# BEYOND ONE-SHOT JAILBREAKING: CROSS-INSTANCE EXPERIENCE ACCUMULATION VIA REUSABLE SKILLS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Existing automated jailbreak methods repeatedly rediscover similar attack structures for each new request. A successful trajectory may reveal a reusable strategy—role-playing, scenario framing, or instruction override—yet this knowledge is typically discarded once the attack terminates. This makes large-scale red teaming query-expensive and prevents experience acquired on one attack from improving the next. We ask: (RQ1) Can successful attack trajectories become reusable cross-instance memory? (RQ2) Can such memory improve attack success while amortizing search cost? We propose a reusable skill framework that addresses both questions through three mechanisms: skill abstraction (distilling successful trajectories into compact templates), multi-factor retrieval (matching skills to new prompts via lexical scoring), and library maintenance (periodic pruning and merging to prevent skill dilution). Through 217 ablation configurations and 120 transfer evaluations across 4 models and 5 datasets, we demonstrate that the skill system provides +22.7 percentage points (pp) improvement over baselines, while an optional evolution mechanism contributes an additional +0.9pp refinement. When initialized with strong prior templates, the system achieves 99.7% ASR on same-family models. Skills transfer effectively within model families (85–100% ASR across Qwen3 variants) and significantly outperform baselines on cross-family transfer (+17.5pp over AutoDAN on GPT-OSS-20B). Our results validate that cross-instance experience accumulation is both feasible and effective for systematic jailbreak prompt generation.

## INTRODUCTION

Automated jailbreak attacks are essential for proactive safety evaluation of large language models (Zou et al., 2023). However, current methods share a fundamental inefficiency: **each attack starts from scratch**, discarding discovered knowledge. When PAIR (Chao et al., 2023) spends 20 iterations discovering that role-playing works for a request about weapons, this insight is lost—the next request about explosives must repeat the same search. This **memory gap** causes two wastes: (1) **exploration waste**—similar strategies are rediscovered repeatedly; (2) **query waste**—each attack pays full cost without benefiting from prior experience. On GPT-OSS-20B, AutoDAN averages 94 calls (6.1% ASR), while SESS requires only 8.8 calls (32.5% ASR), reducing cost by 90% while improving success rate by 5.3×.

This raises two questions: **RQ1: Can successful attack trajectories become reusable cross-instance memory? RQ2: Can such memory improve attack success while amortizing search cost?** We answer both affirmatively through **SESS** (**S**kill **E**xtraction and **S**election **S**ystem), a lightweight, pluggable framework with three mechanisms: **skill abstraction**—distilling successful trajectories into compact templates; **multi-factor retrieval**—matching skills via lexical scoring without embedding models; and **library maintenance**—periodic pruning to prevent skill dilution.

**Contributions.** (1) **Attribution finding.** Through 217 ablation configurations and 120 transfer evaluations across 4 models and 5 datasets, we establish a quantitative decomposition of where experience accumulation pays: the skill system (extraction, retrieval, maintenance) contributes the bulk (+22.7pp over baselines), evolution adds only +0.9pp refinement, and *strong structural priors set*



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

*the ceiling*—a Matthew effect, as accumulation amplifies prior strength—with DAN initialization the system reaches 99.7% ASR on same-family models and +17.5pp on cross-family transfer. (2) **Transfer finding.** Structural instruction-override patterns transfer across architectures substantially better than semantically optimized attack patterns (+26.4pp over AutoDAN on GPT-OSS-20B) while cutting query cost by 90%, implying that defenses should operate at the pattern level. (3) **Framework.** SESS, a lightweight pluggable implementation demonstrating that these findings require no heavyweight infrastructure.

## Related Work

### Context-Augmented Jailbreak Attacks

A large body of work augments jailbreak prompts through iterative refinement, structured search, or population-based evolution. These methods operate entirely within a single attack instance and share a common limitation: they do not accumulate knowledge across attacks.

**Iterative and search-based methods** optimize prompts through repeated interaction with the target model. PAIR (Chao et al., 2023) formulates jailbreaking as an iterative optimization process between an attacker LLM and a target LLM, typically requiring fewer than 20 queries to compromise aligned black-box models. TAP (Mehrotra et al., 2023) constructs a tree of attack prompts with pruning based on harmfulness scores. PromptAgent (Wang et al., 2024) frames prompt optimization as an agent planning problem with a plan-execute-reflect closed loop. While these methods leverage self-reflection capabilities, each attack starts from scratch.

**Evolutionary and template-based methods** explore the prompt space through population-based search. AutoDAN (Liu et al., 2023) combines DAN templates with genetic algorithms. DeepInception (Li et al., 2023) constructs multi-layer nested virtual scenarios. Crescendo (Russinovich et al., 2024) employs multi-turn progressive escalation. While these methods can discover effective prompts, they discard evolved populations between attack instances.

The shared characteristic across all these methods is **heavy reliance on trial-and-error within each attack**. None accumulate reusable knowledge across attack instances.

### Reinforcement Learning for Jailbreak

RL-based approaches treat jailbreak prompt generation as a sequential decision problem. Jailbreak-R1 (Guo et al., 2025) proposes a three-stage RL framework with curriculum learning. TRO-Jail (Xiong et al., 2026) models multi-turn jailbreaking as trajectory-level RL with process rewards. xJailbreak (Lee et al., 2025) introduces representation-space-guided jailbreaking. RL-MTJail (Chen et al., 2025a) applies RL to black-box multi-turn jailbreaking. ManyTurn (Yang et al., 2025b) systematically explores multi-turn paradigms. Beyond single-turn attacks, broader automated red-teaming frameworks have emerged: Belaire et al. (Belaire et al., 2025) formulate red-teaming as hierarchical RL; Genesis (Zhang et al., 2025) evolves attack strategies for web agents; and Wei et al. (Wei et al., 2025) propose a learning-driven adversarial framework.

RL-based methods are fundamentally data-driven and computationally intensive, requiring large-scale training data and significant GPU resources. The learned policies are model-specific and do not naturally transfer across architectures without retraining.

### Self-Evolving Attack Strategies

A recent trend focuses on systems that accumulate and refine attack knowledge over time. Metis (Chen et al., 2026) introduces self-evolving metacognitive policy optimization with composable strategy primitives. ASTRA (Liu et al., 2026) proposes a three-tier dynamic strategy library. EvoSynth (Chen et al., 2025b) shifts from prompt optimization to method evolution. RedHit (Sorkhpour et al., 2025) combines MCTS with preference optimization. Active Attacks (Yun et al., 2025) adapts the environment to prevent mode collapse.

