

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

COGNITION: From Evaluation to Defense against Multimodal LLM CAPTCHA Solvers

Junyu Wang, *Missouri University of Science and Technology;* Changjia Zhu, *University of South Florida;* Yuanbo Zhou, *Missouri University of Science and Technology;* Lingyao Li, *University of South Florida;* Xu He, *Visa Inc.;* Mingkui Wei, *George Mason University;* Junjie Xiong, *Missouri University of Science and Technology*

https://www.usenix.org/conference/usenixsecurity26/presentation/wang-junyu

This paper is included in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

ARTIFACT EVALUATED
USENIX
AVAILABLE

COGNITION: From Evaluation to Defense against Multimodal LLM CAPTCHA Solvers

Junyu Wang[1], Changjia Zhu[2], Yuanbo Zhou[1],
Lingyao Li[2], Xu He[3], Mingkui Wei[4], Junjie Xiong[1]*
[1]*Missouri University of Science and Technology;*
[2]*University of South Florida;* [3]*Visa Inc.;* [4]*George Mason University*

Abstract

This paper studies how multimodal large language models (MLLMs) undermine the security guarantees of visual CAPTCHA. We identify the attack surface where an adversary can cheaply automate CAPTCHA solving using off-the-shelf models. We evaluate 7 representative MLLMs on 18 real-world CAPTCHA task types, measuring single-shot accuracy, success under limited retries, end-to-end latency, and per-solve cost. We further validate our findings through a supplemental external dataset and an adaptive-attacker setting with session memory, while also analyzing the impact of task-specific prompt engineering and few-shot demonstrations on solver effectiveness. We reveal that MLLMs can reliably solve recognition-oriented and low-interaction CAPTCHA tasks at human-like cost and latency, whereas tasks requiring fine-grained localization, multi-step spatial reasoning, or cross-frame consistency remain significantly harder for current models. By examining the reasoning traces of such MLLMs, we investigate the underlying mechanisms of why models succeed/fail on specific CAPTCHA puzzles and use these insights to derive defense-oriented guidelines for selecting and strengthening CAPTCHA tasks. To validate these principles, we present a proof-of-concept by hardening a vulnerable CAPTCHA type using our guidelines. We demonstrate that incorporating fine-grained localization and implicit counting reduces the success rate of state-of-the-art MLLMs from over 95% to 0%, confirming that structural changes can effectively mitigate the threat. We conclude by emphasizing the urgent need for CAPTCHA redesign as MLLM capabilities increasingly threaten existing defenses. Code Availability[1].

1 Introduction

Today, visual CAPTCHA *(Completely Automated Public Turing test to tell Computers and Humans Apart)* challenges are an essential security mechanism in the modern web ecosystem. CAPTCHA has been integrated into more than 10-million commercial websites worldwide, reflecting its role as a standard security layer in online services [13]. Online services rely on CAPTCHAs to distinguish human users from automated scripts, thereby protecting registration flows, content submission portals, and valuable resources from large-scale abuse. Their ubiquity reflects their continued importance in securing everyday online interactions.

However, the advancement of multimodal large language models (MLLMs) poses a serious threat to this web security mechanism. Unlike prior attack methods that require task-specific training data and architectures [10, 16, 24], MLLMs can jointly interpret images and natural language instructions through a unified interface, precisely the capability that modern CAPTCHAs assume only humans possess. Recent work has demonstrated that MLLMs can serve as general-purpose CAPTCHA solvers [6, 27], building on earlier efforts that evaluated multimodal models under fixed templates or curated synthetic datasets [7, 17, 28]. Meanwhile, commercial CAPTCHA-solving services, often referred to as CAPTCHA farms [26], have begun combining MLLM automation with human labor to improve throughput [1, 4, 5]. If off-the-shelf MLLMs can reliably bypass CAPTCHAs at low cost and high speed, the consequences for web security are severe: attackers could automate account creation, credential stuffing, content spam, and resource scraping at unprecedented scale.

This security concern motivates our study. Rather than merely confirming that MLLMs can solve CAPTCHAs, we ask a more critical question: *which CAPTCHA designs can still raise the bar against MLLM-based solvers?* Existing studies mostly report accuracy on narrow benchmarks while overlooking solving time, retry limits, and monetary cost that determine whether an attack is viable at scale [19, 33], and they rarely analyze why models succeed on some tasks but systematically fail on others [14]. Motivated by these gaps, we consider the following research questions:

- RQ1: How well can MLLMs solve diverse visual

*Corresponding author.
[1]https://doi.org/10.5281/zenodo.20406852

CAPTCHA tasks under practical constraints such as limited time and retries?

- RQ2: How do prompting strategies, such as direct prompting, optimized instructions, and few-shot demonstrations, affect solver performance across different CAPTCHA types?
- RQ3: What reasoning behaviors do MLLMs exhibit during successful and failed attempts, and what do these behaviors reveal about the challenges that CAPTCHAs pose to automated solvers?
- RQ4: What strategies should web service providers adopt to deploy CAPTCHA schemes that remain effective against increasingly capable MLLMs?

To answer these questions, we evaluate seven representative MLLMs on 18 real-world visual CAPTCHA types under multiple prompting strategies. Our results reveal a pronounced hardness gap across task types. Recognition-oriented and low-interaction CAPTCHAs, such as Path_Finder, Select_Animal, and Image_Recognition, are already solvable by MLLMs through simulated human reactions within realistic retry and time budgets. In contrast, tasks including Click_Order, Place_Dot, Pick_Area, Dice_Count, and Patch_Select, remain consistently challenging: even with strong models and carefully designed prompts, they exhibit low success rates and substantially higher per-solve time and cost. We further study a black-box attack setting in which an adversary repeatedly sends CAPTCHA images and instructions to an off-the-shelf MLLM API, treating each model call as one attempt and stopping until the challenge is solved or a fixed retry cap is reached. Under this setting, most recognition-style CAPTCHAs can be bypassed easily, raising questions about the reliability of current human-verification systems. In addition, to confirm above findings generalize beyond the main benchmark and remain robust against stronger attacker strategies, we further evaluate a supplemental external dataset and an adaptive session-memory attack setting.

Importantly, through analysis of reasoning traces from MLLMs, we uncover that the observed robustness of these harder CAPTCHA types is neither accidental nor an artifact of poor prompting. Instead, it reflects deeper structural limitations of current MLLMs. We find that tasks such as Click_Order, Place_Dot, Pick_Area, Dice_Count, and Patch_Select share a set of structural properties that consistently stress these models. Specifically, they require fine-grained spatial grounding in continuous image space, precise binding between objects and their locations, and explicit counting or aggregation under visual clutter. While MLLMs often exhibit correct high-level reasoning in natural language, they frequently fail to translate this reasoning into exact coordinates, stable multi-object bindings, or accurate counts. These failure modes persist even under optimized prompts, retries, and few-shot guidance. Besides identifying which CAPTCHA types remain hard for MLLMs, we analyze the causes of these structural hardness patterns, distill the insights into a concrete defense methodology, and validate it by strengthening a vulnerable CAPTCHA design.

Building on these observations, our key insight is that CAPTCHA robustness against MLLMs is governed less by semantic recognition difficulty and more by grounding precision and compositional consistency. This insight motivates a defense strategy that shifts CAPTCHA design away from discrete recognition or classification and toward challenges that tightly couple perception with fine-grained localization, ordering, and counting. As a proof of concept, we apply this guideline to structurally redesign an existing MLLM-solvable CAPTCHA task, Select_Animal, and demonstrate that the success rate of state-of-the-art MLLMs is substantially reduced, decreasing from over 95% to 0%.

We summarize our contributions as follows:

1. We formulate a realistic black-box threat model for MLLM-based CAPTCHA solving and propose an evaluation framework that jointly considers accuracy, finite-retry behavior, end-to-end latency, and token-based cost.
2. We empirically evaluate seven MLLMs on 18 real-world CAPTCHA task types, revealing a pronounced and stable hardness gap and identifying which task types are already broken, nearly broken, or still robust. We further strength this evidence using an external evaluation set and an adaptive session-memory test that examines whether prior feedback enables a stronger attacker to weaken hard task types.
3. We analyze MLLM reasoning traces on the hardest task types, extract the structural factors underlying their robustness, and translate these insights into a concrete defense methodology. We validate this methodology through stronger CAPTCHA design transformations.

2 Related Work

CAPTCHA Solvers: in response to each other's advancement [18, 21]. Traditional text- and image-based CAPTCHAs are weakened by deep-learning methods employing convolutional models and segmentation-based techniques [22, 23, 29]. More advanced architectures, including generative adversarial network-based solvers and transformer-based recognizers, further improved solvers robustness against noisy and diverse schemes [32, 34]. Besides visual recognition, reinforcement-learning approaches showed that even behavior-based CAPTCHAs can be bypassed by agents learning human-like interaction pattern [2, 30]. However, the emergence of reasoning-based CAPTCHAs introduces semantics, interactive, and multi-modal designs, necessitating solvers with advanced logical-reasoning capabilities alongside strong image-comprehension skills [9, 14, 31].

MLLM in CAPTCHA: With the advent of MLLMs and

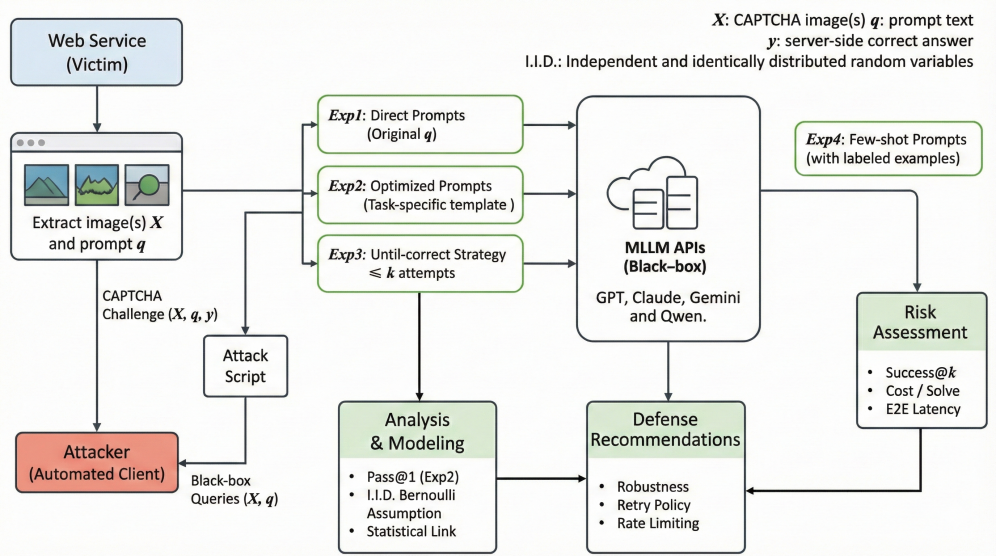


Figure 1: CAPTCHA robustness evaluation framework against MLLMs.

vision-language models (VLMs), recent work has begun to treat CAPTCHA solving as general interactive reasoning tasks. Teoh et al. propose Halligan, a generalized visual CAPTCHA solver built around an agentic VLM that plans browser actions and achieves high success rates across widely deployed CAPTCHA schemes [27]. Deng et al. presents Oedipus, which casts CAPTCHA solving as LLM-guided multi-step reasoning over challenge instructions and candidate visual cues, combining chain-of-thought prompting with perception modules [6]. In parallel, Ding et al. design IllusionCAPTCHA which introduces visual illusions that cause state-of-the-art VLM and LLM-based agents to hover near random-guess performance, which illustrates that the carefully crafted perception traps can systematically mislead LLM/VLM CAPTCHA solvers [8].

