THE GRANULAR GAMBIT: CHECKMATING BEHAVIOR-BASED CAPTCHAS

Anonymous authors
Paper under double-blind review

ABSTRACT

Behavior-based CAPTCHAs such as Google reCAPTCHA v3 enable frictionless verification by assigning risk scores from fine-grained user interaction traces. Prior studies, however, largely rely on coarse-grained analyses and fail to capture the atomic behavioral signals underlying v3's scoring mechanism. This work examines whether agentic AI can synthesize human-like low-level interactions to obtain high trust scores and bypass reCAPTCHA v3. The problem is formulated as sequential decision-making over atomic behaviors, and Seqion, an LLM-driven agentic framework for fine-grained web interaction execution, is introduced. Experiments on a controlled reCAPTCHA v3 testbed, including registration, forum posting, and content reading tasks, show that agentic AI can consistently increase trust scores and achieve up to a 90% bypass success rate, even without domain-specific knowledge. Furthermore, evaluations on an external, real-world website reveal that Seqion achieves a mean trust score of 0.60, significantly outperforming the state-of-the-art baseline of 0.34 and closely approximating the genuine human average of 0.69. These results expose fundamental limitations of frictionless CAPTCHA systems under rapidly advancing agentic AI.

1 INTRODUCTION

Distinguishing humans from machines is a cornerstone of online security. The pervasive automated scripts promote the unprecedented threats, including mass account registration, scalable data scraping. To mitigate these automated abuses, CAPTCHAs Xu et al. (2020) have been widely adopted as the primary line of defense. Generally, CAPTCHAs operate as challenge-response mechanisms, which present cognitive or perceptual tasks that are intuitively solvable by human but intractable for machine. By erecting this barrier, CAPTCHAs serve an indispensable role in preserving the overall integrity of web ecosystems.

CAPTCHAs have evolved from text-based puzzles Google (2025a); Bursztein et al. (2011) to image-based challenges Google (2025b); Ding et al. (2025), as shown in Fig. 1. However, advancements in computer vision have rendered these barriers increasingly vulnerable to automated solvers Ye et al. (2018); Teoh et al. (2025). Furthermore, such explicit tests impose cognitive friction and task interruption on legitimate users. Thus, behavior-based CAPTCHAs like reCAPTCHA v3 Google (2025c) have been widely adopted Built with (2025). They replace one-shot, intrusive puzzles with implicit, continuous interaction analysis, assuming solvers cannot synthesize human-like behaviors.

The rapid progress of modern AI, especially LLM-powered agentic systems, makes it increasingly plausible to attack behavior-based CAPTCHAs such as reCAPTCHA v3. Unlike traditional automation scripts that operate at a coarse task level, agentic AI can interpret web semantics, plan multi-step procedures, and execute fine-grained interactions, potentially matching the low-level behavioral signals used for risk scoring. This raises a key question: can agentic AI consistently generate human-like atomic behaviors to obtain high trust scores and bypass reCAPTCHA v3 in practice? To answer this question, we build Seqion, an LLM-driven automation framework that navigates websites in an agentic manner. We formulate reCAPTCHA v3 evasion as a sequential decision process over atomic behavioral units and design agents that plan and execute web interactions with fine temporal and spatial control. We evaluate our approach on a controlled testbed website protected by reCAPTCHA v3, covering representative tasks including registration, forum replying, and blog reading. Our results show that agentic AI can significantly improve trust scores and, in certain settings, achieve up to a

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

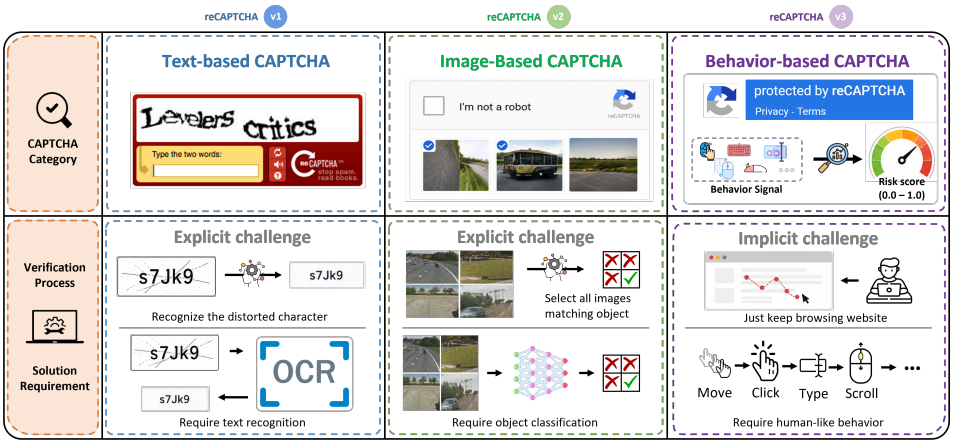


Figure 1: Google reCAPTCHAs of text-based v1, image-based v2 and behavior-based v3

90% bypass success rate, even without specialized domain knowledge. Furthermore, evaluations on an external, real-world website reveal that Seqion achieves a mean trust score of 0.60, significantly outperforming the baseline of 0.34 and closely approximating the genuine human average of 0.69.