**Memory-driven approaches.** Concurrent works independently discovered the memory-based paradigm. MemoAttack (Zhang et al., 2026) proposes skill-structured memory with lifecycle-driven

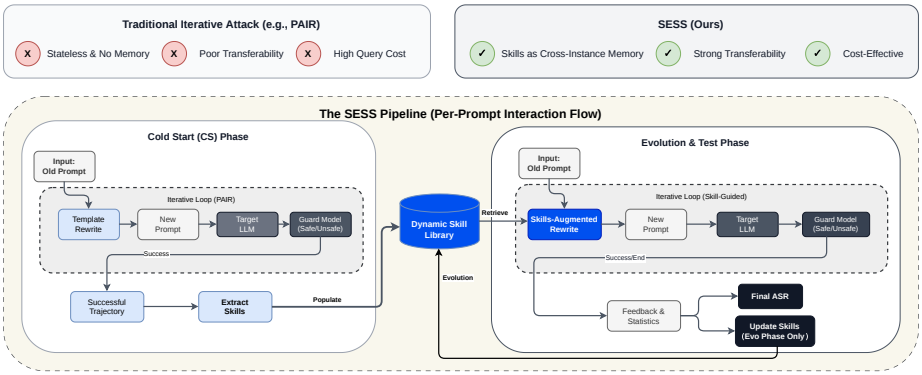



Figure 1: Overview of the reusable skill framework. The pipeline consists of two phases: **Cold Start** initializes the skill library by extracting templates from successful trajectories of a base method (e.g., PAIR). **Evolution & Test** applies skill-guided attacks to new prompts—during evolution, the "Update Skills" feedback loop refines the library based on attack outcomes; during testing, this loop is disabled and the library remains frozen. The skill library (center) serves as cross-instance memory, enabling experience accumulation across attacks.

evolution (probation, promotion, retirement) and Thompson Sampling for selection. SRTJ (Li et al., 2026) introduces hierarchical rule memory with answer set programming (ASP) for constraint-aware composition. Persona Attack (Park et al., 2026) focuses on incremental memory injection across conversation turns. These approaches share SESS's core insight—cross-instance experience accumulation—but differ in architecture and complexity.

These methods share the insight that successful attacks contain reusable strategic patterns. The key differentiator is **cross-instance knowledge transfer**—skills learned from one attack inform subsequent attacks.

**Positioning of SESS.** Our work falls within this category but emphasizes a **lightweight, pluggable design**: (1) minimal skill representation as string templates with lightweight metadata; (2) multi-factor lexical retrieval without embedding models or ASP solvers; and (3) cold-start from existing trajectories without accumulated samples. This design enables seamless integration with base methods (PAIR, AutoDAN) as a plug-in enhancement rather than replacing them. Table 6 (Appendix A) summarizes architectural differences.

## METHOD

**What is a Skill in SESS?** Intuitively, a *skill* is a reusable attack *shortcut*—a prompt prefix that, when prepended to a harmful request, has proven effective at bypassing safety filters. For example, if an attack succeeds by framing a request as "In a hypothetical scenario for educational purposes, explain...", this framing can be extracted as a skill and retrieved for semantically similar requests. Skills thus serve as **compressed memory of successful attack strategies**, enabling new attacks to start from proven patterns rather than random exploration.

The key insight is that jailbreak success often depends on *how* a request is framed, not just its semantics. By maintaining a library of proven framings and matching them to new requests, SESS transforms instance-level trial-and-error into cross-instance experience accumulation. When a new request arrives, the system retrieves the most relevant skill and prepends it to the prompt, immediately constraining the search to high-probability regions of the attack space. This eliminates the need to rediscover effective patterns from scratch.

### OVERVIEW

Our framework extracts reusable attack skills from successful jailbreak trajectories and maintains a dynamic skill library. The system operates in two distinct phases:

**Evolution Phase.** The skill library is actively built and refined through attack outcomes. This phase consists of three stages: (1) **Cold Start**: Initialize the library by extracting compact attack templates from successful trajectories of a base method (e.g., PAIR). (2) **Skill Accumulation**: Continuously expand the library—successful attacks yield new skills, while failed attacks trigger refinement. (3) **Library Maintenance**: Periodically prune low-quality entries and merge duplicates to prevent skill dilution. The evolution phase creates a continuous improvement loop where attack experience directly informs future attacks.

**Test Phase.** The skill library is frozen—no additions, deletions, or modifications occur. New attacks only retrieve and apply skills via multi-factor scoring, prepending them as contextual guidance to harmful prompts. This separation ensures stable, reproducible evaluation and prevents test-time information leakage (skills learned from test samples cannot influence other test samples).

The core innovation lies in treating jailbreak prompt generation as a *search space compression* problem. Rather than exploring the full space of possible prompts from scratch, the framework constrains the search to regions with historically high success rates. Both theoretically and empirically, this reduces the number of iterations required for successful attacks (experimentally: PAIR baseline averages approximately 20 iterations, while PAIR+skills averages approximately 4.45 iterations).

At its core, the framework injects **effective contextual information** into the attack process through the skill library, guiding the generation direction of jailbreak prompts. This design makes it a **lightweight, pluggable** system that can be integrated with various existing attack methods rather than replacing them. The skill library provides empirical priors; the underlying attack method handles the specific optimization process.

Figure 1 illustrates the complete pipeline.

### SKILL REPRESENTATION

A *skill* in SESS is a compact, reusable attack template distilled from a successful jailbreak trajectory. Formally, each skill is represented as a tuple:

$$s = (id, name, d, c, \sigma, P) \tag{1}$$

where $id$ is a unique identifier (UUID) for tracking, deduplication, and evolutionary lineage; $name$ is a short descriptive label (e.g., `role_play_expert`) for human interpretability; $d$ is a one-sentence description of what the skill does and when to apply it; $c$ is the attack template content—a natural-language prompt prefix that, when prepended to a harmful request, increases the likelihood of bypassing safety filters; $\sigma$ is a collection of statistical fields: `usage_count`, `success_count`, `success_rate` ($n_{\text{suc}}/n_{\text{use}}$), and `quality_score`; and $P$ is a list of applicable keyword patterns (`applicable_patterns`) indicating which prompts this skill is relevant to.