Building on this trend, subsequent research has shifted from proposing individual solvers to systematically benchmarking multimodal agents [14, 31], analyzing their failure modes [25], and designing LLM-aware CAPTCHA defenses [11]. However, these efforts largely focus on accuracy and query statistics while overlooking the practical time budgets for CAPTCHA solving, MLLMs deep reasoning (i.e., thinking) capability, and deep analysis on why these models would fail at specific CAPTCHA puzzles. Our work complements these early efforts by explicitly incorporating solving latency as a core metric, examining the reasoning traces of state-of-the-art MLLMs on diverse CAPTCHA types, and deriving concrete, defense-oriented design guidelines.

3 Framework Development and Threat Model

This section describes a framework for evaluating CAPTCHA robustness against MLLMs, as shown in Figure 1. We introduce the problem formulation, outline the attack strategy, and present the modeling assumptions used in this study.

3.1 Threat Model and Attack Assumptions

We consider a generic web service that deploys visual CAPTCHAs as part of its abuse-mitigation pipeline, for example, to protect account creation, content submission, or access to high-value resources. When a user reaches such a protected step, the service displays a CAPTCHA widget in the browser. Each CAPTCHA challenge has three pieces of information, which we will refer to throughout the paper: (i) X: one or more images shown in the CAPTCHA widget (a single picture, a grid of tiles, or a composed layout) that form the visual input to the solver; (ii) q: the human-readable instruction on the page (e.g., "click all squares with traffic lights"); (iii) y: the unique correct answer stored and checked on the server.

In this setting, only the images X and the instruction q are visible in the browser; the ground-truth answer y is never revealed to the client. In typical deployments, after the user submits an answer, the CAPTCHA widget returns a response token r to the browser. The client then forwards r to the relying party's backend, which verifies r with the CAPTCHA provider and uses the resulting pass/fail outcome to accept or reject the request.

As a result, benign users simply see the CAPTCHA, solve it by hand, and continue their normal interaction with the service. The malicious users (i.e., *attacker*), in contrast, controls an automated client (or a script that drives a browser) with the goal of solving CAPTCHAs at scale so that large numbers of malicious requests can be automated. In this scenario, the web service together with its CAPTCHA deployment plays the role of the *victim*.

We assume that the attacker can capture the images X as rendered in browsers (e.g., via screenshots or image URLs), read human-facing instructions q, and submit answers on the client-side. However, the attacker cannot inspect/modify the server-side verification logic and never directly observes y. The attacker can leverage one or more MLLMs as automated CAPTCHA solvers. These models are accessed through black-box APIs, either from commercial providers or self-hosted open-source models. For each attempt, the attacker submits the images X and a prompt derived from q, optionally enhanced with task-specific prompt optimizations such as clearer reasoning instructions or constrained output formats, and receives an instruction text response from the MLLMs. We additionally consider an adaptive setting in which the attacker retains prior reasoning across attempts based on the accumulated interaction history together with binary pass/fail feedback. The attacker has no access to gradients, internal activations, or model parameters; the models operate strictly as off-the-shelf oracles.

3.2 MLLM Modeling Assumptions

We evaluate MLLM CAPTCHA-solving capabilities on an offline dataset under four regimes representing plausible solver behaviors. Consistent with the formulation in Section 3.1, the solver receives only the client-side inputs (X, q) and produces a response $\hat{y}$. Results are determined by validating $\hat{y}$ against the ground truth y, without interacting with verification services. To systematically evaluate performance across multiple levels of adversary capability, we use the `Dice_Count` task (where the requirement is to sum the pips on dice faces) and employ the following four specific strategies:

(i) Direct Prompts (Exp1): We forward the human-facing instruction q directly to the model. For Dice_Count, the prompt is: "*Sum up the numbers on all the dice.*" This simulates a naive attacker who relies entirely on the MLLM's zero-shot instruction following without optimization.

(ii) Optimized Prompts (Exp2): The attacker utilizes a task-specific template that clarifies reasoning steps and enforces a machine-parseable format. For Dice_Count, the prompt is expanded to: "*Identify all visible dice. Sum the pips on their top faces only. Output the final integer sum in JSON format (e.g., {'answer': 14}).*" This isolates the model's visual reasoning capability from formatting errors.

(iii) Until-correct Strategy (Exp3): We model a determined attacker using an "until-correct" strategy with a budget of k attempts. Upon failure, the solver requests a new instance of the same task type (e.g., a fresh Dice_Coun puzzle) and retries using the *Optimized Prompt* (as in Exp2). This accounts for the attacker's willingness to trade higher latency and cost for a successful bypass.

(iv) Few-shot Prompts (Exp4): To maximize solver performance on the hardest tasks, we prepend labeled examples to the optimized prompt. The model is presented with a context such as: "*User: Sum the dice. Assistant: {'answer': 5}*," followed by the target query.

To be noted here, the prompt template is fixed during evaluation to ensure consistent benchmarking for each model–task-type pair. Our analysis relies on the following assumptions. First, conditional on a fixed model m, task type t (defined in Section 4.1), prompt template, and hyperparameters, we treat each attempt as an independent and identically distributed (i.i.d.) Bernoulli trial. This allows us to model the multi-attempt solving in Exp3 using standard binomial reasoning and to analytically estimate the probability of success within k attempts from the single-shot results in Exp2.

Then, we focus strictly on the visual reasoning capabilities of the models. We do *not* consider out-of-scope vectors such as compromising the web service, exploiting widget implementation bugs, or human-in-the-loop attacks. Finally, while emerging agentic frameworks allow for tool use (e.g., precise mouse control), the fundamental bottleneck remains the visual perception and reasoning capability of the underlying MLLM. If the model cannot correctly identify the target coordinates or counting logic, external tools cannot recover the correct solution. Therefore, our evaluation of the MLLMs can effectively bound the performance of agentic systems.

4 Methods and Experiments

4.1 Dataset and Tasks

We conduct Exp1 to Exp4 on the recent CaptchaWorld dataset [14], which collects visual CAPTCHAs from both real-world services and synthetic environments. The raw dataset covers 20 task types and more than 400 instances; each instance contains the image(s), a natural-language instruction, and the ground-truth answer. In this work, we apply cleaning and standardization to enable a unified MLLM-based evaluation: we remove two task types incompatible with our pipeline, correct and align labels and metadata, and normalize answer formats. The resulting main benchmark contains 18 task types and 378 instances.

For the main benchmark, we group the 18 task types into four task families:

- **Counting and aggregation**: Dice_Count, Dart_Count;
- **Pointing and path-based localization**: Place_Dot, Geometry_Click, Pick_Area, Misleading_Click, Click_Order, Path_Finder;
- **Grid selection and matching**: Bingo, Patch_Select, Image_Recognition, Select_Animal, Unusual_Detection, Object_Match, Image_Matching;
- **Relational and transformation puzzles**: Coordinates, Connect_Icon, Rotation_Match.

Detailed task definitions, dataset statistics, and cleaning rules are deferred to Appendix B, with a summary of the supplemental set in Table 7. We also consider a supplemental external evaluation set to test whether the main findings generalize beyond CaptchaWorld; details are provided later in Section 5.4.

4.2 Evaluation Metrics

We evaluate CAPTCHA solving along three key dimensions: accuracy, time cost, and finite-retry behavior. Considering accuracy, we choose to use the first-time-pass-rate (Pass@1) as a metric. For example, Pass@1 measures the accuracy of the model when allowed strictly one attempt per instance, effectively representing the first-time pass rate. In addition, we further consider the end-to-end latency and finite-retry behavior as our metrics to understand MLLM latency and failure-retry performance in real-world situations.

Pass@1: For each sample, let $\hat{y}_i$ be the answer produced by the model in its first attempt. We define

$$\text{Pass@1} = \frac{1}{N}\sum_{i=1}^{N} 1\{\hat{y}_i = y_i\}, \tag{1}$$

where N is the number of evaluation samples. We analogously compute Pass@1 per task type. Since each CAPTCHA instance has a single correct answer and the service returns only a binary pass/fail signal, Pass@1 aligns closely with how real deployments validate submissions.

End-to-end latency: For each model call, we record the end-to-end latency from sending the request to receiving a complete, parseable answer. This metric directly captures the time between submitting a CAPTCHA to the model and learning whether it passed verification.

Finite-retry behavior: Real services typically cap the number of attempts per session. To summarize behaviour under a retry budget k, we report *Success@k*, the probability that this attack succeeds within k attempts. We derive these quantities from single-shot Pass@1 under a simple Bernoulli model described in Section 4.5.

4.3 Models and Inference Setup

We evaluate seven representative MLLMs from major providers, covering both closed-source flagships and strong open-source vision–language models (Table 1). For GPT-series models we consider different `reasoning_effort` settings (e.g., *Medium* vs. *None*) to capture the effect of explicit reasoning modes.

To ensure comparability, we abstract a unified inference interface

$$\text{Invoke}(M, X, q; \theta) \to (a, \text{meta}), \tag{2}$$

where M is the model, X the CAPTCHA image(s), q the instruction prompt, and θ shared decoding parameters (temperature, maximum output length). The returned answer a is a short json text (e.g., ""target click position"":[""x"":405,""y"":535]); meta contains provider-specific metadata such as token counts and timestamps.

Table 1: Multimodal models evaluated in our study.

Provider	Model	Snapshot
OpenAI	GPT-5.1 (Medium)	2025-11-13
OpenAI	GPT-5.1 (None)	2025-11-13
OpenAI	GPT-5 (Medium)	2025-08-07
Anthropic	Claude Sonnet 4.5	2025-09-29
Google	Gemini-2.5-Pro	2025-01
Google	Gemini-2.5-Flash	2025-01
Fireworks[†]	Qwen-3-VL-235B-Instruct	2025-09-24

[†]Exp1–Exp4 used Fireworks. Exp5 used OpenRouter for the same Qwen3-VL-235B-A22B-Instruct model after Fireworks stopped serving it.

For each task type we constrain the output format to match the ground-truth answer (e.g., a single index, a coordinate pair, or a short integer). When APIs support JSON-mode or schema enforcement we use it, otherwise we instruct the model to output JSON-only and discard responses that contain extraneous text. For an error analysis purpose, we additionally ask the model to verbalize its reasoning in additional experiments.

4.4 Experimental Protocol

Our evaluation consists of the following four complementary experiments. These four experiments (**Exp1** to **Exp4**) share the same dataset and metrics but differ in prompting strategies (defined in Section 3.2) and retry settings.

Exp1 (Direct Prompts): We benchmark the raw capability of MLLMs by employing the *Direct Prompts* strategy (Section 3.2-i). For every model and task type, we perform a single invocation per sample using the original human-facing instruction. This establishes a baseline for off-the-shelf solver performance without prompt engineering.

