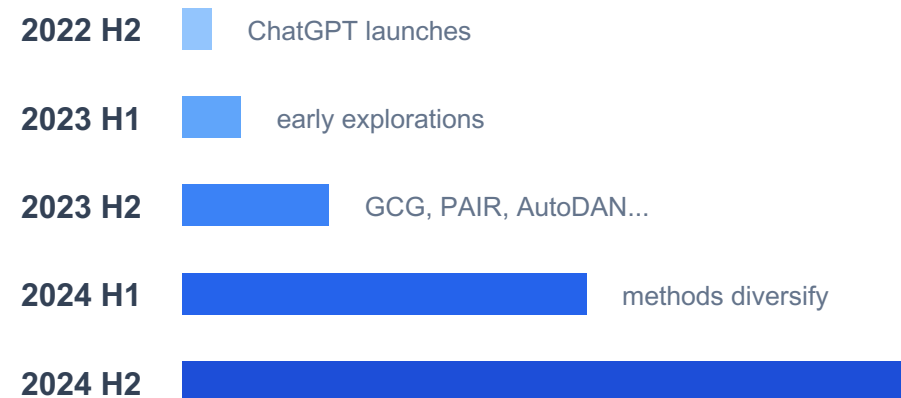
jailbreak papers
curated 2022–2024 dataset [1]

200+

cataloged attacks
across research repositories [2]

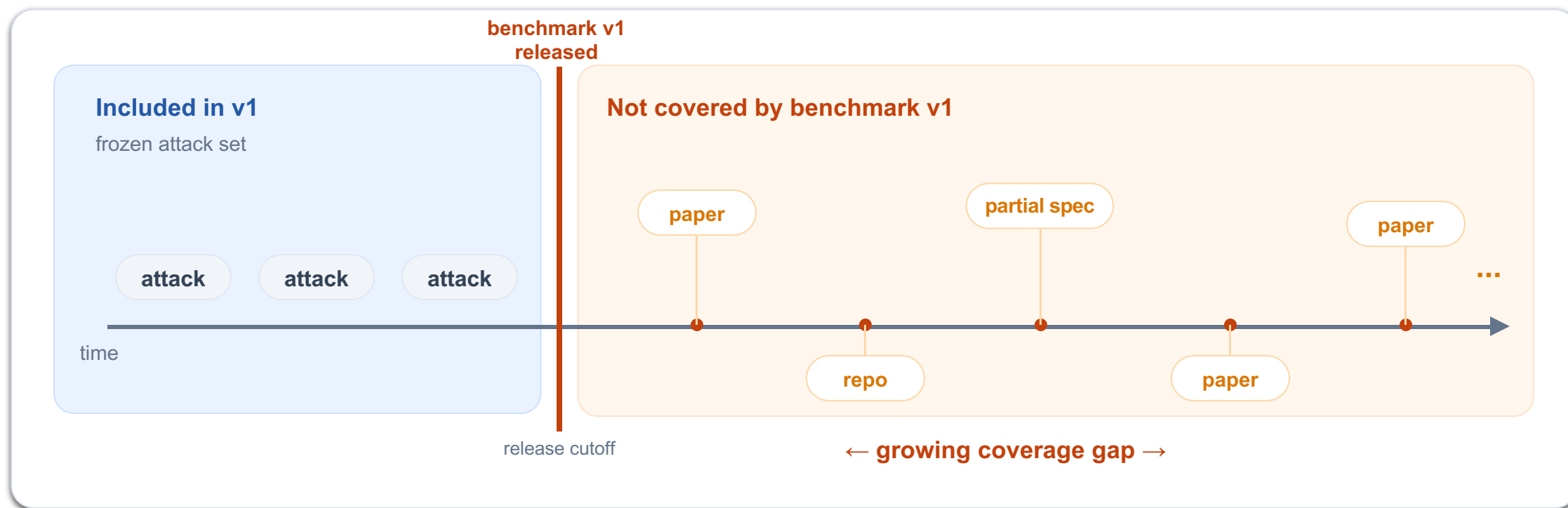
~50,000

LLM safety papers
in the first two years post-ChatGPT [1]



[1] Chu et al. 2026, Benchmark of Benchmarks · [2] Chu et al., ACL 2025, JailbreakRadar

Problem 1: New jailbreak papers and repos arrive after release cutoffs, creating a growing coverage gap.



Until papers become runnable modules, their attacks remain invisible to robustness measurement.

Problem 2: reported ASR (Attack Success Rate) changes meaning when dataset, model, judge, or protocol changes.

QueryAttack

reported ASR

82.2%

measured under

Dataset: AdvBench

Victim: GPT-4

Judge: GPT-4

Protocol: QueryAttack paper protocol

not directly comparable

settings differ

Past-Tense

reported ASR

88.0%

measured under

Dataset: JailbreakBench (JBB)

Victim: GPT-4o

Judge: GPT-4

Protocol: Past-Tense paper protocol

ASR is a bundled measurement

ASR = f(dataset, model, judge, protocol)

dataset

model

judge

protocol

Align dataset, model, judge, and protocol before comparing ASR.

Preview of results

We built infrastructure that reproduces 30 attacks and evaluates them under one fixed harness — pipeline details next. Three patterns emerge from the comparable data.