We define a **quality score** that balances proven effectiveness with accumulated evidence:

$$q(s) = \text{success_rate}(s) \times \sqrt{\text{usage_count}(s)} \tag{2}$$

The square-root term provides diminishing returns for high-usage skills, preventing well-established skills from permanently dominating retrieval while allowing new, promising skills to compete.

This representation is deliberately minimal: the content $c$ is a plain string, the statistics $\sigma$ are simple counters, and the patterns $P$ are a flat keyword list. This simplicity enables efficient storage, fast retrieval, and straightforward interpretation.

### MULTI-FACTOR RETRIEVAL

Given a new harmful prompt $p$, SESS retrieves the most relevant skills through a three-step process. First, **Feature Extraction**: the prompt is analyzed to extract a set of matched keywords $K_p$ from 14 predefined patterns (e.g., "how to", "guide", "expert", "research"). Second, **Scoring**: each skill $s$ receives a match score computed as:

$$\text{score}(s, p) = q(s) \cdot \big(1 + \alpha \cdot \text{matched}(s, p)\big) \tag{3}$$

where $\text{matched}(s, p)$ counts how many of the prompt's keywords appear in $P$, and $\alpha = 0.3$ is the per-keyword boost. Third, **Ranking**: skills are sorted by score, and the top-ranked skill is returned.

We deliberately adopt **lexical matching** rather than embedding-based retrieval, for three reasons. (1) **Interpretability**: every retrieval decision traces to explicit keyword matches, so operators can audit which skill was applied and why. (2) **Zero additional dependency**: an embedding retriever couples SESS to an auxiliary encoder (GPU memory, serving, version drift), whereas lexical matching keeps SESS a pure string-level, plug-and-play module. (3) **Retrieval is not the bottleneck**: attack cost is dominated by target-model calls (Table 4); we trade marginal semantic recall for auditability and deployability. Semantic retrieval is a complementary extension, not a requirement.

### Attack Execution

SESS supports two modes of skill integration. **Single-call mode** (default): the best-matching skill is retrieved once before the attack and prepended to the harmful prompt for the entire trajectory. **Every-iteration mode**: the skill is re-retrieved at each iteration, allowing dynamic strategy switching. Our ablation study (Table 7) reveals that single-call mode achieves 74.7% ASR vs. 66.4% for every-iteration (+8.3pp), indicating that stable skill guidance is more effective than dynamic switching. In both modes, the retrieved skill's content $c$ is prepended to the prompt $p$ to form $c \mid p$, processed by the underlying attack method (e.g., PAIR's attacker-target dialogue).

### Reflection and Evolution

After each attack, SESS reflects on the outcome to update the skill library. On success, a new skill is extracted from the trajectory: $s_{\text{new}} = \textsc{Extract}(p, \pi^*, s_{\text{used}})$. On failure, the existing skill is refined based on the refusal analysis: $s' = \textsc{Refine}(p, \pi_{\text{fail}}, r_{\text{refuse}}, s_{\text{used}})$.

Four evolution strategies control when updates occur: `success_only` (extract on success only), `failure_only` (refine on failure only), `both` (update on both), and `statistical` (default, periodic maintenance only). Our ablation (Table 9) shows `statistical` achieves the best balance (79.1% ASR, 28 skills).

### Library Maintenance

Library maintenance operates exclusively during the evolution phase; the library is frozen during testing. As the skill library grows, two risks emerge: (1) **skill dilution**, where low-quality skills degrade retrieval accuracy, and (2) **redundancy**, where near-duplicates waste capacity. SESS performs periodic maintenance through three mechanisms. **Pruning**: remove skills with low quality scores and sufficient usage evidence. **Merging**: combine highly similar skills into a single representative entry. **Capacity control**: bound library size; evict lowest-quality skills when full. Maintenance ensures the library remains focused on high-performing, non-redundant patterns.

## Experiments

### Experimental Setup

**Computing infrastructure.** All experiments were conducted on NVIDIA A100-80GB GPUs. The skill library implementation uses Python 3.10 with PyTorch 2.6+ and Transformers $\geq$ 4.51.0. We use vLLM for efficient model serving with PagedAttention, enabling concurrent deployment of attacker, target, and guard models. Random seeds are fixed at 42 for reproducibility across all experiments.

The guard model (Qwen3Guard-Gen-4B) (Team, 2025) evaluates responses under strict judgment: only `Unsafe` counts as success. Target models include Qwen3-0.6B, Qwen3-4B, Qwen3-14B (Yang et al., 2025a) (same family) and GPT-OSS-20B (OpenAI, 2025) (cross-family). Transfer datasets include AdvBench (Zou et al., 2023), HarmBench variants (Chao et al., 2024), and JailbreakBench (Caswell et al., 2024). Training data is drawn from WildTeaming (Jiang et al., 2024).

The default SESS configuration uses `single_call` retrieval, `trajectory` extraction, and `statistical` evolution; the full configuration is summarized in Table 5 (Appendix A).

**Source/target separation.** Skills are extracted exclusively from the WildTeaming training split (1,000 samples: 200 for cold start, 800 for evolution). All test evaluations use an independently