Exp2 (Optimized Prompts): To isolate the effect of instruction tuning, we evaluate all models using the *Optimized Prompts* strategy (Section 3.2-ii). This experiment measures the performance gain achievable solely through clearer reasoning rules and standardized output formatting, serving as the primary basis for our accuracy analysis. We also apply this same setup to the external evaluation set described in Section 5.4 to examine whether the insights observed on the main benchmark generalize beyond its original data sources.

Exp3 (Until-correct Strategy): To assess operational viability, we model an attacker using an "until-correct" strategy with a retry budget k. Rather than performing redundant queries, we derive Success@k and expected cost analytically from Exp2's single-shot accuracy, relying on the Bernoulli model detailed in Section 4.5. We empirically validate the accuracy of this extrapolation on a subset of tasks.

Exp4 (Few-shot Prompts): For the specific task types iden-

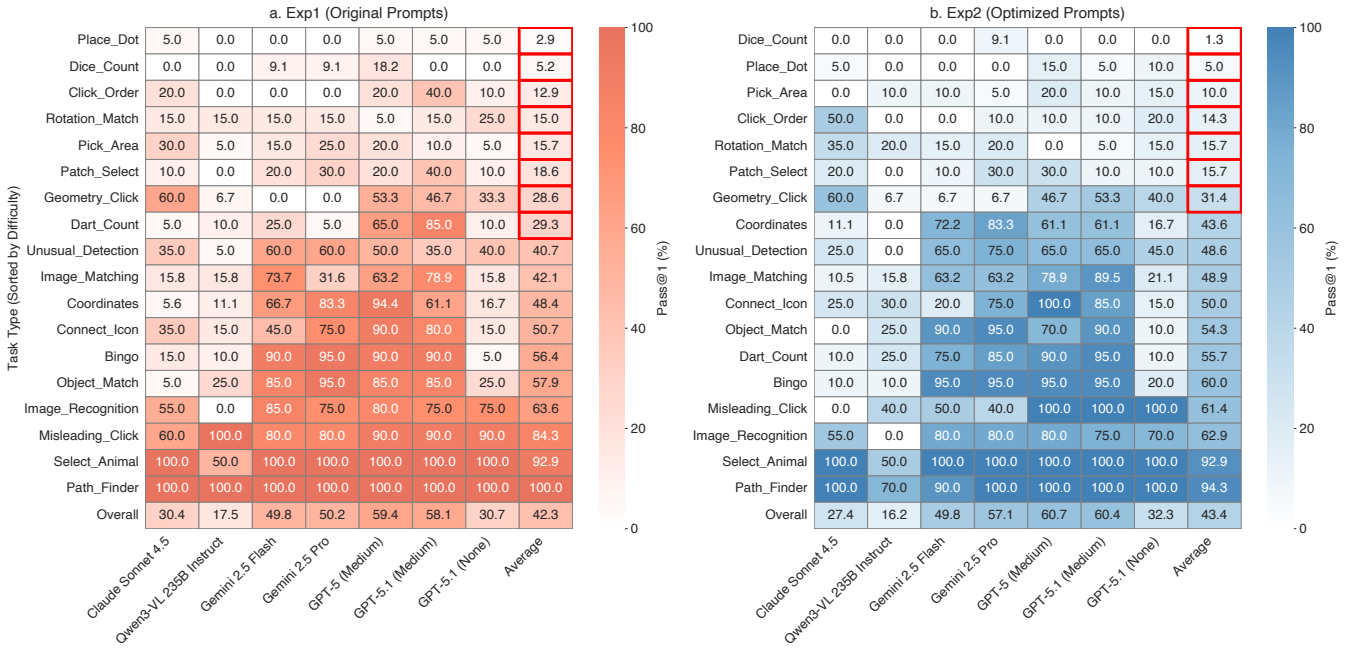


Figure 2: Heatmap of CAPTCHA task difficulty: (a) Exp1 (original prompts), and (b) Exp2 (optimized prompts). Task types (rows) are sorted by cross-model average Pass@1. Columns correspond to MLLMs, and each cell reports Pass@1 (%).

tified as "hard" in Exp2, we apply the *Few-shot Prompts* strategy (Section 3.2-iii). By prepending two solved examples to the context, we test whether the observed failures are due to instruction misunderstanding or fundamental limitations in visual reasoning.

Exp5 (Adaptive session-memory): We further evaluate whether session-level adaptation can improve the attacker's capability. In this setting, the attacker retains reasoning traces and submitted answers from earlier attempts within the same session, and may revise its local strategy after each failure. The only external feedback is the binary pass/fail outcome; ground-truth labels, target coordinates, counts, and corrective hints are never revealed. For each task type, we run independent memory-isolated rounds, resetting the conversation context between rounds. This experiment complements Exp3 by directly testing adaptive behavior rather than estimating independent fixed-prompt retries from single-shot Pass@1. For Qwen3-VL in Exp5, we used the same Qwen3-VL-235B-A22B-Instruct model through OpenRouter, with a 2026-05-20 access date, because Fireworks no longer offered this model at this time.

4.5 Single-shot Accuracy and Finite-retry Cost

Non-adaptive finite-retry modeling (Exp3). For a fixed model m and CAPTCHA type t, all attempts use the same prompt template, decoding hyperparameters, and parsing rules, and are evaluated on instances drawn from the same dataset. Conversational memory and tool use are disabled, so randomness arises only from the sampled instance and the model's internal sampling. Under these controlled conditions, it is natural to treat each attempt's correctness as an independent Bernoulli trial with success probability $p_{m,t}$, which we estimate by the corresponding Pass@1 from Exp2.

Given this model, if an attacker is allowed at most k attempts on type t, the probability that at least one attempt succeeds is

$$\text{Success@}k = 1-(1-p_{m,t})^k,$$

and the expected attempt count under this until-correct strategy is

$$E[A] = \frac{1-(1-p_{m,t})^k}{p_{m,t}}.$$

These quantities let us summarize, for each (m,t), how a small retry budget amplifies single-shot accuracy and how many API calls an attacker is expected to make.

We estimate per-call monetary cost from provider price tables. Let c_{in} and c_{out} be the per-thousand-token prices for prompts and completions, and let t_i^{in} and t_i^{out} be the corresponding token counts for sample i. The cost of that invocation is

$$\text{cost}_i = \frac{t_i^{\text{in}}}{1000}\,c_{\text{in}} + \frac{t_i^{\text{out}}}{1000}\,c_{\text{out}}.$$

Multiplying the average per-call cost by $E[A]$ yields an estimate of the expected cost per successful solve. For comparison, prior measurements of human CAPTCHA-solving

services report roughly 80–90% one-shot accuracy, solving times of a few to tens of seconds, and market prices on the order of 0.5–2 US cents per challenge [3,15,16]. We use these ranges as a reference when interpreting the cost–performance trade-offs of MLLM-based solvers.

Adaptive session-memory setting (Exp5). The Bernoulli extrapolation above applies only to non-adaptive retries with a fixed per-attempt success rate. Exp5 instead directly evaluates whether an attacker can improve across attempts by using session memory together with binary pass/fail feedback. For each task type t, we run n memory-isolated adaptive rounds. At the beginning of each round, the conversation context is reset. Within a round, the attacker is allowed up to k attempts; after each failed attempt, it can review its previous reasoning traces, submitted answers, and the binary failure signal, then revise its prompt or local strategy before proceeding to another sampled instance from the same task type.

We report observed Success@k as the fraction of the n rounds in which at least one of the first k adaptive attempts succeeds:

$$\widehat{\text{Success}}@\text{k}_t = \frac{1}{n}\sum_{r=1}^{n} \mathbf{1}\{\exists a \le k : \hat{y}_{t,r,a} = y_{t,r,a}\},$$

where r indexes the memory isolated round, a indexes the attempt within that round, and $y_{t,r,a}$ is the ground truth for the sampled instance. Because this metric is based on k rounds of observations per task type, we use it as a stress test to assess whether feedback can qualitatively change the hardness conclusion under a limited retry budget, rather than as a precise deployment-wide success-rate estimate.

5 Experiment Results

The results reveal a clear difficulty gap across CAPTCHA types. Recognition-oriented and low-interaction tasks, such as object selection or simple grid matching, are already solved reliably by MLLMs with high accuracy, strong performance under small retry budgets, and low per-solve cost. In contrast, tasks requiring fine-grained localization, ordering, or counting remain challenging even for the strongest models and therefore serve as the most promising directions for defense-oriented deployment. We present our empirical findings in the order of RQ1–RQ3 and discuss defense implications (RQ4) in Section 6.

5.1 RQ1: Overall Solving Capability

RQ1 asks how well MLLMs can solve diverse CAPTCHA tasks under realistic constraints such as limited time, retries, and cost. We first examine single-shot accuracy across models and task types, then connect these results to finite-retry success rates and time/cost trade-offs.

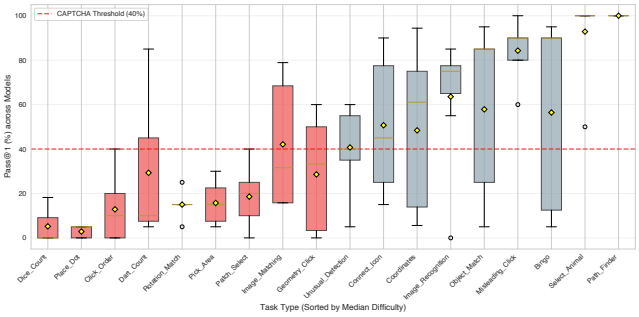


Figure 3: Cross-model Pass@1 distributions per task type in Exp1 (original prompts). Each box shows the spread across models. The dashed line marks a 40% threshold.

Figure 2(a) summarizes Exp1, where models receive the original human-facing instructions. The cross-model average Pass@1 is 42.3%, showing that MLLMs already solve a substantial fraction of CAPTCHAs even with unoptimized prompts. Strong closed-source models (GPT-5, GPT-5.1, Gemini 2.5 Pro/Flash) reach around 50–60% Pass@1, while weaker systems lag noticeably behind.

More importantly, there is a pronounced task-level hardness gap. A small set of types, including Place_Dot, Dice_Count, Click_Order, Rotation_Match, Pick_Area, Patch_Select, and Geometry_Click, have cross-model averages well below 30%, with the first six mostly below 20%. In contrast, recognition-style tasks such as Path_Finder and Select_Animal already exceed 60% on average, suggesting that they are easy targets for automated solvers.

To verify that the hardness gap is not model-specific, we examine cross-model stability. Figure 3 plots Pass@1 distributions for each task type, with a dashed line at 40% as a working CAPTCHA threshold. Tasks whose distributions lie well below this are consistently difficult across models, while those above it are reliably easy.

Exp2 replaces the prompts with task-optimized instructions. As shown in Figures 2(b) and 4, prompt optimization yields modest gains for strong models (e.g., GPT-5 increases from 59.4% to 60.7%, GPT-5.1 from 58.1% to 60.4%), and slightly raises the overall average from 42.3% to 43.4%. At the task level, however, the separation between easy and hard types becomes sharper: recognition-oriented tasks such as Bingo, Object_Match, Image_Recognition, and Path_Finder move further above 60%, while six types—Dice_Count, Place_Dot, Pick_Area, Click_Order, Rotation_Match, and Patch_Select—remain below 20% even after optimization. Geometry_Click improves modestly but stays around 30%.