The contributions of this work are as follows: ❶ This work provides the first systematic study of how low-level interactions influence behavior-based verification in reCAPTCHA v3 under agentic AI. ❷ Behavior-based CAPTCHA solving is formulated as a sequential decision-making and planning problem, in which atomic behaviors are selected dynamically. ❸ An agentic AI framework is introduced that integrates multiple specialized agents, each responsible for a distinct function analogous to human web-browsing behavior. ❹ Extensive evaluations are conducted on a controlled website across tasks, including registration, forum posting, and content reading, assessing agent capabilities under dynamic behavioral conditions.

2 Related Work

CAPTCHAs. Over two decades, CAPTCHAs Guerar et al. (2021) have progressed through three distinct generations: text- Google (2025a); Bursztein et al. (2011); Shi et al. (2020), image- Google (2025b); hcaptcha (2025), and behavior-based CAPTCHAs Google (2025c), as further detailed in Appendix B. Earliest text-based variants Google (2025a) can be easily solved Ye et al. (2018); Ishkov & Terekhov (2022); Nian et al. (2022). To address the vulnerabilities, image-based variants Google (2025b); hcaptcha (2025); Nejati et al. (2014) were introduced. However, such interruptive challenges Shi et al. (2021); Zhang et al. (2020) disrupted user experience, prompting the pursuit of frictionless alternatives. The third generation, exemplified by reCAPTCHA v3 Google (2025c), eliminated explicit puzzles, moving towards continuous, invisible risk assessment for returning a risk score $\in \{0.1, 0.3, 0.7, 0.9\}$ to the backend. Very recently, researches explored the novel CAPTCHA paradigms, including reasoning-based Wang et al. (2018), audio-based Wang et al. (2023), illusion-based CAPTCHAs Ding et al. (2025), which render AI-hard while remaining accessible for human.

CAPTCHA Solvers. The focus of CAPTCHA research shifted with the advent of AI, moving from classical algorithms Mori & Malik (2003) to machine learning-based solvers for text- Bursztein et al. (2014); Gao et al. (2016); Ye et al. (2018); Tang et al. (2018) and image-based CAPTCHAs Hossen & Hei (2021); Hossen et al. (2020); Sivakorn et al. (2016a;b); Plesner et al. (2024). A more recent foundation Teoh et al. (2025); Aissi et al. (2025) was established by tool-augmented LLMs, which enable browser control through natural language. Additionally, studies also examined audio-based CAPTCHAs and their vulnerabilities Abdullah et al. (2022); Bock et al. (2017); Fanelle et al. (2020). While behavior-based studies Akrout et al. (2019); Tsingenopoulos et al. (2022) are the most closely related to our work, they treat interaction patterns as a black box. We fundamentally advance this domain by deconstructing the detection logic to pinpoint exactly how individual versus sequential actions dictate defense outcomes.

3 METHODOLOGY

3.1 THREAT MODEL

Attacker (CAPTCHA Solver). We model the attacker as an external adversary with no privileged access to the target service's infrastructure, verification logic, or private training data. The goal is to automate the solution of behavior-based CAPTCHA challenges at a success rate sufficient to undermine their security purpose, effectively bypassing a mechanism designed for human-exclusive verification. The attacker's interaction is strictly black-box: they may observe a rating feedback upon submission but cannot inspect the intermediate process of verification.

Defender (CAPTCHA Service). We model the defender as the service operator who deploys behavior-based CAPTCHA to safeguard against automated abuse. The goal is to prevent automated solvers from achieving human-level success rates without manual effort.

3.2 PROBLEM STATEMENT

To formalize the evasion objective, we define the probability measures over the product trace space $\mathcal{T} = \mathcal{T}_{sem} \times \mathcal{T}_{mot}$, where $\mathcal{T}_{sem}$ is the discrete space of semantic plans and $\mathcal{T}_{mot}$ is the continuous space of kinematic trajectories.