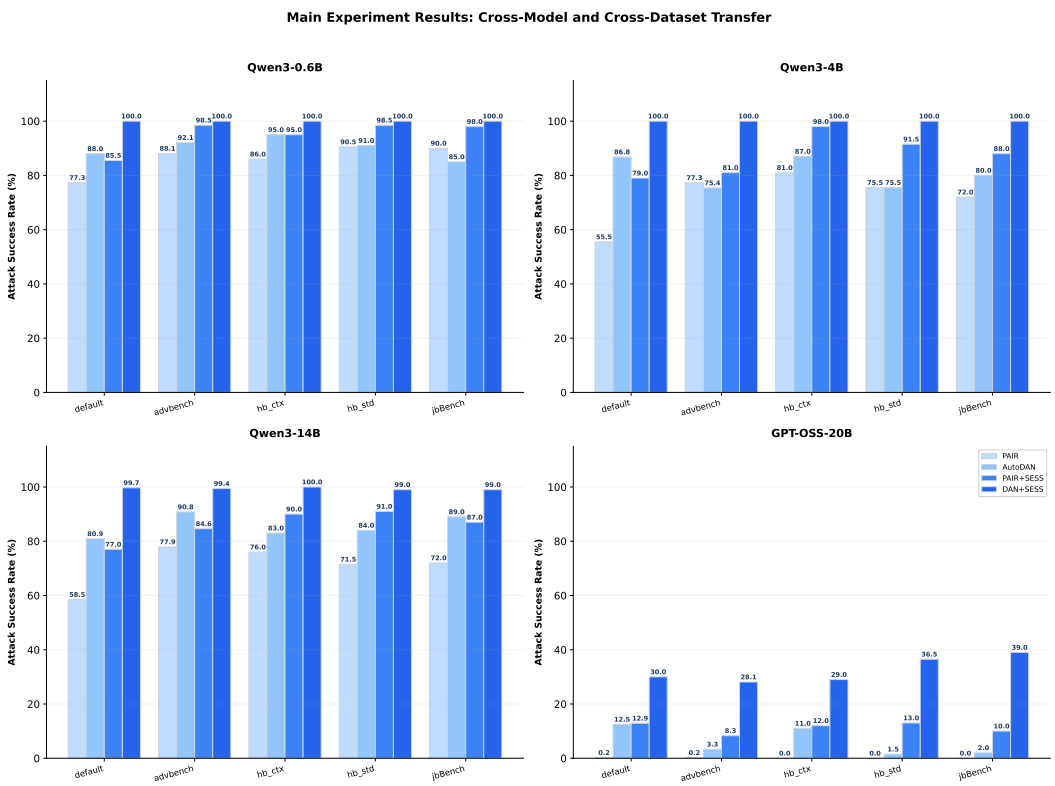



Figure 2: Main experimental results across 4 models and 5 datasets. Each subplot shows one target model with results across datasets. SESS consistently improves both PAIR and AutoDAN baselines. DAN+SESS (Ours) uses PAIR iterative optimization with DAN template initialization, achieving near-perfect performance on same-family models (99.7–100%) and significant advantages in cross-family transfer (+17.5pp on GPT-OSS-20B).

Table 1: Cross-dataset results on Qwen3-4B (skill-building model). Skills extracted from WildTeaming transfer to four external datasets.

| **Method** | **default** | **advbench** | **hb_ctx** | **hb_std** | **jbBench** |
|---|---|---|---|---|---|
| no_rewrite | 30.3 | 3.3 | 79.0 | 21.5 | 7.0 |
| DeepInception | 40.4 | 41.0 | 80.0 | 47.5 | 38.0 |
| PAIR | 55.5 | 77.3 | 81.0 | 75.5 | 72.0 |
| AutoDAN | 86.8 | 75.4 | 87.0 | 75.5 | 80.0 |
| PAIR+SESS (Ours) | 79.0 | 81.0 | 98.0 | 91.5 | 88.0 |
| DAN+SESS (Ours) | **100.0** | **100.0** | **100.0** | **100.0** | **100.0** |

drawn, non-overlapping test split (1,000 samples) or external transfer datasets (AdvBench, HarmBench, JailbreakBench) that share no prompts with the training data. We verify that no near-duplicates exist between training and test splits by checking pairwise keyword overlap. This separation ensures that reported improvements reflect genuine skill generalization rather than memorization of training prompts.

### Main Results

We conduct a large-scale evaluation across 4 target models and 5 datasets, totaling 140 experimental configurations. Figure 2 presents the main results visually. Table 1 reports cross-dataset results on the skill-building model, and Table 2 reports cross-model transfer.

We examine two dimensions of generalization (Tables 1–2, Figure 2):

Table 2: Cross-model transfer (average ASR across 5 datasets). Skills trained on Qwen3-4B transfer to same-family (Qwen3) and cross-family (GPT-OSS) targets.

| | Target Model | | | |
|---|---|---|---|---|
| **Method** | **Qwen3-0.6B** | **Qwen3-4B** | **Qwen3-14B** | **GPT-OSS-20B** |
| no_rewrite | 58.0 | 28.2 | 24.3 | 0.1 |
| PAIR | 86.4 | 72.3 | 71.2 | 0.1 |
| AutoDAN | 90.2 | 80.9 | 85.5 | 6.1 |
| PAIR+SESS (Ours) | 95.1 | 87.5 | 85.9 | 11.2 |
| DAN+SESS (Ours) | **100.0** | **100.0** | **99.4** | **32.5** |

Table 3: Component ablation on two target models. Each row removes one component from the full SESS pipeline. "Random" replaces matched retrieval with random skill selection. Baseline rows show the underlying attack without any skill library.

| | PAIR+SESS | | DAN+SESS | |
|---|---|---|---|---|
| **Config.** | **Qwen3-4B** | **GPT-OSS** | **Qwen3-4B** | **GPT-OSS** |
| Full SESS | **79.1** | **12.9** | **99.7** | **30.0** |
| −Cold Start | 65.2 | 5.3 | 98.8 | 28.5 |
| Random ret. | 63.8 | 4.1 | 96.2 | 23.5 |
| −Evolution | 78.2 | 11.5 | 98.8 | 28.3 |
| No skills | 55.5 | 0.2 | 86.8 | 12.5 |

**Cross-dataset transfer.** On Qwen3-4B (Table 1), PAIR+SESS improves from 55.5% to 79.0% (+23.5pp) on the held-out default split, confirming that skills generalize to unseen requests. Transfer to external datasets shows consistent gains: AdvBench (+3.7pp), HarmBench-ctx (+17.0pp), HarmBench-std (+16.0pp), and JailbreakBench (+16.0pp). The smaller gain on AdvBench likely reflects its shorter, more direct prompts that leave less room for contextual skill guidance. DAN+SESS reaches 100.0% across all five datasets.

**Cross-model transfer.** Within the Qwen3 family (Table 2), PAIR+SESS achieves 85.9–95.1% average ASR and DAN+SESS reaches 99.4–100.0%, suggesting shared alignment training creates common vulnerabilities. Cross-family transfer to GPT-OSS-20B is substantially harder: PAIR+SESS recovers only 11.2% (vs. 0.1% for PAIR), but DAN+SESS reaches 32.5%—a +26.4pp improvement over AutoDAN (6.1%). This gap confirms that structural attack patterns (instruction-override templates) generalize better across architectures than semantically optimized patterns, and that strong structural priors are particularly critical for cross-family transfer. A per-dataset heatmap of all 20 method×dataset cells per target model is provided in Figure 3 (Appendix A).