The experimental results from Exp1 and Exp2 reveal a distinct performance dichotomy. While prompt optimization further boosts the accuracy of recognition-based tasks, it fails to significantly improve performance on tasks requiring complex spatial grounding or counting. Consequently, the hard-

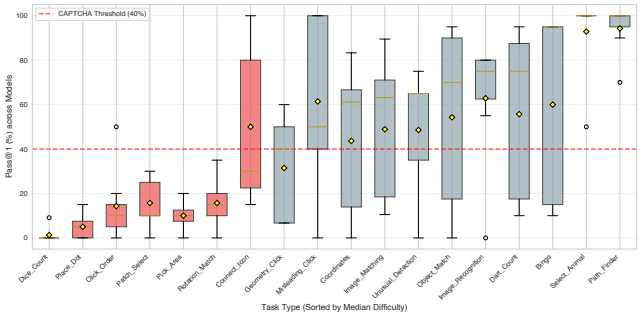


Figure 4: Cross-model Pass@1 distributions per task type in Exp2 (optimized prompts). Compared to Exp1, most recognition-style tasks rise well above the 40% threshold, while the six hard task types remain low across models.

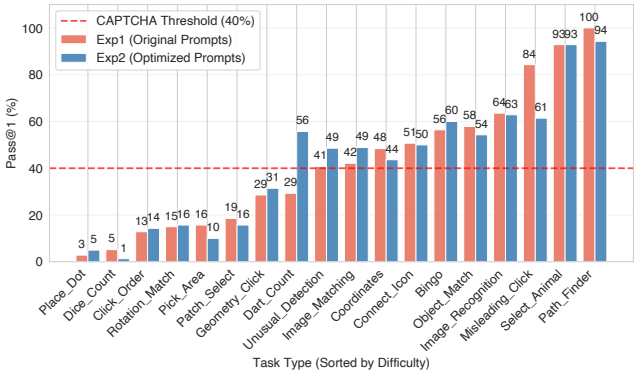


Figure 5: Per-task Pass@1 for GPT-5 (Medium) in Exp1 (original prompts) and Exp2 (optimized prompts). The dashed line at 40% marks the hardness threshold.

ness gap observed in Exp1 is not a transient artifact of poor prompting but a stable characteristic of the tasks themselves. Based on this observation, we categorize the 18 task types into the following three security tiers to guide analysis:

- **Broken types:** Task types whose cross-model average Pass@1 exceeds 40% in both Exp1 and Exp2. This group contains recognition and simple grid-based families such as Path_Finder, Select_Animal, Misleading_Click, Image_Recognition, Bingo, Object_Match, Unusual_Detection, Image_Matching, Coordinates, Connect_Icon, and Dart_Count.

- **Borderline types:** Geometry_Click remains below 40% on average but shows clear upward trends when prompts are optimized; we treat it as effectively broken.

- **Hard types:** Patch_Select, Rotation_Match, Click_Order, Pick_Area, Place_Dot, and Dice_Count stay below 20% in both experiments with minimal improvement and are our candidate robust CAPTCHAs.

We focus on GPT-5 (Medium) as a representative strong solver, which attains the best overall performance. Figure 5 compares GPT-5's Pass@1 under original and optimized prompts. Most recognition tasks lie well above the 40% line (often above 80%) and benefit further from optimized prompts, whereas the six hard types remain below this threshold with only small or even negative changes. This shows that prompt engineering can polish already-vulnerable tasks but does not eliminate intrinsically hard ones.

To connect single-shot accuracy with realistic retry budgets, we map Exp2 Pass@1 to Success@3 using the statistical model in Section 4.5. Figure 6 shows that once Pass@1 exceeds 40%, Success@3 quickly rises above 80%, and tasks around 60% Pass@1 already reach over 90% Success@3. All broken types fall into this regime, implying that a small retry budget suffices to make them almost always solvable. The hard types remain in the lower-left corner: with Pass@1 below 30%, their Success@3 stays moderate even with three attempts, and Dice_Count and Rotation_Match remain effectively unsolvable.

Figure 7 reports the expected number of API calls until the first success, again derived from Exp2. Broken types cluster between one and two calls on average, while the six hard tasks typically require between two and ten calls, with Dice_Count and Rotation_Match at the upper end. Thus, even without an explicit retry cap, reliably solving hard CAPTCHAs demands substantially more queries.

We incorporate monetary cost and latency. From an attacker's perspective, RQ1 is not only about whether a CAPTCHA can be solved, but also whether solving it is economically and temporally viable at scale. Using the cost model in Section 4.5, Figure 8a plots per-call cost versus Pass@1, while Figures 8b and 8c show E2E latency and expected cost per successful solve. Broken types occupy the favorable region: high Pass@1, E2E times on the order of tens of seconds, and expected costs per success less than 10 cents. The six hard types sit in the opposite regime: lower accuracy, similar or higher latency, and one to two orders of magnitude higher expected cost per successful solve. Under current pricing, they are therefore economically unattractive for large-scale automated solving.

As a result, Exp1–Exp3 and the cost/latency analysis answer RQ1: under realistic retry budgets and current API pricing, many recognition-style CAPTCHAs are already solvable cheaply and reliably, whereas the six hard task types remain both statistically and economically unattractive to automated MLLM-based solvers.

5.2 RQ2: Impact of Prompting Strategies

RQ2 asks how prompting strategies affect solver performance across CAPTCHA types. We therefore compare direct prompting (Exp1), task-optimized prompts (Exp2), and few-shot

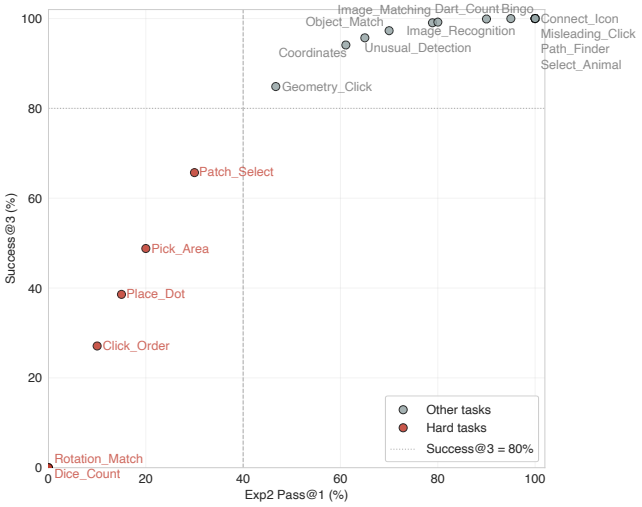


Figure 6: Mapping Exp2 Pass@1 to the finite-retry regime for GPT-5 (Medium). Each point is a task type; x-axis is single-shot Pass@1 and y-axis is the induced Success@3 under a three-attempt until-correct strategy.

Table 2: Few-shot (Exp4) performance of GPT-5 (Medium) on the six hard task types. E2E time is computed from the average per-question latency and reported in seconds.

Task	Pass@1	Avg E2E (s)
Click_Order	0.25	31.4
Dice_Count	0.00	92.2
Patch_Select	0.00	14.0
Pick_Area	0.00	38.7
Place_Dot	0.00	38.7
Rotation_Match	0.50	45.7

demonstrations (Exp4).

At the aggregate level, moving from original instructions to optimized prompts modestly improves overall Pass@1 for strong models (e.g., GPT-5 from 59.4% to 60.7% and Gemini 2.5 Pro from 50.2% to 57.1%). Figures 2 and 5 show that these gains mainly push already-solvable recognition tasks further into the high-accuracy regime, while the six hard types remain well below the 40% threshold. Prompt engineering thus amplifies capabilities where the model is already near the decision boundary but does not change which CAPTCHA families are intrinsically difficult.

We next ask whether few-shot demonstrations can substantially close the performance gap on the hard task types. In Exp4, we prepend two labeled examples per task type to the optimized prompt and evaluate GPT-5 under otherwise identical settings (Section 4.4). Table 2 summarizes the results.

Few-shot guidance provides only limited benefit. Compared to Exp2, Click_Order improves from 10% to 25% Pass@1 and Rotation_Match from 0% to 50%, but

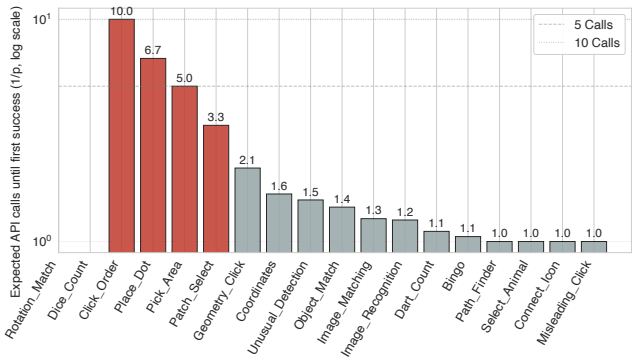


Figure 7: Expected number of API calls until the first success for GPT-5 (Medium), computed as $1/p$ based on Exp2 single-shot accuracy (log scale on the x-axis). Each point is a task type; reference lines at 5 and 10 calls highlight that hard tasks (shown in red in the plot) require significantly more calls than broken tasks.

Dice_Count, Patch_Select, Pick_Area, and Place_Dot remain unsolved. At the same time, multi-turn prompts with embedded examples substantially increase latency: the average end-to-end time per query on these six tasks is about 43.4 s, with Dice_Count exceeding 90 s. This corresponds to an average slowdown of roughly 2–3× relative to Exp2. In other words, few-shot prompting does not reliably solve hard CAPTCHAs and makes each query considerably more expensive.

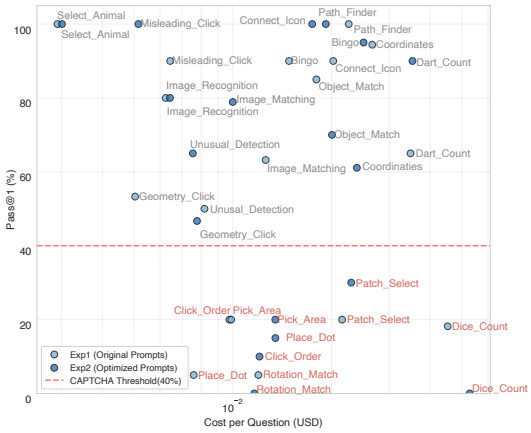
Taken together, the results of Exp1, Exp2 and Exp4 answer RQ2: task-specific prompt optimization and few-shot demonstrations improve MLLM solvers on already-vulnerable recognition CAPTCHAs, but do not qualitatively change which task families are hard. The same six hard types identified under direct prompting remain out of reach even with stronger prompting strategies.