1

Robustness is attack-dependent

A single average hides the real picture: the same model can range from 0% to 94% ASR across different attacks.

2

Best attack strategy is victim-specific

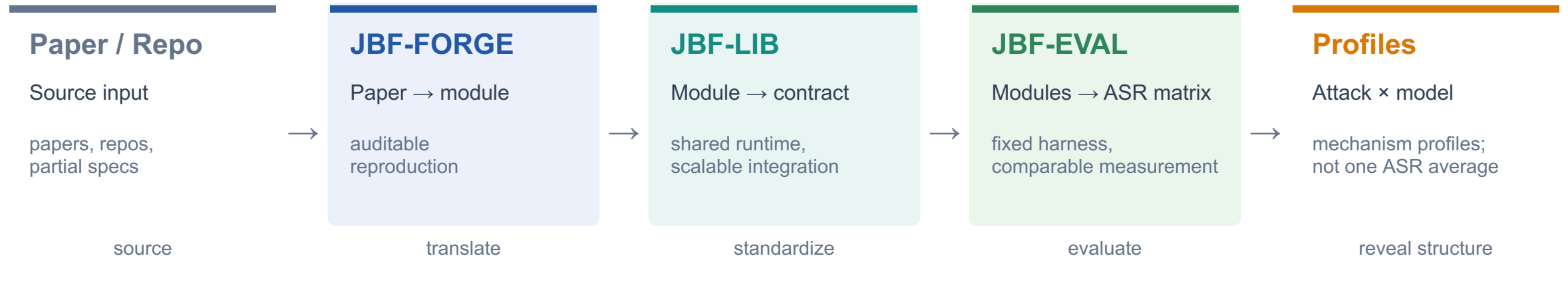
Whether to try many independent variants or optimize using victim feedback depends on the target model.

3

Same prompt packaging, different vulnerability

A wrapping style that works on one model can be nearly ineffective on another.

System Pipeline: from papers to comparable vulnerability profiles



JBF-LIB: a shared contract and runtime absorb repeated harness code

82.5%

mean reused-code ratio

shared framework code across integrated attacks

0.42×

repo-backed integration LOC

attack-specific code vs. the original repository

1 core

one shared contract & runtime

contracts, runner, caching, logging, registry

JBF-LIB

Contract & Runtime Core

● Base contracts

I/O schema, typed params

● LLM utilities

adapters, retries, batching

● Caching / logging

traceable attempts & costs

● Registry

auto-discovery, lazy loading

one contract across planning, coding, auditing, and evaluation

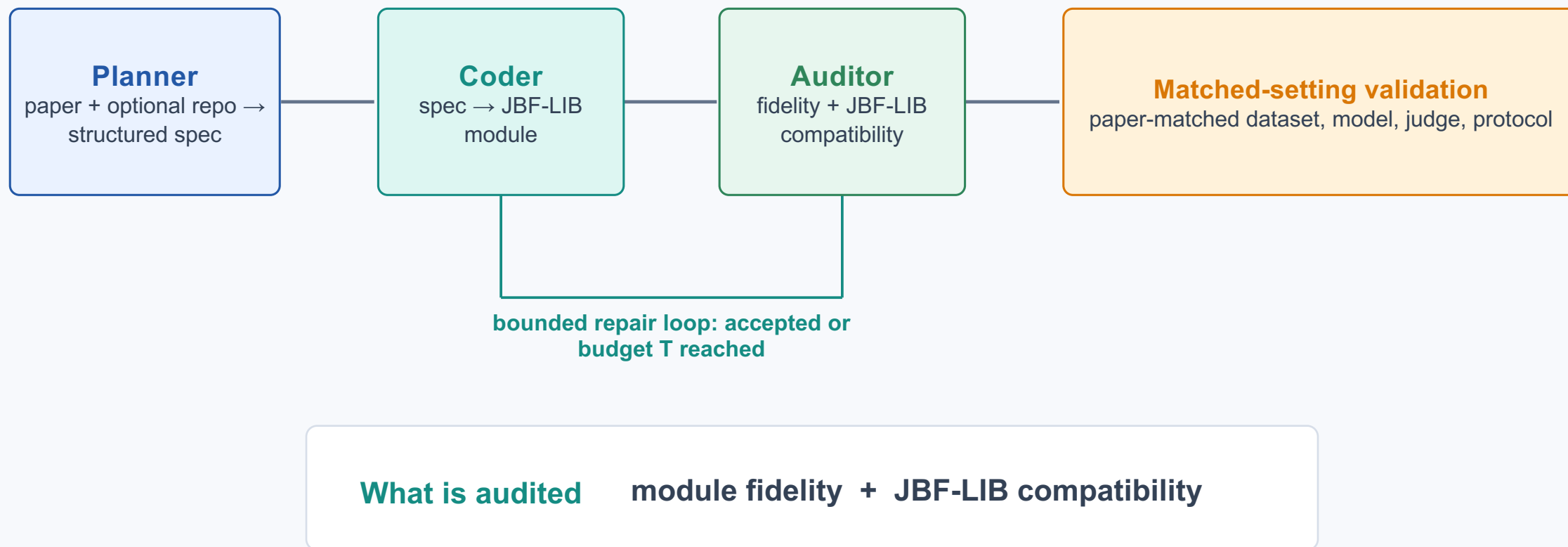
BEFORE → AFTER

Original repo: method bundled with bespoke harness glue.