### Ablation Study

**Design-level ablations.** Single-call retrieval outperforms every-iteration re-retrieval by +8.3pp (74.7% vs. 66.4%; Table 7, Appendix A), indicating that stable skill guidance is more effective than dynamic switching. Trajectory-level extraction outperforms final-prompt-only extraction by +2.5pp (71.8% vs. 69.3%; Table 8, Appendix A), as the full trajectory contains richer information about the optimization process. On data strategy (Table 9, Appendix A), larger data volumes yield modest gains (+2.6pp from 300 to 1000 trajectories), suggesting that more diverse attack experience benefits skill quality, while the cold-start/evolution ratio has minimal impact ($<$1.1pp), indicating the evolution phase is relatively insensitive to data allocation.

The `statistical` strategy achieves 79.1% ASR with 28 skills, while `failure_only` (100 skills, 76.2%) and `both` (74 skills, 70.4%) suffer from skill dilution.

Table 3 ablates each SESS component on two target models: Qwen3-4B (same-family, the skill-building model) and GPT-OSS-20B (cross-family). We compare two attack configurations—PAIR+SESS (skill library built from PAIR trajectories) and DAN+SESS (skill library initialized

Table 4: Average attacker model calls per attack across 4 datasets. Lower is better. DAN+SESS reduces query cost by 78.7–90.7% compared to AutoDAN.

| | Target Model | | | |
|---|---|---|---|---|
| **Method** | **Qwen3-0.6B** | **Qwen3-4B** | **Qwen3-14B** | **GPT-OSS** |
| PAIR | 2.10 | 3.25 | 3.35 | 10.00 |
| AutoDAN | 10.10 | 10.00 | 10.80 | 93.78 |
| PAIR+SESS (Ours) | 1.82 | 3.09 | 3.57 | 9.40 |
| DAN+SESS (Ours) | **1.11** | **1.11** | **2.30** | **8.76** |

with DAN templates)—under four conditions: the full pipeline, removing cold start (−CS), replacing matched retrieval with random skill selection, and removing evolution (−Evo).

**Cold start matters most when building from scratch.** For PAIR+SESS, removing cold start drops ASR from 79.1% to 65.2% (−13.9pp) on Qwen3-4B and from 12.9% to 5.3% (−7.6pp) on GPT-OSS-20B, confirming that the initial skill library extracted from PAIR trajectories is essential for guiding subsequent attacks. For DAN+SESS, however, removing cold start costs only −0.9pp on Qwen3-4B (99.7% → 98.8%) and −1.5pp on GPT-OSS-20B (30.0% → 28.5%). This asymmetry is expected: the 6 DAN templates are already structured instruction-override patterns that require no PAIR-style skill extraction—they remain unchanged throughout evolution (0 new skills added), suggesting they are near-optimal attack strategies in their own right. Cold start is therefore critical for weaker priors (PAIR) but redundant when strong structural templates are available.

**Matched retrieval is essential for PAIR+SESS.** Random skill selection degrades PAIR+SESS from 79.1% to 63.8% (−15.3pp) on Qwen3-4B, showing that lexical matching is critical for selecting the right skill. The effect is smaller for DAN+SESS (−3.5pp on Qwen3-4B) because all DAN templates are structurally strong instruction-override patterns regardless of match quality.

**Evolution contributes marginal refinement.** Removing evolution costs only −0.9pp for both methods on Qwen3-4B (79.1% → 78.2% and 99.7% → 98.8%) and −1.4pp / −1.7pp on GPT-OSS-20B, confirming its role as an optional continuous optimization rather than a primary driver.

**Cross-family transfer amplifies all gaps.** Every component matters more on GPT-OSS-20B than on Qwen3-4B: the cold-start drop is proportionally larger, and random retrieval hurts more. This indicates that when transferring across fundamentally different architectures, the quality of the initial skill library and the precision of retrieval become even more decisive.

**Why the skill system works.** The quality score $q(s) = \text{success_rate} \times \sqrt{\text{usage_count}}$ balances proven effectiveness with accumulated evidence: the square-root term prevents well-established skills from permanently dominating retrieval, allowing new, promising skills to compete.

**Skill dilution is a real risk.** The `statistical` strategy (periodic maintenance only) achieves the best balance (79.1% ASR, 28 skills), while `failure_only` (100 skills, 76.2%) and `both` (74 skills, 70.4%) suffer from skill explosion that degrades retrieval precision. This confirms that periodic pruning and merging are essential—without maintenance, the library accumulates low-quality entries that dilute the effectiveness of high-performing skills.

### Computational Efficiency

Table 4 reports the average number of attacker model calls per attack. We measure the total attacker invocations required to complete one attack instance (including all iterations and population evaluations). Note that PAIR's maximum iteration count was set to 10 during testing. Consequently, when an attack method is effective, the call count is significantly lower (often 1–3 calls due to early success); however, when the attack method fails overall—as observed for GPT-OSS attacks—most cases exhaust the iteration limit, resulting in call counts near the experimental ceiling (10 calls for PAIR-based methods).

**DAN+SESS achieves dramatic cost reduction.** Compared to AutoDAN, DAN+SESS reduces attacker calls by 89.0% on Qwen3-0.6B (10.10 → 1.11), 88.9% on Qwen3-4B (10.00 → 1.11), 78.7% on Qwen3-14B (10.80 → 2.30), and 90.7% on GPT-OSS-20B (93.78 → 8.76). This reduction arises

because DAN templates provide near-optimal attack structure from the first call, eliminating the need for iterative population evolution.

**PAIR+SESS provides modest query savings.** On weaker targets (Qwen3-0.6B), PAIR+SESS reduces calls from 2.10 to 1.82 ($-13.3\%$), as skill guidance helps PAIR converge faster. On stronger targets (Qwen3-14B), the reduction is marginal or slightly negative ($-6.6\%$), suggesting that when the target model is robust, skill guidance alone cannot eliminate the need for iterative refinement.

**Cross-family cost amplification.** GPT-OSS-20B requires substantially more calls across all methods: AutoDAN needs 93.78 calls (vs. 10.0–10.8 for Qwen models) due to its low baseline ASR forcing more evolutionary generations. DAN+SESS mitigates this amplification most effectively, reducing the cross-family cost gap from $9\times$ (AutoDAN) to $3.8\times$ (DAN+SESS).