5.3 RQ3: Which CAPTCHAs Break, Which Survive, and Why?

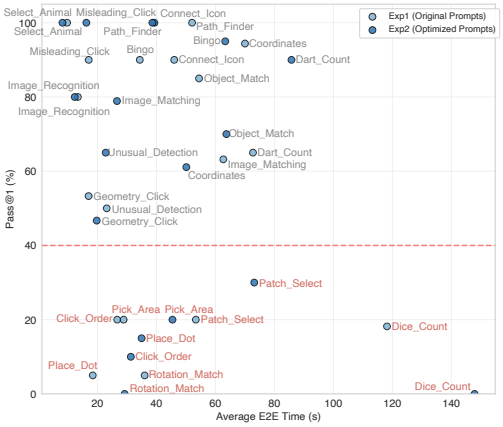
Combining the above results, we explain which CAPTCHA families current MLLMs reliably solve, which are close to being solved, and which remain comparatively robust.

Broken and near-broken types: Across Exp1 and Exp2, recognition-oriented and low-interaction tasks (e.g., object selection and coarse path finding) exhibit high Pass@1 for strong models and near-certain Success@3 with a small retry budget. These tasks require only coarse visual recognition with minimal spatial precision or multi-step reasoning, making them effectively broken under current MLLMs. Rotation_Match is an intermediate case: it remains moderately difficult with only optimized prompts, but becomes largely solvable with two few-shot demonstrations in Exp4, reducing to simple pattern matching. We therefore treat it as a near-broken type that is likely to fall completely as model

(a) Cost–performance frontier for GPT-5 in Exp1 and Exp2.



(b) Time–performance trade-off for GPT-5 in Exp1 and Exp2.



(c) Expected cost per successful solve versus Pass@1 across CAPTCHA task types for GPT5

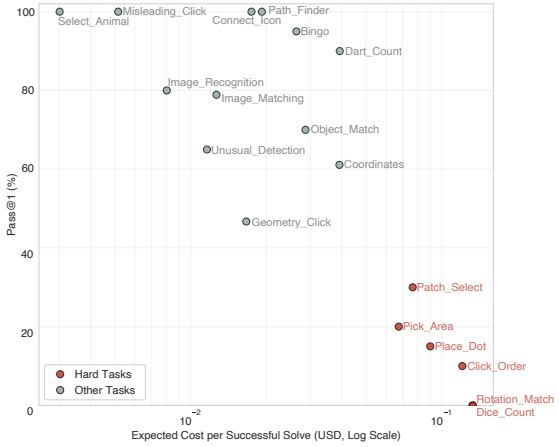


Figure 8: Cost and latency trade-offs for GPT-5 across CAPTCHA task types.

capabilities improve.

Persistently hard types and shared structure: After excluding broken and near-broken CAPTCHA types, five task types remain consistently hard across models and prompting regimes: Click_Order, Place_Dot, Pick_Area, Dice_Count, and Patch_Select. Even GPT-5 with optimized prompts and few-shot examples exhibits low Pass@1, limited Success@3, and higher time and token consumption on these tasks. As discussed in Section 6, they share several structural factors: a reliance on high-precision 2D localization, multi-object binding and ordering, and counting or aggregation under clutter. These properties stress aspects of spatial grounding and compositional reasoning that current MLLMs handle poorly.

Recurrent error modes: The reasoning traces logged for GPT-5 on these five task types exhibit recurring error modes:

- *Correct reasoning, incorrect discrete answer:* the model describes the right region or counting strategy, but the final coordinate, tile index, or numeric result does not match its own reasoning.
- *Local correctness, global inconsistency:* the model correctly interprets local parts of the scene (e.g., icons or dice groups) but aggregates them inconsistently, leading to wrong click orders or off-by-one counts.
- *Overconfident approximations in spatial decisions:* the model outputs locations that are "approximately right" from a human perspective but fall outside the exact tolerance required by the verification logic.

In the following subsections, to understand the mechanisms behind these failures, we analyze representative reasoning traces on the hardest task types, such as the dice counting challenge or spatial localization.

5.3.1 Counting Challenge

The Dice_Count task presents a 3×2 grid of panels, each containing several dice with pips and printed digits. The objective is to sum all visible top faces. Our analysis of Exp4 failures reveals that GPT-5 systematically overestimates the total by a wide margin, despite correctly articulating the task rules. For example, in one representative failure, the model answers 92 while the ground truth is 69. Its reasoning trace exposes the root cause:

> *"I examined the six sub-images and counted only the top faces of each die … Subimage sums: top-left* $= 5+4+1+5$, *top-middle* $= 2+2+4+3, \ldots$ *bottom-right* $= 3+5+7+7$. *Adding these gives* $15+11+17+11+16+22 = 92$."

We analyzed four such sampled items, finding overestimations ranging from +9 to +26. The rationales consistently emphasize that only upward faces are counted and provide panel-wise decompositions that appear internally coherent.

However, the model likely misinterprets side faces as top faces, double-counts dice that appear across panel boundaries, or misreads digits under perspective distortion. Once these local perceptual errors are propagated into the panel sums, the final aggregation step simply preserves the error. This suggests that the dominant difficulty is not the final arithmetic, but the robust perception and grouping of small objects in cluttered scenes.

5.3.2 Spatial Grounding Failures

Tasks requiring fine-grained spatial localization, such as Place_Dot and Click_Order, expose a fundamental disconnect between semantic understanding and coordinate prediction in MLLMs. We demonstrate the following three different tasks:

Coarse Path Tracing in Place_Dot. *Place_Dot* requires placing a dot at the end of a car's path in a stylized 3D city grid. In several Exp4 failures, GPT-5 describes a detailed and correct procedure for tracing the path from the start pin to its terminal tip, yet selects a coordinate hundreds of pixels away from the target. For instance, in one puzzle (target at $(290, 235)$), the model clicks $(565, 895)$—a Euclidean distance error of over 700 pixels. The model explains:

> *"I identified the bright green path connected to the car pin (start) at the top-right. Tracing the single continuous ribbon through its 90-degree turns leads to the opposite end near the lower central-left area. I clicked the visual center of that squared tip."*

Despite claiming to follow the path, the actual coordinates lie on a different segment or in a completely different region. This illustrates that while the global structure is understood conceptually, the model struggles to maintain a precise, metric-consistent representation of the polyline in image space, especially under perspective effects and 3D shading.

Qualitative Area Estimation in Pick_Area. In Pick_Area, the solver must identify the largest region among several enclosed shapes. GPT-5 frequently provides a plausible qualitative description of the partition structure but fails to perform precise geometric comparisons. In one failure case, the correct largest region occupied a bounding box from $(200, 300)$ to $(510, 500)$. However, the model selected a point at $(660, 110)$ in a completely different, smaller cell, reasoning:

> *"The broadest region lies in the top-right... I clicked near the geometric center of this largest area."*

The reasoning traces indicate that GPT-5 approximates "largest area" based on coarse visual impressions of width and shape rather than strict geometric computation. When several regions have comparable sizes or complex boundaries, this qualitative heuristic leads to consistent mis-ranking.

Boundary Violations in Click_Order. Similarly, Click_Order requires clicking icons in a specific sequence. GPT-5 generally identifies the correct semantic order (e.g., light bulb $\rightarrow$ wheel $\rightarrow$ briefcase) but fails at the final discrete mapping. In one example, the model's predicted coordinates were visually close to the target icons but deviated by approximately 40–300 pixels, exceeding the tolerance radius. The model effectively "solves the puzzle in words" but fails to ground these words into acceptable pixel coordinates.

5.3.3 Contextual Hallucinations

One of the most surprising failure modes emerged in the Patch_Select task, where the model is required to select grid patches containing a target object (e.g., a bridge). GPT-5 exhibited a recurring "binding mistake" where it confused the few-shot demonstration image provided in the prompt with the actual test image.

In an analysis of four distinct puzzles with different targets, e.g., bridge, hourglass, bench, maple leaf, GPT-5 consistently returned an empty set with high confidence. Strikingly, the "raw observation" segment of every trace described the exact same scene: a nighttime street festival with a large moon-like installation. This scene corresponded to the few-shot example, not the test query. For example, when asked to find a *bridge* in a new image, the model reasoned:

> *"I scan for classic bridge cues... The scene shows a nighttime street festival with a large illuminated sphere... but no continuous span with a walkable deck. Therefore, no grid cells contain a bridge."*

This task-level binding error arises from implicitly treating the example image as the query and is most prominent in Patch_Select. It suggests that for dense grid tasks, strong few-shot guidance can paradoxically degrade performance by inducing contextual hallucinations, contributing directly to the near-zero success rate on this task type.

5.4 External Validation of the Core Findings

The preceding results identify a stable boundary between CAPTCHA types that are already vulnerable to current MLLM-based solvers and those that remain more resistant. We next validate these core findings in three complementary ways: using a supplemental external evaluation set, using an adaptive session-memory test, and comparing our conclusions with prior solver and benchmark studies.

External dataset evaluation. To further examine whether the insights observed on this benchmark generalize beyond its original data sources, we additionally construct a disjoint supplemental external evaluation set of 80 instances from prior sources [12, 20]. We select these instances because they explicitly include the same failure-inducing properties identified in our main findings: counting under visual clutter, grid-level structural selection, and relation-based object-location binding. These categories therefore provide an external stress

Table 3: Results on the external evaluation dataset. "Pooled Pass@1" aggregates attempts from all 7 MLLM-based model configurations within each category, "Best per-model Pass@1" reports the highest Pass@1 achieved by any single model.

Category	Pooled Pass@1	Best-model Pass@1
Hole_Counting	0.0%	0.0%
Relation_Match	8.1%	16.7%
Symbol_Count	2.9%	10.0%

test of whether MLLM difficulty is driven by general structural properties rather than by the special data distribution in CaptchaWorld. The supplemental set follows the same instance schema and answer-format normalization as the main benchmark. Detailed task definitions, dataset statistics, and cleaning rules are deferred to Appendix B, with a summary of the supplemental set in Table 7. These three supplemental categories are revisited later in Table 4, where we compare them against the six hard or near-hard task types from the main benchmark under the same GPT-5 adaptive setting.

Table 3 summarizes the supplemental external evaluation set under the Exp2 setup. For each category, we report a pooled Pass@1 over all seven models and the best per-model Pass@1. Hole_Counting has zero successful solves over 140 attempts. Symbol_Count has 6 successes over 210 attempts, yielding an pooled Pass@1 of 2.9%, with the best per-model reaching 10.0%. Relation_Match has 17 successes over 210 attempts, yielding an aggregate pooled Pass@1 of 8.1%, with the best per-model reaching 16.7%. All three external categories remain well below the 40% threshold used in our main analysis. These results support the same core finding on externally sourced data: all three supplemental categories remain difficult for off-the-shelf MLLMs under optimized prompting, as their success rates remain well below the 40% threshold.

Adaptive session-memory evaluation. Table 4 groups together the six hard or near-hard task types from the main CaptchaWorld benchmark and the three supplemental external categories introduced above. We test a GPT-5 (Medium) attacker that retains within-round history and updates its strategy using only binary pass/fail feedback. Table 4 compares observed Adaptive-Success@3 with the Bernoulli-Success@3 estimated from the same GPT-5 (Medium) Pass@1 reference. This is not a direct comparison, since Bernoulli estimate assumes independent non-adaptive retries, whereas adaptive results come from a five-round session-memory experiment. On average across the nine tasks, Success@3 increases only slightly, from 21.0% under the Bernoulli to 24.4% under adaptive evaluation. The main upward deviations occur on near-zero static tasks, while structurally hard tasks still remain low, including Hole_Counting (0%), Symbol_Count (0%), and Dice_Count (20%). Thus, adaptive memory does not change the overall difficulty pattern: it can only help some boundary cases, but precise grounding, counting, and object-location binding remain much less reliable than simple recognition or selection tasks.