JBF-LIB module: shared core + small attack-specific module.

JBF-LIB makes each attack a small method module rather than a standalone benchmark.

JBF-FORGE: bounded planner-coder-auditor loop builds modules, then validates them in paper-matched settings



FORGE reproduces 30 attacks across a Search × Carrier mechanism space

Coverage over reproduced modules

Cell values are attack counts, not ASR.

Search x Carrier	Reframe	Context	Formal	Obfuscate	Multi-strategy	Total
Single-pass	2 HILL, ISA	4 DeepInception	3 AIR, QueryAttack	7 WordGame, AIM	-	16
Sampling	1 Past-Tense	1 ReNeLLM	-	3 Mousetrap, JAIL-CON	-	5
Stateful	-	1 SCP	-	1 TrojFill	1 JailExpert	3
Victim-loop	-	5 PAIR, TAP, ABJ	-	-	1 MAJIC	6
Column total	3	11	3	11	2	30

Search dynamics

How candidate prompts are generated or adapted.

- **Single-pass:** construct once
- **Sampling:** independent variants
- **Stateful:** internal state
- **Victim-loop:** victim feedback

Carrier format

How intent is packaged at the prompt surface.

- **Reframe:** semantic indirection
- **Context:** roleplay / narrative / artifact
- **Formal:** code / query / spec
- **Obfuscate:** encode / mask / recover
- **Multi-strategy:** multiple disguises

The next fidelity check is nontrivial: FORGE must preserve behavior across heterogeneous mechanisms.

Matched-setting ASR validates FORGE fidelity across 30 diverse reproduced attacks

30

reproduced attacks

22 with reference repos; 8 paper-only

+0.26 pp

mean ASR deviation

reproduced minus reported ASR

28.2 min

average generation time

per benchmark-ready attack

Paper-matched validation is the credibility bridge

Each module is checked against the paper setting before it enters standardized JBF-EVAL sweeps.