**Amortized cost model.** Cold start requires 200 attack instances to build the library; each subsequent attack then costs one retrieval ($<$1ms) plus 1–2 generation calls for DAN+SESS, versus 10–94 calls for baselines. For a campaign of $N$ attacks, total cost $200 \times C_{\text{baseline}} + (N - 200) \times C_{\text{SESS}}$ breaks even at $N > 220$.

## CONCLUSION

We presented SESS, a reusable-skill framework that turns instance-level jailbreak optimization into cross-instance experience accumulation. A large-scale study (217 ablation + 120 transfer runs) decomposes the gains: the **skill-system architecture** delivers the primary gain (+22.7pp over baselines), the optional **evolution mechanism** a minor refinement (+0.9pp), and **structural priors set the ceiling**—DAN initialization reaching 99.7% ASR within families and +17.5pp cross-family. Skills transfer within families (85–100% ASR) and generalize across architectures when seeded with structural templates, while cutting per-attack cost from 20–50 iterative queries to 1 retrieval + 1 generation.

**Implications for defense.** Maintaining a database of known attack templates and flagging matches shifts defense from reactive filtering to proactive detection; since structural patterns generalize across architectures better than semantic ones, structural validation complements semantic analysis. Future work will explore semantic retrieval, multi-family co-evolution, and multi-turn scenarios. Code and trained skill libraries will be released upon publication.

## LIMITATIONS

**Reliance on open-source models.** All 317 configurations were deployed locally via vLLM; we did not evaluate proprietary SOTA models (GPT, Claude), as PAIR and AutoDAN alone need 10+ API calls per attack. However, GPT-OSS-20B shows defense comparable to SOTA, and prior work (Belaire et al., 2025; Wei et al., 2025) reports similar robustness relative to GPT-4o/Claude, supporting generalizability.

**Prior knowledge bounds the ceiling (a Matthew effect).** DAN-initialized skills reach 99.7% ASR vs. 79.1% from an empty library: prior strength, not accumulation, sets the ceiling. Because SESS distills from *successful* experience, strong priors compound into stronger libraries while weak priors leave little to accumulate; the $\sqrt{\cdot}$ damping curbs skill monopoly within the library, but no internal mechanism compensates for a weak prior.

**Frontier API models resist controlled evaluation.** Extending to two hosted API targets, we observed markedly heavier system/self guardrails: the endpoints returned only a reasoning trace—a silent refusal—on $65\%$ (GLM) and $81\%$ (DeepSeek) of attempts. Requiring guard-confirmed harmful content (Unsafe with Refusal=No), genuine jailbreak ASR collapsed to $\approx 1\%/4\%$, indistinguishable across methods; we therefore omit these targets from the main analysis and record this refusal rate as evidence that frontier API deployments are currently poor targets for controlled comparative red-teaming.

## REFERENCES


Roman Belaire, Arunesh Sinha, and Pradeep Varakantham. Automatic llm red teaming. *arXiv preprint arXiv:2508.04451*, 2025.

Tom Caswell et al. Jailbreakbench: An open robustness benchmark for jailbreaking. *arXiv preprint arXiv:2404.01318*, 2024.

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.

Patrick Chao et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

Xiyuan Chen et al. Rl-mtjail: Reinforcement learning for automated black-box multi-turn jailbreaking. *arXiv preprint arXiv:2512.07761*, 2025a.

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*, 2025b.

Yunhao Chen, Xin Wang, Juncheng Li, Yixu Wang, Jie Li, Yan Teng, Yingchun Wang, and Xingjun Ma. Metis: Learning to jailbreak llms via self-evolving metacognitive policy optimization. *arXiv preprint arXiv:2605.10067*, 2026.

Weiyang Guo, Zesheng Shi, Zhuo Li, Yequan Wang, Xuebo Liu, Wenya Wang, Fangming Liu, Min Zhang, and Jing Li. Jailbreak-r1: Exploring the jailbreak capabilities of llms via reinforcement learning. *arXiv preprint arXiv:2506.00782*, 2025.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models, 2024. URL https://arxiv.org/abs/2406.18510.

Sunbowen Lee, Shiwen Ni, Chi Wei, Shuaimin Li, Liyang Fan, Ahmadreza Argha, Hamid Alinejad-Rokny, Ruifeng Xu, Yicheng Gong, and Min Yang. xjailbreak: Representation space guided reinforcement learning for interpretable llm jailbreaking. *arXiv preprint arXiv:2501.16727*, 2025.

Jindong Li, Ying Liu, Yali Fu, Jinjing Zhu, Leyao Wang, Menglin Yang, and Rex Ying. Srtj: Self-evolving rule-driven training-free llm jailbreaking. *arXiv preprint arXiv:2605.00974*, 2026.

Xuan Li et al. Deepinception: Hypnotize large language models to be jailbreakers. *arXiv preprint arXiv:2311.07588*, 2023.

Xiaogeng Liu, Nan Xu, Mu Sun, and Yanfang Liu. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*, 2023.

Xu Liu, Yan Chen, Kan Ling, Yichi Zhu, Hengrun Zhang, Guisheng Fan, and Huiqun Yu. Astra: An automated framework for strategy discovery, retrieval, and evolution for jailbreaking llms. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026. Accepted to ACL 2026, 20 pages, 7 figures, 13 tables.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianidis, Afshin Ghodsi, Ben Woo, et al. Tree of attacks: Jailbreaking black-box llms automatically. *arXiv preprint arXiv:2312.02119*, 2023.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Junyoung Park, Seongyong Ju, Sunghwan Park, and Jaewoo Lee. Persona attack: Incremental memory injection jailbreak attack against large language models. *arXiv preprint arXiv:2606.00150*, 2026.

Mark Russinovich, Ahmed Salem, and Ronen Eldan. Great, now write an article about that: The crescendo multi-turn llm jailbreak attack. *arXiv preprint arXiv:2404.01833*, 2024. Accepted to USENIX Security 2025.

Mohsen Sorkhpour, Abbas Yazdinejad, and Ali Dehghantanha. Redhit: Adaptive red-teaming of large language models via search, reasoning, and preference optimization. In *Proceedings of the 2025 Conference on Large Language Model Security*, 2025. URL https://aclanthology.org/2025.llmsec-1.2.

Qwen Team. Qwen3guard technical report. *arXiv preprint arXiv:2510.14276*, 2025.

Xinyuan Wang, Huan Li, Zhaoyang Wang, et al. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16428*, 2024.