Table 4: GPT-5 (Medium) comparison of static and adaptive performance. The first six rows are hard or near-hard task types from the main CaptchaWorld benchmark, in the same order as Table 2. The final three rows are supplemental external categories introduced in Table 3.

Task	Pass@1	Success@3 (Bernoulli)	$\widehat{\text{Success}}$@3 (Adaptive)
Click_Order	10%	27.1%	20%
Dice_Count	0%	0.0%	20%
Patch_Select	30%	65.7%	40%
Pick_Area	20%	48.8%	40%
Place_Dot	15%	38.6%	20%
Rotation_Match	0%	0.0%	40%
Hole_Counting	0%	0.0%	0%
Relation_Match	0%	0.0%	40%
Symbol_Count	3%	8.7%	0%

Comparison with external solvers and benchmarks. Our study focuses on the native performance of off-the-shelf MLLMs in a controlled black-box setting, rather than on solver pipelines augmented with task-specific decomposition, fine-tuning, or additional engineering in prior works. We therefore compare our findings with recent CAPTCHA solver and benchmark studies [6, 9, 25, 27, 31] only as contextual baselines, not as direct head-to-head measurements. In COGNITION, the solver interface is intentionally restricted to direct off-the-shelf MLLM API calls on raw CAPTCHA instances, without task decomposition, family specific modules, or richer agent orchestration. In particular, systems such as Oedipus [6] do not merely change the prompt; they alter the solver architecture by decomposing a CAPTCHA into structured intermediate substeps before invoking the LLM, thereby evaluating engineered attack capability rather than the native capability of an off-the-shelf model. Halligan et al. [27], for example, study an end-to-end agentic attacker in both closed world and live field settings, so the reported success rates reflect a richer interactive attack pipeline rather than the native one-shot behavior of a fixed off-the-shelf API. Earlier Visual Reasoning CAPTCHA [9] are often built around fixed CAPTCHA families or vendor templates, with modular recognition components and task engineering tailored to those structures rather than to a mixed benchmark of raw CAPTCHA instances. Benchmark suites such as MCA-Bench and CAPTCHA-X [25, 31] broaden task coverage and are valuable for identifying which CAPTCHA families remain difficult, but they do not provide a matched protocol for measuring direct off-the-shelf API performance under a uniform black-box interface. Table 8 summarizes these differences and Appendix C provides a fuller methodological comparison. As a result, these comparisons support the external validity of our

main conclusion: recognition and simple selection tasks are increasingly easy for modern solvers, whereas tasks requiring precise grounding, counting, or relation binding remain more resistant.

6 Defense and Limitation Discussion

Based on our experimental evaluation, we identify visual CAPTCHA task types that remain challenging for state-of-the-art MLLMs, even under optimized prompting, retries, and few-shot examples. We then address RQ4: how can web service providers keep CAPTCHA schemes effective against increasingly capable MLLM-based solvers? To answer this question, we distill the key structural *factors* underlying the robust task types, translate them into a practical defense methodology, and provide a preliminary validation showing how this methodology can strengthen a vulnerable CAPTCHA task.

6.1 Hardness Factors from Robust Task Types

To understand why certain CAPTCHA task types remain resistant to MLLMs, we examine the structural properties. As shown in Section 5, while recognition-based tasks are often solvable, a smaller subset continues to resist automated solvers. From the observations, we distill three key factors underlying robustness, which expose a persistent gap between semantic reasoning and visual grounding in current models.

Fine-grained spatial localization: Across the five robust task types, the solver must localize targets on a continuous canvas rather than select from a few discrete options. The answer can be a specific point (Click_Order and Place_Dot) or a region defined by geometric or semantic cues (Pick_Area). Verification logic allows a tolerance window, but the model must still map language to precise coordinates. In practice, MLLMs often describe the correct region yet output points or areas that fall outside the acceptable zone.

Counting and aggregation beyond pattern recognition: Dice_Count and counting-based variants of Pick_Area couple visual recognition with explicit counting and light-weight arithmetic. Models frequently recognize local objects but miscount or mis-aggregate them, or make small arithmetic slips in the final answer. But, humans are reliable at counting small sets and performing simple sums, making these tasks structurally more robust than pure recognition or binary decisions.

Stability across models and prompting regimes: Click_Order, Place_Dot, Pick_Area, Dice_Count, and Patch_Select remain hard for all evaluated models under direct prompting and optimized prompts, and they stay hard even with task-specific few-shot demonstrations. In contrast, many recognition-style CAPTCHAs are highly sensitive to prompt wording and quickly become solvable under modest prompt engineering. Robust types thus appear constrained by deeper perceptual and reasoning limitations rather than by prompt-level misunderstandings, making them better candidates for long-lived deployment.

These hardness factors provide the basis for a practical hardening methodology: rather than relying on generic distortion, operators can redesign CAPTCHA tasks to selectively increase demands on grounding, counting, and multi-step consistency.

6.2 A Practical Defense Methodology

Drawing on the hardness factors above, we recommend a simple four-step defense methodology for practitioners. **First**, replace coarse recognition or discrete choice mechanics with tasks that require fine-grained visual grounding. **Second**, when localization alone is insufficient, further increase difficulty by coupling perception with counting, aggregation, or other lightweight multi-step reasoning. **Third**, when stronger protection is needed, compose multiple hardness factors within a single challenge so that the solver must complete several dependent subtasks correctly. **Fourth**, validate that the resulting task remains usable for legitimate users, monitor solver performance after deployment, and periodically refresh templates or instructions as model capabilities evolve. The four steps below elaborate this workflow.

First, favor fine-grained localization over discrete choice: Operators can replace discrete-choice mechanics with tasks that require precise coordinate prediction, thereby shifting the challenge from coarse recognition to fine-grained visual grounding. This transformation directly targets one of the most persistent weaknesses of current MLLM-based solvers. Unlike discrete selection, e.g., choosing a tile from a grid, continuous-space tasks require the user to pinpoint a specific location within a strict tolerance window. As evidenced by the robustness of Place_Dot and Click_Order, forcing the model to ground semantic instructions into precise pixel coordinates introduces a significantly higher error rate than standard multiple-choice formats.

Second, couple perception with counting and simple arithmetic: When localization alone is insufficient, operators can further increase difficulty by coupling perception with counting or lightweight aggregation, thereby requiring the solver not only to recognize visual elements but also to correctly quantify or combine them. Instead of asking which tiles contain a certain object, prefer tasks that require counting small objects or aggregating visual evidence into a numeric answer, as in Dice_Count or counting-based Pick_Area. Even simple arithmetic, e.g., summing small numbers, interacts poorly with current visual pipelines while being easy for humans, but counts should remain small to avoid excessive user burden.

Third, combine multiple hardness factors in a single challenge: Operators can compose multiple hardness factors within the same challenge so that the solver must complete several dependent sub-tasks correctly. Robustness increases when a single challenge incorporates multiple difficulty factors. A single challenge can require users to locate specific

features and verify a condition simultaneously. For example, interact with objects in a prescribed order, and optionally report an aggregate quantity. Such compositions forces the MLLM to perform a sequence of dependent sub-tasks consisting of attribute recognition followed by precise localization where any single error leads to verification failure.

Finally, preserve usability and plan for evolution: Throughout this process, operators should validate that the hardened task remains usable for humans, monitor solver performance after deployment, and periodically refresh templates or instructions as model capabilities evolve. Static templates eventually become vulnerable to fine-tuning. Operators can implement dynamic rotation strategies to maintain hardness. This includes regularly varying rendering styles to disrupt model overfitting and periodically rephrasing instructions, such as changing "*Select the largest region" to "Identify the area with the maximum surface*." This can prevent attackers from stabilizing their system prompts. Furthermore, operators should actively monitor solving metrics. A sudden spike in success rates or a convergence of solving times can indicate that an automated solver has bypassed the current scheme, necessitating an immediate template update.

Taken together, these four steps provide a practical defense methodology: increase grounding precision, add lightweight counting or aggregation when needed, compose multiple hardness factors for stronger protection, and iteratively validate the resulting design against both security and usability objectives. By integrating multiple hardness factors, such as asking a user to "*click on all visible eyes of the cat*," we combine counting with fine-grained localization. This composition forces the solver to execute a chain of dependent reasoning steps, where a single failure in logic or grounding invalidates the attempt. This significantly raises the technical and economic bar for attackers by rendering cheap, off-the-shelf MLLMs ineffective and necessitating the development of specialized, expensive solvers. In the following section, we provide a preliminary validation of these principles by hardening a representative vulnerable task.

6.3 Preliminary Validation for Defense

To illustrate how this methodology can be applied in practice, we instantiate it on the Select_Animal task type, which was classified as "Broken" in our baseline evaluation. In our baseline evaluation (Exp1 and Exp2), this task with state-of-the-art MLLMs achieved near-perfect Pass@1 rates (e.g., GPT-5 $> 95\%$). The original task involved selecting grid tiles containing a target animal, which MLLMs solved easily via coarse semantic recognition. Following the hardening methodology in Section 6.2, we transformed this task by replacing coarse grid selection with fine-grained localization and by introducing an implicit counting requirement.

- *Fine-grained spatial localization:* Instead of selecting a tile, the solver is required to "click on all *visible* eyes" of the target animal. This shifts the requirement from discrete classification to continuous-space coordinate prediction with a tight tolerance (15 px).
- *Implicit counting and existence detection:* The target animals are presented in varying poses (e.g., profile vs. frontal). The solver must visually ground the number of visible eyes (1 or 2) rather than relying on the prior knowledge that animals typically have two eyes.

Table 5: Defense validation on the hardened Select_Animal task. The proposed design changes reduced the success rate of SOTA models from near-perfect (Baseline) to near-zero (Hardened), effectively mitigating the threat.

Model	Baseline Pass@1	Hardened Pass@1
GPT-5 (Medium)	$> 95\%$	**0.0%**
GPT-5.1 (Medium)	$> 95\%$	**0.0%**
GPT-5.1 (None)	$> 95\%$	**0.0%**
Gemini 2.5 Pro	$> 90\%$	**0.0%**
Claude Sonnet 4.5	$> 90\%$	23.3%
Qwen3-VL	$> 80\%$	20.0%

We evaluated this hardened variant against the same suite of MLLMs using task-optimized prompts (Exp2). The results, summarized in Table 5, show a dramatic collapse in solver performance. In the following analysis, we dissect these results to verify the security gain (*Effectiveness*), confirm the defense mechanism through trace analysis (Failure Modes), and ensure that this added security does not come at the cost of user experience (*Usability Implications*).