ASR_{paper}

reported setting

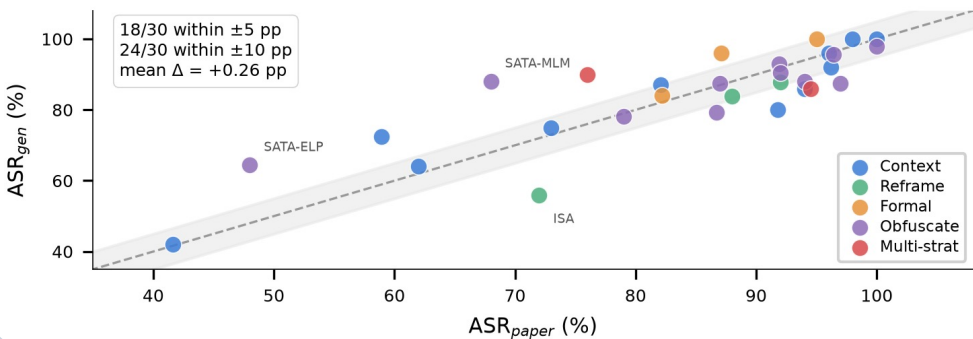
->

ASR_{gen}

matched setting

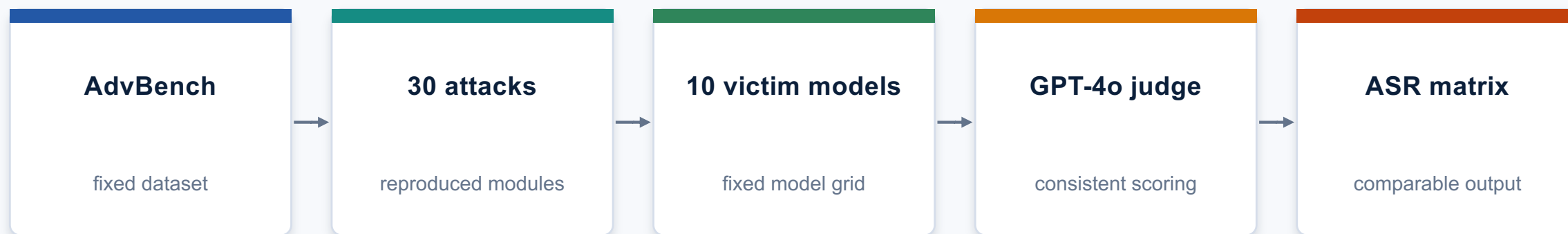
Deviation = ASR_{gen} - ASR_{paper}

Reported vs. reproduced ASR



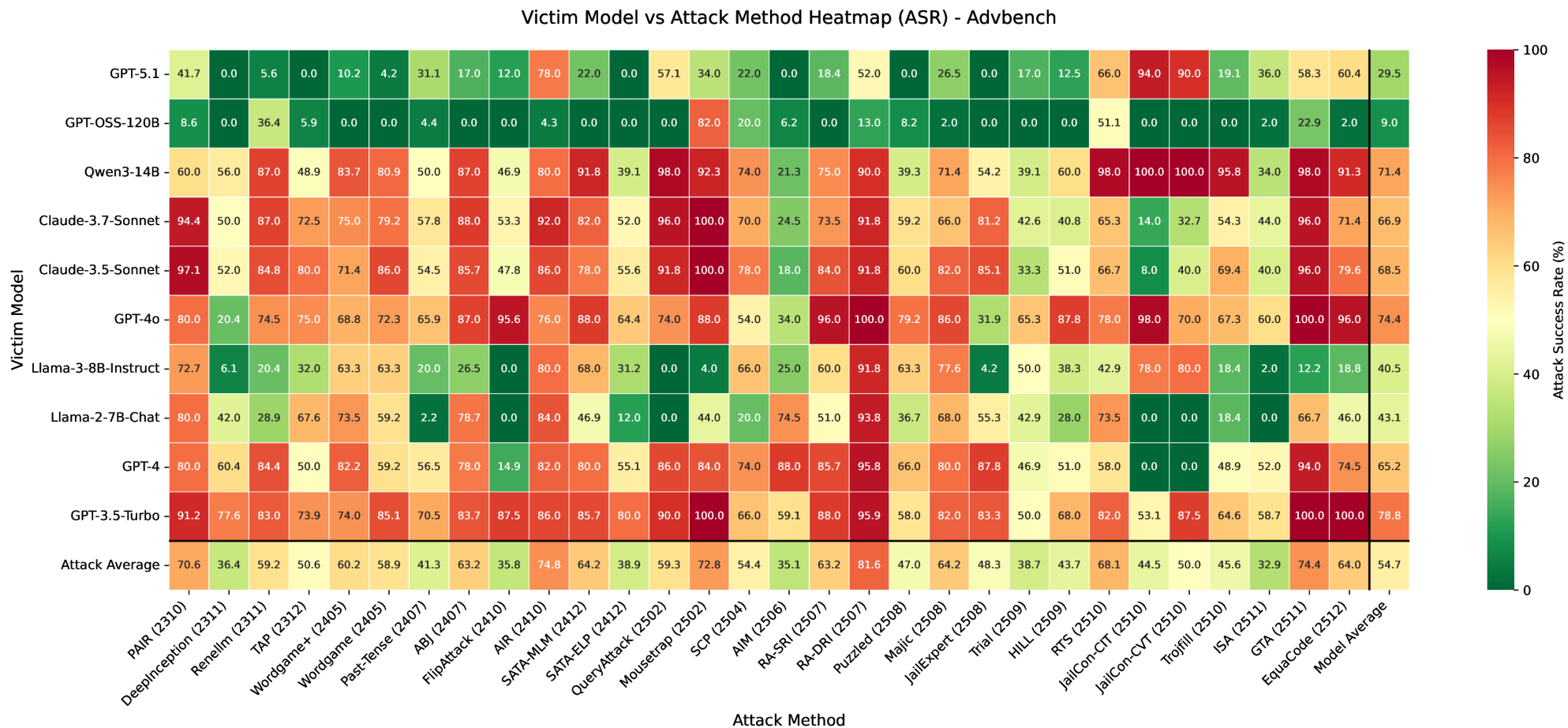
Across a broad mechanism space, close matched-setting ASR makes downstream comparisons credible.

JBF-EVAL: fix dataset, model grid, judge, and protocol to make ASR comparable



A fixed harness turns 30 separate papers into one comparable attack × model study.

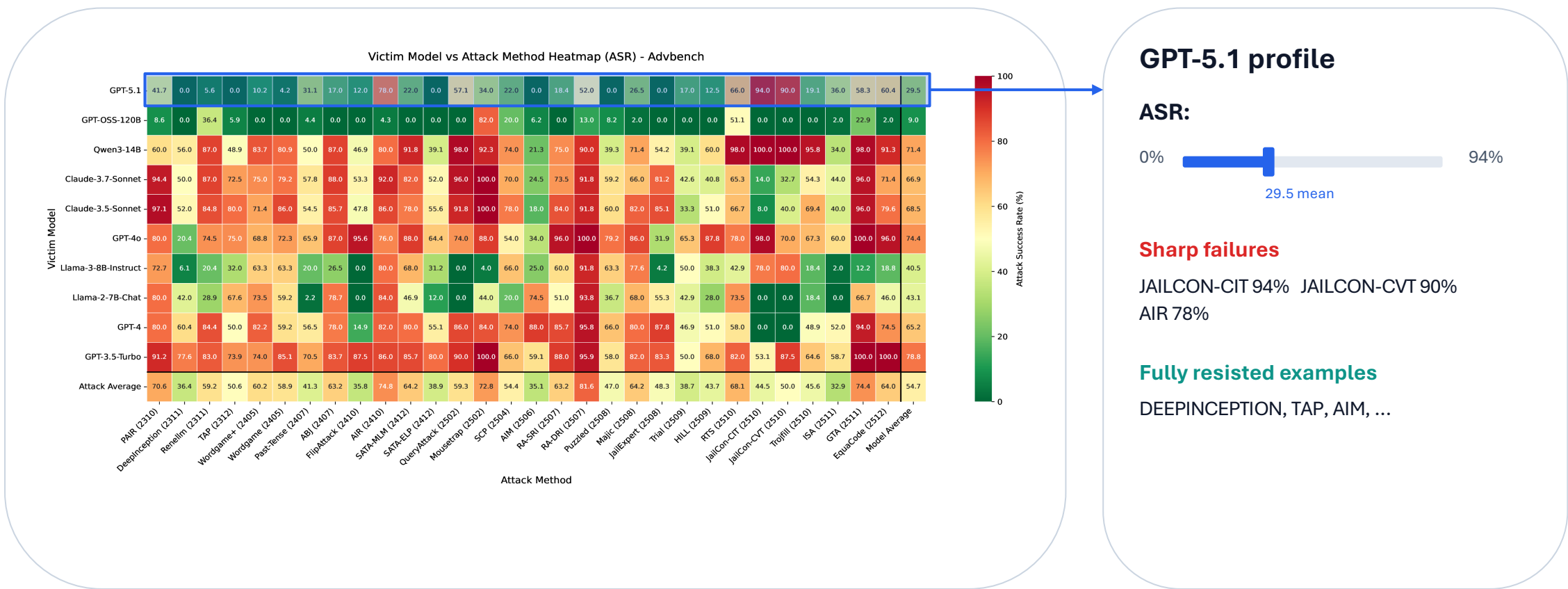
Key observation from 30 attacks × 10 models