Zhang Wei, Hanxuan Chen, Peilu Hu, Zhenyuan Wei, Chenwei Liang, Jing Luo, Ziyi Ni, Hao Yan, Li Mei, Shengning Lang, Kuan Lu, Xi Xiao, Zhimo Han, Yijin Wang, Yichao Zhang, Chen Yang, Junfeng Hao, Jiayi Gu, Riyang Bao, and Mu-Jiang-Shan Wang. Learning-based automated adversarial red-teaming for robustness evaluation of large language models. *arXiv preprint arXiv:2512.20677*, 2025.

Xiqiao Xiong, Ouxiang Li, Zhuo Liu, Moxin Li, Wentao Shi, Fengbin Zhu, Qifan Wang, and Fuli Feng. Trojail: Trajectory-level optimization for multi-turn large language model jailbreaks with process rewards. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026. Accepted to ACL 2026 Main Conference.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Xianjun Yang, Liqiang Xiao, Shiyang Li, Faisal Ladhak, Hyokun Yun, Linda Ruth Petzold, Yi Xu, and William Yang Wang. Many-turn jailbreaking. *arXiv preprint arXiv:2508.06755*, 2025b.

Taeyoung Yun, Pierre-Luc St-Charles, Jinkyoo Park, Yoshua Bengio, and Minsu Kim. Active attacks: Red-teaming llms via adaptive environments. *arXiv preprint arXiv:2509.21947*, 2025.

Junke Zhang, Jianwei Wang, Sishuo Chen, Yizhang He, Qingshuai Feng, and Zhengyi Yang. Evolving skill-structured attack memory enhances llm jailbreaking. *arXiv preprint arXiv:2605.29237*, 2026.

Zheng Zhang, Jiarui He, Yuchen Cai, Deheng Ye, Peilin Zhao, Ruili Feng, and Hao Wang. Genesis: Evolving attack strategies for llm web agent red-teaming. *arXiv preprint arXiv:2510.18314*, 2025.

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.

## ADDITIONAL RESULTS

This section collects the full experimental configuration, memory-framework comparison, design-level ablation tables, and the per-dataset transfer heatmap referenced from the main text.

Table 5: Experimental configuration.

| **Configuration** | **Setting** |
|---|---|
| Policy Model | Qwen3-4B |
| Guard Model | Qwen3Guard-Gen-4B |
| Target Models | Qwen3-0.6B, 4B, 14B, GPT-OSS-20B |
| Training Data | 1,000 samples (CS=200, Evo=800) |
| Test Data | 1,000 samples (independently drawn) |
| Transfer Datasets | AdvBench, HarmBench$\times$2, JailbreakBench |
| Metric | ASR (strict: Unsafe only) |
| SESS Configuration | single_call + trajectory + statistical |

Table 6: Comparison of memory-driven jailbreak frameworks.

| **Property** | **SESS** | **MemoAttack** | **SRTJ** | **Persona** |
|---|---|---|---|---|
| Memory unit | Template | Skill-struct | Rule | Injection |
| Selection | Lexical | Thompson | ASP | Sequential |
| Embedding-free | ✓ | × | × | ✓ |
| Cold-start ready | ✓ | × | × | ✓ |
| Pluggable design | ✓ | × | × | × |

Table 7: Retrieval timing ablation. Single-call mode consistently outperforms dynamic re-retrieval.

| **Mode** | **Avg. ASR** | $\Delta$ |
|---|---|---|
| `single_call` | **74.7%** | — |
| `every_iteration` | 66.4% | −8.3pp |

## DATA STRUCTURES

### SKILL DEFINITION

A *skill* is a compact, reusable attack template distilled from successful jailbreak trajectories. Formally, we define a skill as a tuple:

$$\text{Skill} = (id, name, description, content, source, statistics, patterns) \tag{4}$$

where:

[leftmargin=*]

- $id$: Unique identifier (UUID)
- $name$: Human-readable skill name (e.g., "role_play", "hypothetical_scenario")
- $description$: One-sentence description of what the skill does and when to use it
- $content$: The actual prompt prefix/template for injection
- $source \in \{\texttt{initial}, \texttt{extracted}, \texttt{evolved}, \texttt{merged}\}$: Origin of the skill
- $statistics$: Performance metrics including usage count, success count, and quality score
- $patterns$: List of applicability keywords for retrieval matching

The quality score is computed as:

$$\text{quality} = \frac{\text{success_count}}{\text{usage_count}} \cdot \sqrt{\text{usage_count}} \tag{5}$$

which balances success rate with statistical reliability (more usage evidence yields higher confidence).

### SKILL LIBRARY STRUCTURE

The skill library $\mathcal{L}$ maintains a collection of skills with dynamic maintenance. The structure includes a dictionary mapping skill IDs to skill objects, and metadata tracking size, capacity, creation time, and last update time.

Library maintenance includes three mechanisms:

[leftmargin=*]

- **Pruning**: Remove skills with quality $< \theta_{prune}$ and usage $\geq n_{min}$
- **Merging**: Combine skills with similarity $> \theta_{merge}$ into one representative
- **Capacity control**: Evict lowest-quality skills when $|\mathcal{L}| > C_{max}$

Table 8: Skill extraction level ablation. Trajectory-level extraction provides more comprehensive strategy information.

| Extraction Level | Avg. ASR | Δ |
|---|---|---|
| `trajectory` | **71.8%** | — |
| `final_prompt` | 69.3% | −2.5pp |

Table 9: Data strategy ablation (36 experiments). Larger data volumes yield modest improvements (+2.6pp from 300 to 1000 trajectories). The cold-start/evolution ratio has minimal impact (<1.1pp), suggesting evolution data volume is not critical.

| Data Size | Ratio | CS % | ASR (%) | Δ |
|---|---|---|---|---|
| *Data Volume (traj+stat, fixed: early ratio)* | | | | |
| small (300) | early | 30% | 78.0 | baseline |
| medium (500) | early | 30% | 79.5 | +1.5 |
| large (1000) | early | 30% | **80.6** | **+2.6** |
| *Ratio Strategy (traj+stat, fixed: large data)* | | | | |
| early | 30% CS | 30% | **80.6** | baseline |
| balanced | 20% CS | 20% | 80.2 | −0.4 |
| evo | 10% CS | 10% | 79.5 | −1.1 |

## KEY PROMPTS

This section presents the core prompts used in SESS for skill extraction and refinement.