Effectiveness: As shown in Table 5, the success rates for GPT-5, GPT-5.1 (both Medium and None), and Gemini 2.5 Pro dropped to **0.0%**. Notably, the failure of GPT-5.1 (None) confirms that the difficulty is structural and cannot be bypassed simply by reducing reasoning effort or relying on faster inference modes. Even the strongest performer on this new task, Claude Sonnet 4.5, achieved only 23.3%, falling well below the 40% threshold for a "Broken" task.

Failure Analysis: Analysis of the reasoning traces reveals that models frequently succumbed to two error modes predicted by our guidelines: (1) *Hallucination based on priors*, where models predicted the location of a second eye on a profile face where only one was visible, e.g., clicking empty space on the animal's cheek. (2) *Imprecise grounding*, where clicks landed on the face but missed the specific tolerance window of the eye.

Usability Implications: Crucially, this increase in security does not impose a proportional burden on human users. Identifying and pointing to visible eyes is a natural perceptual task for humans. While it requires slightly more precision than simple grid selection, it avoids the visual strain induced by adversarial noise, trading image degradation for a light-weight cognitive task that humans intuitively excel at. This validation

confirms that by structurally aligning CAPTCHA mechanics with MLLM weaknesses—specifically imprecise localization and hallucination—we can restore security guarantees without degrading the human user experience.

While demonstrated on Select_Animal, these results validate the broader efficacy of the structural guidelines proposed in Section 6.2. The complete failure (**0.0%**) of both GPT and Gemini series, combined with the suppressed performance of Claude Sonnet 4.5 (23.3%) and Qwen3-VL (20.0%), confirms that the defense is robust across diverse model families. This cross-model collapse indicates that coupling fine-grained localization with implicit counting successfully exploits the shared neuro-symbolic gap in current MLLMs. By retrofitting such primitives onto existing tasks, operators can force attackers to abandon scalable, general-purpose APIs in favor of costly, specialized solutions, fundamentally altering the economic asymmetry of the attack.

6.4 Limitations

Although *Open CaptchaWorld* provides a broad and useful benchmark for evaluating the security of visual CAPTCHA systems against MLLMs, it does not fully capture the long tail of edge-case, highly customized, or domain-specific CAPTCHA implementations that may appear in practice. Our findings should therefore be interpreted as evidence on a representative benchmark of modern visual CAPTCHA designs rather than as a comprehensive evaluation of all real-world deployments; they also reflect the current capability landscape of MLLMs, which may shift as models continue to improve, and do not exhaust the space of stronger custom or agentic attackers. In particular, the current benchmark does not yet include direct comparison against engineered solvers such as Oedipus or Halligan, which limits the scope of our empirical comparison to off-the-shelf MLLM settings; we plan to extend the benchmark in future work to support such comparisons. *Open CaptchaWorld* combines curated prior sources with our unified benchmark construction and evaluation pipeline, but the resulting benchmark remains limited in scale. The adaptive session-memory experiment should be interpreted as a stress test rather than a deployment-wide probability estimate, and both model capability and attack cost may shift over time. Notably, while the dataset is sufficient to reveal clear empirical trends, the quantitative results should be interpreted within the scope of this benchmark and the evaluated model/pricing snapshot, rather than as precise universal estimates of attack success rates or costs for every CAPTCHA implementation in the wild.

7 Conclusion

This paper examines how modern MLLMs challenge the security of visual CAPTCHAs, evaluating seven representative models across 21 real-world task types under a realistic black-box threat model that accounts for retries, latency, and cost. Our results reveal a sharp hardness gap: most recognition oriented CAPTCHAs are now reliably solvable, often at a low cost and within a few retries, while a compact set of tasks requiring precise localization, cross-panel consistency and counting remains consistently robust even under optimized prompting, few-shot demonstrations or adaptive session-memory. Analysis of model reasoning traces shows that failures on these hard tasks stem from persistent weaknesses in spatial grounding and object–position binding, which informs a set of defense guidelines that emphasize continuous space localization, perception combined with counting, and the inclusion of multiple difficulty factors within a single challenge. We validate these strategies by hardening a representative vulnerable task, demonstrating that structural defenses can reduce MLLM success rates from near-perfect levels (>95%) to zero. Crucially, compared to traditional defenses that rely on aggressive image blurring and cause visual fatigue, our redesign maintains high visual clarity. While it introduces a slight shift towards fine-grained perceptual interaction, it leverages natural human capabilities rather than frustrating users with degraded images.

Ethical Considerations

This research studies the security posture of visual CAPTCHA systems against emerging MLLMs. While our work demonstrates methods to bypass existing security mechanisms, we have taken strict measures to ensure our methodology adheres to ethical research principles and minimizes potential harm.

Interaction with Online Services: A primary ethical concern in CAPTCHA research is the potential impact on the availability and integrity of live web services. To mitigate this risk, our evaluation was primarily conducted on the *Open CaptchaWorld* dataset, which aggregates historical and synthetic CAPTCHA instances. We did not perform high-volume, automated attacks against production registration flows or protected endpoints of active web services. The black-box attack setting described in our methodology simulates an attacker's perspective using offline instances, ensuring no disruption to real-world server infrastructure or degradation of user experience for legitimate users.

Usage of MLLM APIs: Our experiments utilized commercial APIs (e.g., OpenAI, Google, Anthropic). We adhered to the standard usage tiers and rate limits provided by these services. While automated interaction with web interfaces is often restricted, our research focuses on the safety evaluation of the models' reasoning capabilities rather than exploiting the service providers themselves. All API usage was paid and authenticated, ensuring fair compensation for the computational resources consumed.

Human Subjects: This study implies comparisons with human performance (e.g., human-like cost and latency). We explicitly state that no new human subject experiments were con-

ducted for this paper. All baselines regarding human solving accuracy, speed, and cost were derived from established prior literature and publicly available market data from CAPTCHA-solving services. Therefore, Institutional Review Board (IRB) approval was not required.

Stakeholder Impacts. Beyond direct research risks, our results have implications for several stakeholders. Legitimate users may benefit from stronger protection against automated abuse, but harder CAPTCHA designs can also increase completion time, error rates, and accessibility burden. In particular, increasing task granularity to avoid MLLM solutions should not be interpreted as a recommendation to make challenges arbitrarily more complex. Deployments should preserve visual clarity, keep counting and interaction sequences small, provide accessible alternatives, and measure human completion time and failure rates before rollout. Researchers may benefit from reproducible evaluation artifacts for follow-up studies, but such artifacts also create dual-use risk; accordingly, our release focuses on offline evaluation and omits production-oriented automation or service-specific bypass code. Companies operating CAPTCHA-protected services can use our findings to guide redesign and monitoring, but should treat the proposed hardening strategies as a security–usability trade-off that requires staged deployment and periodic reassessment.

Responsible Disclosure: Consistent with minimizing harm, we initiated disclosure to affected stakeholders upon identifying clear weaknesses and shared non-weaponized reproduction materials for verification. Specifically, we disclosed the findings to Google, OpenAI, Anthropic, hCaptcha, Cloudflare, and Alibaba. For Google, we submitted an AI VRP report, which had been reviewed and closed by Jan. 26, 2026; Google indicated that the submission was considered out of scope rather than a technical vulnerability in Google's infrastructure. For OpenAI, we sent a disclosure message in Jan. 29, 2026 summarizing the study setting, the main findings, and high-level mitigation directions, and OpenAI replied with guidance on the appropriate reporting route. For Anthropic, we sent a similar disclosure in Jan. 28, 2026, and Anthropic acknowledged receipt. For hCaptcha, we sent a disclosure in early February 2026, and hCaptcha replied that the report had been forwarded to the relevant internal team. For Cloudflare, we sent a disclosure message in Jan. 29, 2026 describing the findings, the offline evaluation setting, and possible mitigations, and we received a routing or intake response. For Alibaba, we likewise sent a disclosure message in Jan. 30, 2026 with the same scope and level of detail, and we received a routing or intake response. Accordingly, we believe that publication is justified because these findings help defenders better understand the current limitations of visual CAPTCHAs and support the development of more robust defenses.

Dual-Use and Code Release: To balance reproducibility with safety, our released code serves as a research framework for evaluating model capabilities, lacking the engineering optimizations required for weaponized, large-scale bot operations. We believe that the defensive insights specifically the design guidelines for MLLM-resistant CAPTCHAs outweigh the risks. While the MLLM capabilities we evaluate are already available to the public, documenting their impact on CAPTCHAs is essential for the security community to develop countermeasures.

Open Science

The public artifact necessary to evaluate the contributions of this paper is permanently available through Zenodo at https://doi.org/10.5281/zenodo.20406852. The released package includes the offline CAPTCHA/MLLM evaluation framework, the cleaned benchmark data and supplemental external categories used in the paper, recorded results and outputs for reproducing figures and tables, and supporting reproducibility infrastructure such as tests, dependency files, and artifact-building utilities. Consistent with our open science and safety commitments, the release is limited to evaluation and verification artifacts. It excludes local secrets, provider credentials, live-service automation, and production-oriented CAPTCHA bypass infrastructure; these excluded materials are not required to evaluate the paper's claims.

References

[1] 2Captcha. Image captcha solver — online image captcha solving service, 2025. Accessed: 2025-11-20. URL: https://2captcha.com/p/image-picture-captcha-solver.

[2] Ismail Akrout, Amal Feriani, and Mohamed Akrout. Hacking google recaptcha v3 using reinforcement learning. 2019. arXiv:1903.01003.

[3] Elie Bursztein, Steven Bethard, Celine Fabry, John C Mitchell, and Dan Jurafsky. How good are humans at solving captchas? a large scale evaluation. In *2010 IEEE symposium on security and privacy (SP)*, pages 399–413. IEEE, 2010.

[4] CaptchaCoder. Hybrid captcha solving service: Api & human/ocr based service, 2025. Accessed: 2025-11-20. URL: https://captchacoder.com/.

[5] DeCaptcher. Captcha decoding, 2025. Accessed: 2025-11-20. URL: https://www.decaptcher.com/.

[6] Gelei Deng, Haoran Ou, Yi Liu, Jie Zhang, Tianwei Zhang, and Yang Liu. Oedipus: Llm-enchanced reasoning captcha solver. 2025.

[7] Elie Dessant. Buster: bypass captcha by filling fake audio challenges. https://github.com/dessant/buster, 2020. GitHub repository.

[8] Ziqi Ding, Gelei Deng, Yi Liu, Junchen Ding, Jieshan Chen, Yulei Sui, and Yuekang Li. Illusioncaptcha: A captcha based on visual illusion. In *Proceedings of the ACM on Web Conference 2025 (WWW)*, pages 3683–3691, 2025.

[9] Yipeng Gao, Haichang Gao, Sainan luo, Yang Zi, Shudong Zhang, Wenjie Mao, Ping Wang, Yulong Shen, and Jeff Yan. Research on the security of visual reasoning CAPTCHA. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3291–3308. USENIX Association, August 2021.

[10] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. Browser fingerprinting: A survey. *ACM Transactions on the Web (TWEB)*, 14(2):1–33, 2020.

[11] Jingmeng Li, Lukang Fu, Surun Yang, and Hui Wei. Micaptcha: Enhance the security of captcha using mooney images. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 39, pages 1383–1391, 2025.