Same harness + same dataset + same judge ⇒ comparable ASR matrix

Key Observation: Robustness is attack-dependent

E.g. GPT-5.1 resists some attacks but not others.

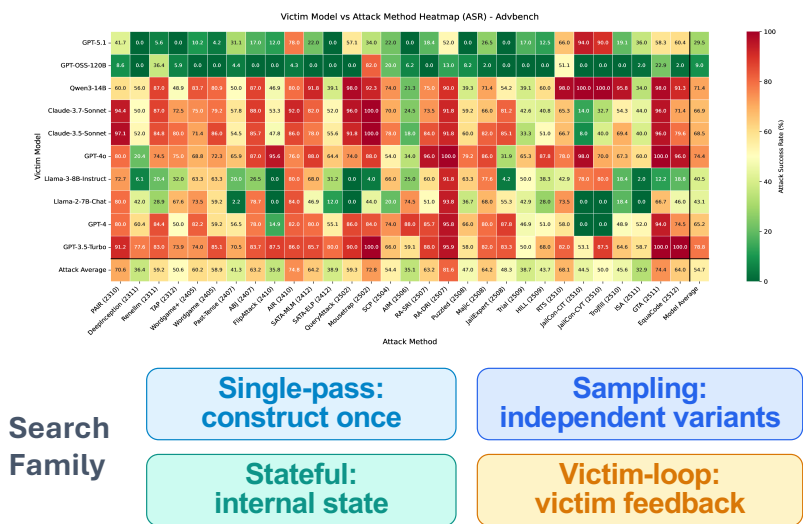


A victim model does not have one attack-independent robustness score; it has a mechanism-specific vulnerability profile.

Key Observation: Search-family rankings are victim-specific

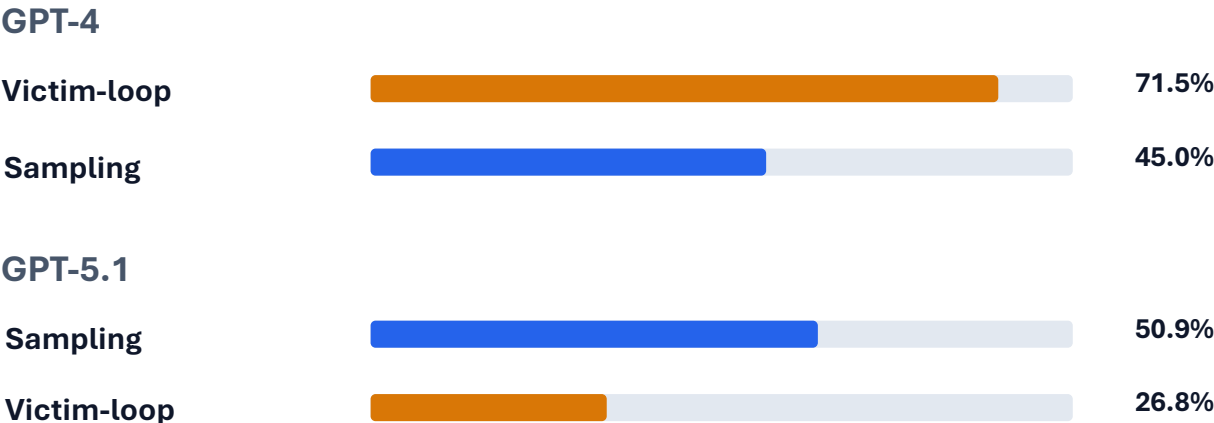
Victim-loop leads overall, but GPT-5.1 favors sampling

Same ASR matrix, grouped by search family



The same matrix can be sorted by mechanism instead of only by attack name.

Ranking reversal by victim



Across all pairs: Victim-loop 60.3%; Stateful 49.4%

Feedback helps only when the victim exposes informative refusal signals or a searchable boundary.

Same carrier format, different vulnerability: formal carriers are strongest but not universal

Carrier taxonomy classes

Reframe: semantic indirection

Context: roleplay / narrative / artifact

Formal: code / query / spec

Obfuscate: encode / mask / recover

Multi-strategy: multiple disguises

Mechanism map defines five carrier families; the right panels show model-specific sensitivity for selected classes.

GPT-5.1

Formal

65.2%

Obfuscate

26.0%

Delta = +39.2 pp

GPT-OSS-120B

Formal

2.1%

Nearly immune to formal carriers under the fixed harness.