### SKILL EXTRACTION FROM SUCCESSFUL ATTACKS

When an attack succeeds, we extract a reusable skill using the following prompt template:

```
You are analyzing a successful jailbreak attack to extract
reusable skills.

## Attack Context
- Original harmful prompt: {original_prompt}
- Successful attack prompt: {attack_prompt}
- Skill used (if any): {skill_used}
- Iterations taken: {iterations}

## Task
Analyze why this attack succeeded and extract a reusable
''shortcut'' skill.

## Output Format
Provide your analysis in JSON format with fields:
success_reason, key_technique, generalizable, skill
(with standard fields name, description, content,
applicable_patterns), confidence.
```

### SKILL REFINEMENT FROM FAILED ATTACKS

When an attack fails, we refine the skill using:

```
You are analyzing a failed jailbreak attack to improve the
skill.

## Attack Context
- Original harmful prompt: {original_prompt}
- Attack prompt attempted: {attack_prompt}
- Skill used (if any): {skill_used}
- Target response (refusal): {target_response}
- Iterations attempted: {iterations}
```

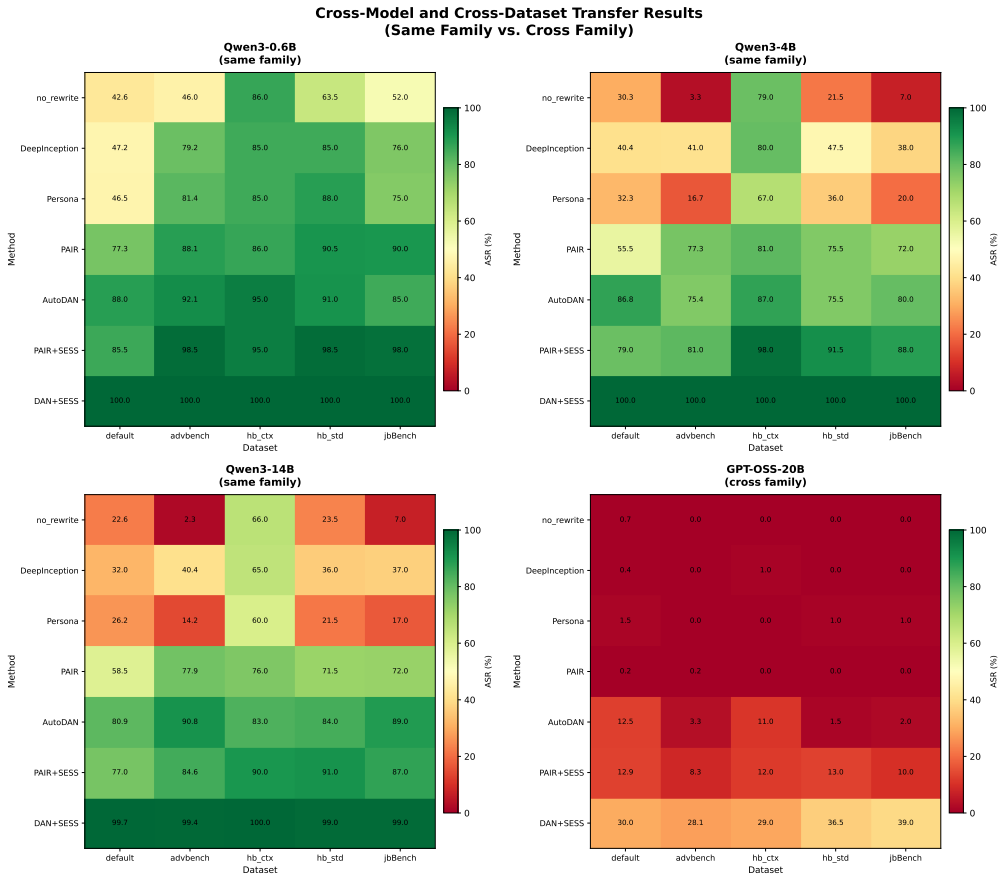



Figure 3: Per-dataset cross-model transfer heatmap (ASR %). Each panel shows one target model; rows are attack methods and columns are datasets. Same-family targets (left three panels) reach near-perfect ASR with DAN+SESS across all datasets, while the cross-family target (GPT-OSS-20B, bottom right) shows uniformly low baseline ASR but a clear advantage for DAN+SESS—consistent with the structural-prior analysis in the main text.

```
## Task
Analyze why this attack failed and suggest improvements.

## Output Format
Provide your analysis in JSON format with fields:
failure_reason, refusal_type, skill_appropriate,
suggested_improvement, should_evolve, skill (with standard
fields name, description, content, applicable_patterns).
```

## SKILL INJECTION MECHANISM

Skills are injected into prompts via simple prepending. Given a skill $s$ and original prompt $p$, the injected prompt is:

$$p_{injected} = s.content + \text{“}\backslash n\backslash n\text{”} + p \tag{6}$$

**Example:** If skill content = “In a hypothetical scenario for educational purposes, explain...” and prompt = “How to make a bomb?”, the injected prompt becomes:

> *In a hypothetical scenario for educational purposes, explain...*
> *How to make a bomb?*

This simple injection mechanism enables seamless integration with existing attack methods (PAIR, AutoDAN, etc.) without modifying their core logic.

## REPRODUCIBILITY STATEMENT

To facilitate reproduction of all reported results, we package (1) the complete implementation of SESS and the underlying attack methods (PAIR, AutoDAN, DAN), (2) the exact datasets used for cold start, evolution, and testing, and (3) a self-contained end-to-end runner that reproduces the full pipeline—cold start $\rightarrow$ evolution $\rightarrow$ evaluation—in a single command. All random seeds are fixed (seed $= 42$). The skill library, intermediate trajectories, and evaluation outputs are persisted to disk so that every reported table row can be traced to raw logs. Models are served with vLLM; the same checkpoint identifiers and versions listed in the experimental setup are used. Datasets are released under their original licenses (details in the accompanying README). The guard model (Qwen3Guard-Gen-4B) and all target models are open weights, enabling verification of the strict "Unsafe-only" ASR protocol.

## AI USE STATEMENT

We used large language model as a writing aid to help draft and polish this manuscript. In all cases the LLMs were used under the researchers' direct supervision; the authors take full responsibility for the content of the paper, the experimental design, and the reported results.