[12] Jiacheng Liu, Yaxin Luo, Jiacheng Cui, Xinyi Shang, Xiaohan Zhao, and Zhiqiang Shen. Next-gen captchas: Leveraging the cognitive gap for scalable and diverse gui-agent defense. *arXiv preprint arXiv:2602.09012*, 2026.

[13] BuiltWith Pty Ltd. Websites using recaptcha, 2025. Accessed: 2025-11-20. URL: https://trends.builtwith.com/websitelist/reCAPTCHA.

[14] Yaxin Luo, Zhaoyi Li, Jiacheng Liu, Jiacheng Cui, Xiaohan Zhao, and Zhiqiang Shen. Open captchaworld: A comprehensive web-based platform for testing and benchmarking multimodal llm agents. *arXiv preprint arXiv:2505.24878*, 2025.

[15] Marti Motoyama, Kirill Levchenko, Chris Kanich, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. Re:{CAPTCHAs—Understanding}{CAPTCHA-Solving} services in an economic context. In *19th USENIX Security Symposium (USENIX Security 10)*, 2010.

[16] Hoang Dai Nguyen, Karthika Subramani, Bhupendra Acharya, Roberto Perdisci, and Phani Vadrevu. C-frame: Characterizing and measuring in-the-wild captcha attacks. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 277–295, 2024.

[17] NopeCHA LLC. Nopecha api documentation, 2025. Accessed: 2025-05-23. URL: https://developers.nopecha.com/.

[18] Behzad Ousat, Esteban Schafir, Duc C. Hoang, Mohammad Ali Tofighi, Cuong V. Nguyen, Sajjad Arshad, Selcuk Uluagac, and Amin Kharraz. An analysis of a brittle security feature on the modern web. In *Proceedings of the ACM Web Conference 2024 (WWW)*, WWW '24, page 1835–1846, 2024.

[19] Andreas Plesner, Tobias Vontobel, and Roger Wattenhofer. Breaking recaptchav2. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, page 1047–1056. IEEE, July 2024.

[20] Minfeng Qi, Dongyang He, Qin Wang, and Lefeng Zhang. Viper strike: Defeating visual reasoning captchas via structured vision-language inference. *arXiv preprint arXiv:2601.06461*, 2026. Accepted by USENIX Security 2026.

[21] Andrew Searles, Yoshimichi Nakatsuka, Ercan Ozturk, Andrew Paverd, Gene Tsudik, and Ai Enkoji. An empirical study & evaluation of modern CAPTCHAs. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3081–3097. USENIX Association, August 2023.

[22] Chenghui Shi, Shouling Ji, Qianjun Liu, Changchang Liu, Yuefeng Chen, Yuan He, Zhe Liu, Raheem Beyah, and Ting Wang. Text captcha is dead? a large scale deployment and empirical study. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 1391–1406. Association for Computing Machinery, 2020.

[23] Suphannee Sivakorn, Iasonas Polakis, and Angelos D. Keromytis. I am robot: (deep) learning to break semantic image captchas. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 388–403, 2016.

[24] Suphannee Sivakorn, Jason Polakis, and Angelos D Keromytis. I'm not a human: Breaking the google recaptcha. volume 14, pages 1–12, 2016.

[25] Python Song, Luke Tenyi Chang, Yun-Yun Tsai, Penghui Li, and Junfeng Yang. Reasoning under vision: Understanding visual-spatial cognition in vision-language models for captcha. *arXiv preprint arXiv:2510.06067*, 2025.

[26] Captcha.eu Team. What is a captcha farm?, 2025. Accessed: 2025-11-20. URL: https://www.captcha.eu/what-is-captcha-farm/.

[27] Xiwen Teoh, Yun Lin, Siqi Li, Ruofan Liu, Avi Sollomoni, Yaniv Harel, and Jin Song Dong. Are {CAPTCHAs} still bot-hard? generalized visual {CAPTCHA} solving with agentic vision language

model. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 3747–3766, 2025.

[28] Theyka. Turnstile-solver: Github repository for cloudflare turnstile bypass scripts, 2025. Accessed: 2025-05-23. URL: https://github.com/Theyka/Turnstile-Solver.

[29] Sheng Tian and Tao Xiong. A generic solver combining unsupervised learning and representation learning for breaking text-based captchas. In *Proceedings of The Web Conference 2020*, WWW '20, page 860–871. Association for Computing Machinery, 2020.

[30] Ilias Tsingenopoulos, Davy Preuveneers, Lieven Desmet, and Wouter Joosen. Captcha me if you can: Imitation games with reinforcement learning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 719–735, 2022.

[31] Zonglin Wu, Yule Xue, Yaoyao Feng, Xiaolong Wang, and Yiren Song. Mca-bench: A multimodal benchmark for evaluating captcha robustness against vlm-based attacks. *arXiv preprint arXiv:2506.05982*, 2025.

[32] Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. Yet another text captcha solver: A generative adversarial network based approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 332–348. Association for Computing Machinery, 2018.

[33] Jiaming Zhang, Jitao Sang, Kaiyuan Xu, Shangxi Wu, Xian Zhao, Yanfeng Sun, Yongli Hu, and Jian Yu. Robust captchas towards malicious ocr. *IEEE Transactions on Multimedia*, 23:2575–2587, 2020.

[34] Ruijie Zhao, Xianwen Deng, Yanhao Wang, Zhicong Yan, Zhengguang Han, Libo Chen, Zhi Xue, and Yijun Wang. Geesolver: A generic, efficient, and effortless solver with self-supervised learning for breaking text captchas. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1649–1666, 2023.

A Model API Pricing

Table 6 summarizes the token-based prices used in our cost analysis. All values are in USD per 1,000 tokens and correspond to the public pricing at the time we ran our experiments. We only account for prompt and completion tokens reported by the providers; internal "reasoning" or hidden tokens (when applicable) are not observable and are therefore excluded from our estimates.

B Dataset Description

Our benchmark contains 458 visual CAPTCHA instances across 21 task types (see Table 7). Each instance consists of one or more images (PNG/JPG) and a structured `ground_truth.json` entry with an English prompt, a short description, and machine-interpretable labels (click coordinates, grid indices, option indices, numeric counts, or rotation angles). Click-based tasks use pixel coordinates with a small tolerance radius; grid-based tasks use zero-based row-major indices; multi-answer tasks are stored as sets or sequences. In addition to cleaned CaptchaWorld benchmark, we construct a supplemental external evaluation set for further validation. These instances follow the same basic schema and standardization principles as the main benchmark, including metadata alignment, provenance and license checks, and answer-format normalization, but are kept separate from the benchmark.

C Comparison with External Studies

Table 8 provides contextual comparison between COGNITION and recent CAPTCHA solvers and benchmarks. Beyond the high-level performance trends summarized in the main text, these studies also align with several of our task-level observations. Halligan reports that continuous actions and 3D spatial reasoning remain weaker than discrete selection and swap-style tasks [27], which is consistent with our failures on *Place_Dot* and *Click_Order*, where models often describe the correct semantic target but fail to output verifier-valid coordinates. Oedipus and earlier specialized visual-reasoning solvers further suggest that semantic difficulty alone is not a reliable defense once a CAPTCHA can be decomposed into stable, task-specific subproblems [6, 9]. This supports our design emphasis on combining semantic understanding with precise grounding, object-location binding, counting, and verifier-sensitive output constraints, rather than relying only on high-level reasoning complexity. Finally, larger benchmarks such as MCA-Bench and CAPTCHA-X show that interaction depth, exact spatial grounding, and multi-step structure continue to separate easier and harder CAPTCHA types [25, 31], reinforcing the broader applicability of the structural factors identified in our benchmark.

Table 6: Token-based model pricing (USD per 1,000 tokens).

Model (API name)	Input	Output
gpt-5	0.00125	0.01000
gpt-5.1	0.00125	0.01000
claude-sonnet-4-5	0.00300	0.01500
gemini-2.5-pro	0.00125	0.01000
gemini-2.5-flash	0.00030	0.00250
qwen3-vl-235b-instruct	0.00022	0.00088

Table 7: Task types and brief descriptions in our dataset, including supplemental external evaluation categories.

Task type	# inst.	Brief description
Bingo	25	Swap two cells in a 3×3 emoji grid to complete a line of identical symbols.
Click_Order	10	Click a set of icons in the same order as shown in a separate reference image.
Connect_icon	20	Choose which arrow connection matches the dotted-line pattern in the reference diagram.
Coordinates	18	Select the option where a character sits at the seat indicated in the reference image.
Dart_Count	20	Choose the dartboard whose darts add up to a given target number.
Dice_Count	11	Read a composite dice scene and output the total number of visible pips.
Geometry_Click	15	Click the specified geometric element (e.g., a letter relative to a shape).
Image_Matching	19	Match an object in the left image to the correct choice on the right.
Image_Recognition	20	In a 3×3 grid, select all images containing a specified object class.
Misleading_Click	10	Click to continue while avoiding prominently highlighted "do not click" regions.
Object_Match	20	Adjust a discrete object count until it matches the reference image.
Patch_Select	10	In a 5×5 patch grid, select all patches containing a target object.
Path_Finder	10	Choose the path that moves an object to a cross-marked destination.
Pick_Area	30	Click on the largest outlined region in a complex scene.
Place_Dot	32	Place a dot at the end of a car's path along a drawn trajectory.
Rotation_Match	48	Rotate an object so that its orientation matches a reference direction.
Select_Animal	30	In a small grid, select all cells containing the target animal (e.g., fox).
Unusual_Detection	30	Select images where an animal has a mismatched head and body.
Symbol_Count [12]	30	Count specified symbols in an occluded repeating grid.
Relation_Match [20]	30	Choose the object crop matching a relational prompt in a 3D scene.
Hole_Counting [12]	20	Select grid tiles whose shapes contain the requested number of holes.

Table 8: Contextual comparison with state-of-the-art CAPTCHA solvers and larger benchmarks. These are not direct head-to-head comparisons to COGNITION because several prior systems evaluate engineered solver pipelines, rather than the native capability of off-the-shelf MLLM APIs in our black-box setting. For example, Oedipus decomposes CAPTCHA solving into structured intermediate substeps before invoking the LLM, whereas COGNITION measures direct off-the-shelf model performance.

System / study	**Solver type**	**Reported setting**	**Reported result**
This Work	Direct Off-the-shelf MLLM APIs	18-task benchmark plus provenance-checked external set; static/adaptive analyses	Best $\sim$60% Pass@1; external categories $<$17%; adaptive Success@3 $<$40%
Halligan [27]	Agentic VLM solver	2,600 closed-world challenges; 3,000 live field tasks	60.7% closed-world solve rate; 70.6% field solve rate
Oedipus [6]	DSL-guided decomposition pipeline over an LLM	Reasoning CAPTCHA benchmark with decomposed substeps	63.5% reported average success rate
Earlier Visual Reasoning CAPTCHA [9]	Task-specific modular pipelines	Fixed visual-reasoning or vendor-specific CAPTCHA families	High success on target families; limited mixed-benchmark coverage
MCA-Bench / CAPTCHA-X [25, 31]	Benchmark suites and agentic VLM attackers	Multimodal CAPTCHA and high-difficulty spatial CAPTCHA benchmarks	Simple visual tasks highly solvable; spatial and multi-step tasks harder