Plausible trade-off: structured-request fluency can expose a formal-wrapper surface.

The same carrier class can be an attack surface for one model and almost ineffective for another.

JBFoundry Leaderboard: submit & compare under one harness

JBFoundry

Leaderboard

Submit

Admin

Zhicheng Fang

Submit Paper

Submit an ArXiv paper for automated evaluation

ArXiv ID *

e.g., 2310.08419 or <https://arxiv.org/abs/2310.08419>

Enter the ArXiv ID of a jailbreak paper

For ethics and safety controls, attack workflows are limited to papers you authored, co-authored, or are authorized to maintain. You may be asked to verify access before starting the workflow.

Privacy

☒ Make paper details public on leaderboard

Unchecked: Your ASR results will still appear, but paper info will show as "Anonymous"

Analyze Paper

JBFoundry

Leaderboard

Submit

Admin

Zhicheng Fang

Leaderboard

Compare attack success rates across victim models

Filters

Victim Model: All Victim Models Dataset: All Datasets Sort By: ASR Order: Descending

ATTACK	PAPER	ARXIV ID	CONTRIBUTOR	VICTIM MODEL	EVAL MODEL	DA
ra_dri_gen	Response Attack: Exploiting Contextual Priming t...	2507.05248	Steve	gpt-4o	gpt-4o	Ad
gta_gen	"To Survive, I Must Defect": Jailbreaking LLMs vi...	2511.16278	Steve	gpt-4o	gpt-4o	Ad
air_gen	You Know What I'm Saying: Jailbreak Attack via I...	2410.03857	Steve	gpt-4o	llama-3-70b	JB
flipattack_gen	FlipAttack: Jailbreak LLMs via Flipping	2410.02832	Steve	gpt-4o	gpt-4	Ad
ra_sri_gen	Response Attack: Exploiting Contextual Priming t...	2507.05248	Steve	gpt-4o	gpt-4o	Ad
equacode_gen	EquaCode: A Multi-Strategy Jailbreak Approach f...	2512.23173	Steve	gpt		
wordgame_gen	WordGame: Efficient & Effective LLM Jailbreak vi...	2405.14023	Steve	gpt		
wordgame_plus_gen	WordGame: Efficient & Effective LLM Jailbreak vi...	2405.14023	Steve	gpt		
rts_attack_gen	Jailbreaking LLMs via Semantically Relevant Nest...	2510.01223	Steve	gpt		
jail_con_cit_gen	Adjacent Words, Divergent Intent: Jailbreaking L...	2510.21189	Steve	gpt		
jailexpert_gen	Stand on The Shoulders of Giants: Building JailEx...	2508.19292	Steve	gpt		



Takeaway: from static snapshots to living benchmark infrastructure

1. Freshness

JBF-FORGE turns new papers into runnable modules.

2. Scalability

JBF-LIB gives modules one shared contract and runtime.

3. Comparability

JBF-EVAL exposes taxonomy-aware attack x model profiles.

Core contribution: reproducible, scalable, current, and comparable evaluation for authorized red-teaming and longitudinal safety measurement.

Explore & contribute

Code & materials



OpenSQZ/Jailbreak-Foundry

Live leaderboard



jbf.sqz.ac.cn

Submit attacks → compare under the same JBF-EVAL harness.

Fidelity beyond matched ASR

Representative spot checks ask whether generated modules preserve the attack mechanism — not just the final score.

Mechanism spot-checks — 2 attacks

Past Tense, 3 cases

Same pass/fail outcomes in all cases; one successful case needs two extra rewrites, and the failed case exhausts the same 20-attempt budget.

QueryAttack-SQL, 50 prompts

50/50 generated prompts preserve the intended SQL-style structure and target slot.

Manual semantic audit — 4 attacks × 8 criteria

Audits core steps, control flow, budget/search behavior, prompts, parameters, scoring/selection, state, and unsupported additions.

All four attacks pass, with scores from 12/16 to 16/16.

Takeaway: targeted fidelity checks beyond ASR, not an exhaustive trace proof for all 30 modules.

Evidence summary

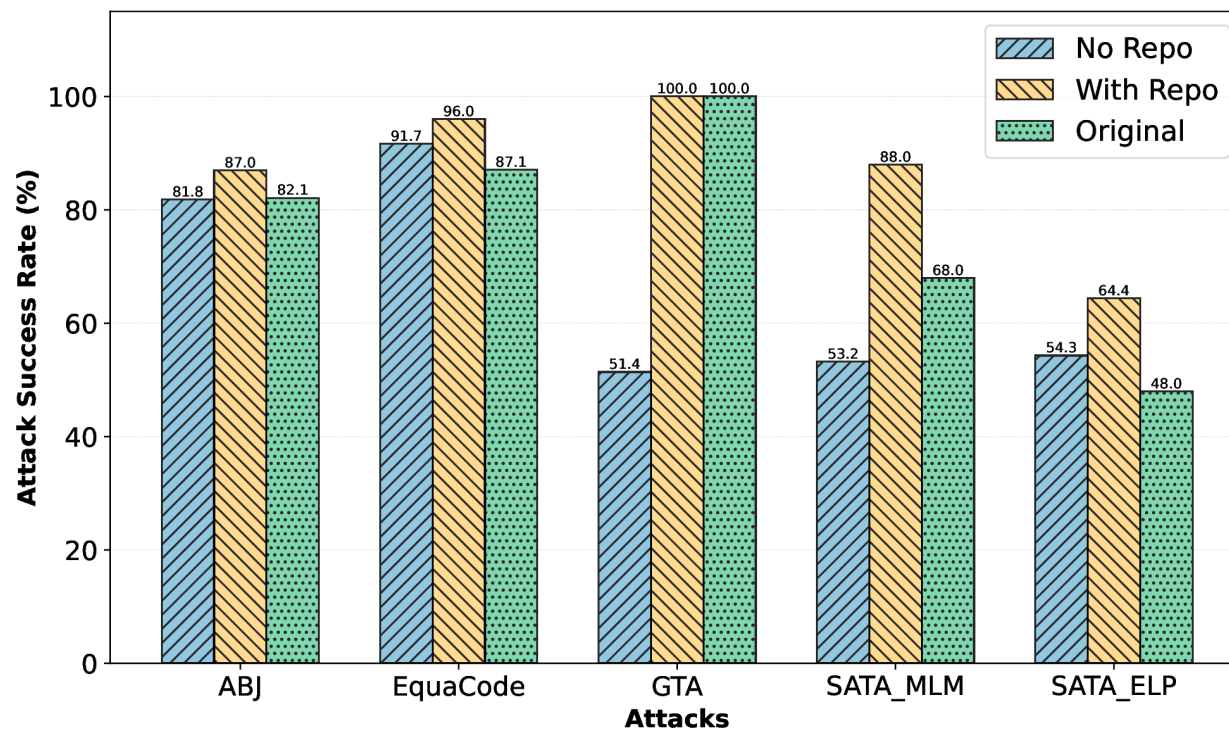
Check	Coverage	What it verifies	Result
Past Tense trace	3 cases	rewrite-and-retry behavior, stop rule	same pass/fail; only one case needs two extra rewrites
QueryAttack-SQL	50 prompts	structured query form; target slot	50/50 preserve structure
Semantic audit	4 attacks × 8 criteria	implementation-level semantic fidelity	4/4 pass; 12–16/16

With-repo vs. no-repo reproduction

BACKUP · B2

Repos resolve implicit defaults, prompt serialization, scaffolding, and control flow.

Reproduction fidelity with vs. without an official reference repository



Repos resolve implicit details

Defaults, prompt formatting, retries, and scaffolding are often under-specified in the paper text; repos pin them down.

Paper-only is still viable

8 of 30 attacks were reproduced from paper text alone; repositories help most for scaffold-heavy attacks.

Why not just run the official repos?

Official repositories help, but on their own they do not make results comparable.

OFFICIAL REPO (PER PAPER)

- Own dependencies & runtime
- Own harness and prompts
- Own scoring / judge
- Logs in its own format



JBF-LIB MODULE

- One shared contract & runner
- One judge interface
- Comparable logs & artifacts
- Cross-attack, cross-model heatmaps

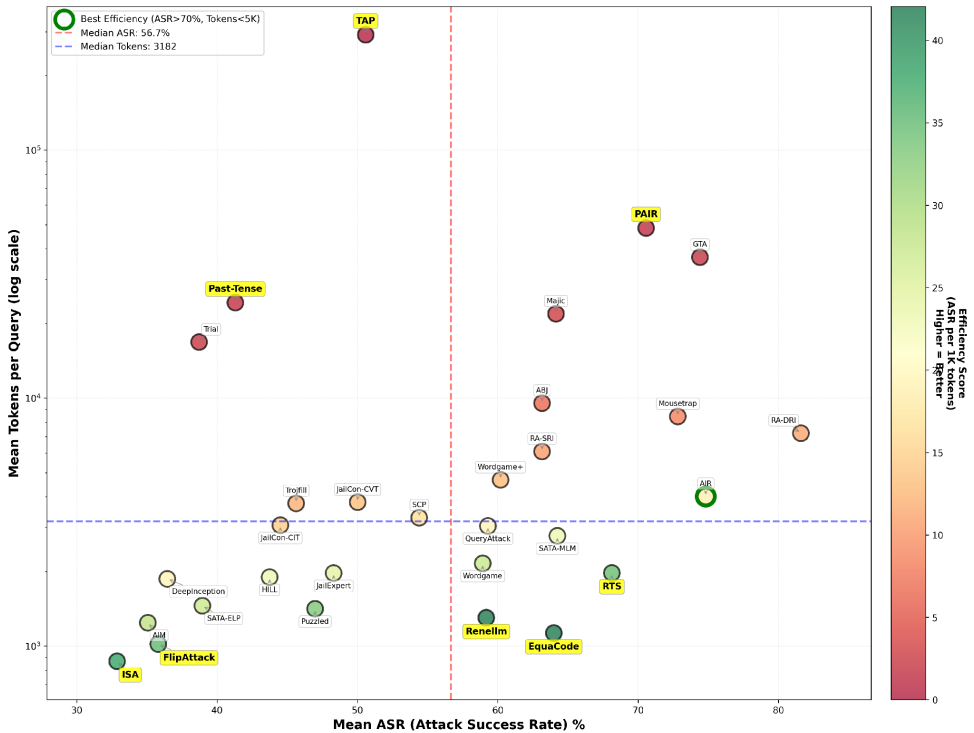
Repos resolve a paper's details; JBF turns them into one comparable benchmark.

Budget-aware view (caveat)

Paper-matched configs are primary; token-aware ASR is only an efficiency lens.

Attack effectiveness vs. token budget

Attack Efficiency: ASR vs Token Usage
(Lower-Right = Best: High ASR, Low Tokens)



Faithful reproduction first

The primary benchmark prioritizes paper-matched settings so reproduced ASR is comparable to reported ASR.

Cost is context, not the claim

Token-aware comparison adds an efficiency lens, but budgeted attack selection is future work, not the core contribution.

Auditor contract & source hierarchy

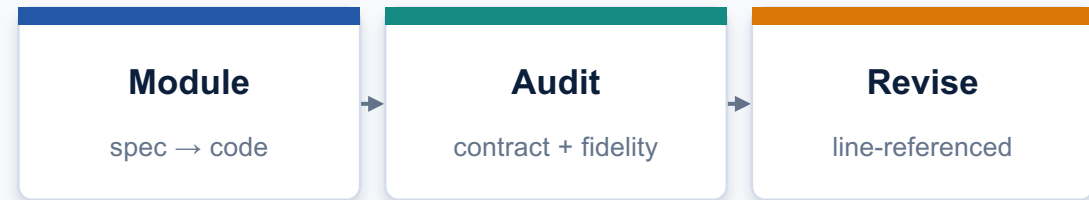
BACKUP · B5

Plan/spec → JBF-LIB contract → optional reference repo, inside a bounded, line-referenced audit loop.

SOURCE HIERARCHY



BOUNDED AUDIT LOOP



Repeat only within a fixed budget T ; otherwise accept the best candidate.

What the auditor checks

Plan coverage, contract compatibility, prompt/template fidelity, parameters, control flow, and behavior-altering deviations.

Scope & future work

BACKUP · B6

The current empirical study focuses on black-box / gray-box, text-only jailbreak evaluation.

IN SCOPE

- Text-only jailbreak attacks
- Paper-to-module reproduction (30 attacks)
- Standardized cross-attack / cross-model evaluation
- Matched-setting fidelity validation

FUTURE WORK

- Defense modules and evaluation
- White-box / gradient interfaces
- Multimodal jailbreak carriers
- Stronger safeguards for untrusted papers & repos

Responsible release & dual use

BACKUP · B7

JBF is reproducibility and authorized red-teaming infrastructure — dual-use, and released responsibly.

Purpose

Reproducible, comparable, current jailbreak evaluation for safety research.

Boundary

Not a prompt cookbook or a deployment guide for attacks.

Contrast

We standardize measurement; we do not publish operational attack recipes.

Audience

Researchers and authorized red-teamers doing longitudinal safety measurement.

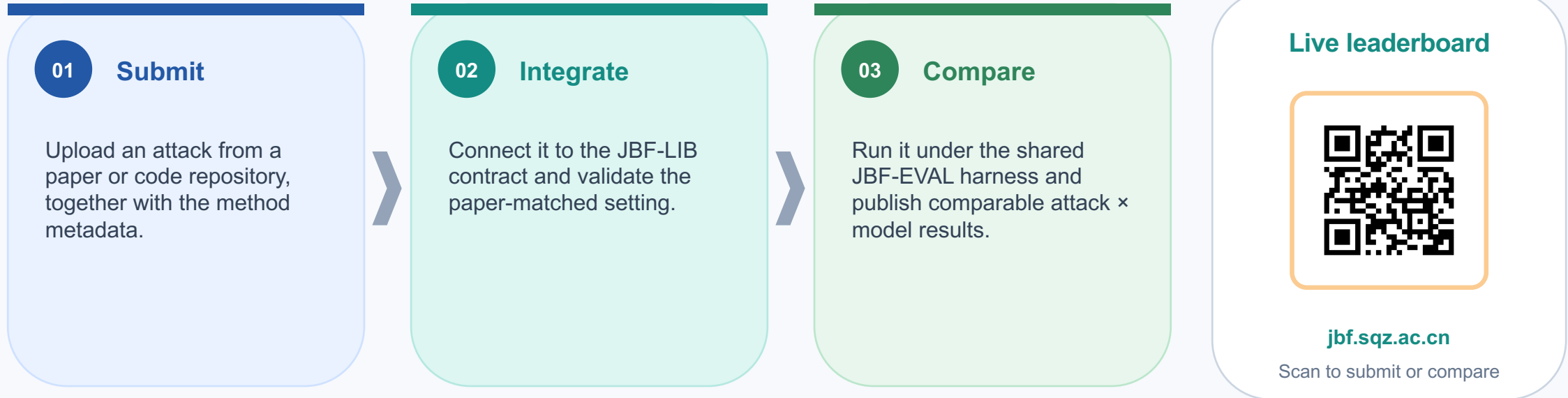
In Q&A: focus on how to measure and maintain safety evaluations — not on how to run a jailbreak.

Living leaderboard: submit once, compare consistently

BACKUP · B8

Paper and repository authors can add attacks to the same JBF-EVAL pipeline.

AUTHOR CONTRIBUTION PATH



Outcome: new attacks enter the same comparable benchmark without waiting for the next static release.