Definition 1 (Induced Distributions)**.** *Let ψ be the task specification. We define the conditional probability distributions over traces $\tau = (\tau_{sem}, \tau_{mot}) \in \mathcal{T}$ as follows:*

Human Prior (P_{Human}): *The ground-truth distribution of legitimate traces given the task:*

$$P_{Human}(\tau \mid \psi) = \Pr(\tau \mid \textit{User is Human}, \textit{Task} = \psi) \tag{1}$$

Agent Distribution (P_{Seqion}): *The distribution induced by the agent's factorized policy $\pi_{seqion} = (\pi_{plan}, \pi_{act})$, computed via the chain rule:*

$$P_{Seqion}(\tau \mid \psi) = \underbrace{\pi_{plan}(\tau_{sem} \mid \psi)}_{\textit{Semantic (Discrete)}} \cdot \underbrace{\pi_{act}(\tau_{mot} \mid \tau_{sem})}_{\textit{Motor (Continuous)}} \tag{2}$$

Definition 2 (δ-Indistinguishability)**.** *An agent is δ-indistinguishable from humans on task ψ if the Kullback-Leibler divergence satisfies:*

$$D_{KL}(P_{Seqion} \| P_{Human}) \triangleq \mathbb{E}_{\tau \sim P_{Seqion}} \left[\log \frac{P_{Seqion}(\tau \mid \psi)}{P_{Human}(\tau \mid \psi)} \right] < \delta, \tag{3}$$

where the expectation $\mathbb{E}$ is taken over the hybrid space $\mathcal{T}$, and δ is the verifier's detection threshold.

Hypothesis 1 (The Decoupling Hypothesis)**.** *While the true human distribution $P_{Human}(\tau \mid \psi)$ exhibits complex entanglement between cognition and motor control, we hypothesize that it can be effectively **approximated** by a modular factorization:*

$$P_{Human}(\tau \mid \psi) \approx P(\tau_{sem} \mid \psi) \cdot \tilde{P}(\tau_{mot} \mid \tau_{sem}) \tag{4}$$

Crucially, we posit that the conditional motor component $\tilde{P}$ follows universal low-dimensional kinematic laws (e.g., Fitts's Law). Thus, optimizing π_{act} via heuristic biological noise (without holistic AGI) is sufficient to minimize the divergence below δ.

3.3 APPROACH OVERVIEW

Fig. 2 provides an overview of Seqion, a system designed to bypass reCAPTCHA v3 by orchestrating the interaction between agents and browser through webpage parsing and external tools.

Setp 1: Website Extraction. We first design a parsing agent to identify and extract web elements and description from website. The extracted data serves as the foundational input for subsequent behavioral simulations and a deeper understanding of the website's functional architecture.

Setp 2: Strategy Planning. Next, the system analyzes the previous elements and description, classifying elements into different types, such as target and interactable elements. To this end, it infers relationships between elements and formulates a step-by-step strategy aligned with task.

Setp 3: CAPTCHA Solving. Then, we deploy two specialized agents to complete the task: ❶ *Tool Agent* generates and advances tools based on the proposed strategy before and ❷ *Running Agent* runs the generated tools on designated elements and during transitions between them.

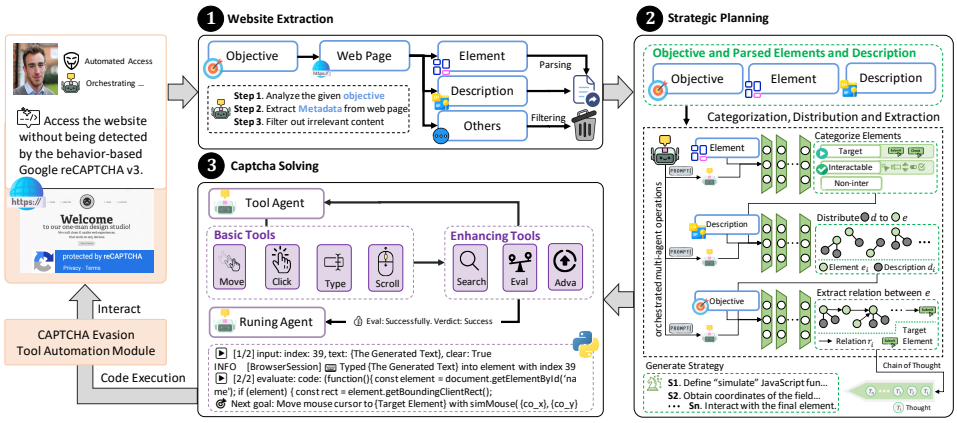


Figure 2: Overview of Seqion, consisting of three steps: ❶ Website Extraction parses the element and description from the website; ❷ Strategy Planning categorizes the interactable elements and orchestrates the plan according to the task; ❸ Captcha Solving generates the executable code and implements the strategy.

3.4 Website Extraction

Given a prompt, the parsing agent launches a web instance and navigates to the target website. It takes as input a page $\mathcal{P}$ including elements $\mathcal{E} = \{e_1, e_2, \cdots, e_n\}$, description $\mathcal{D} = \{d_1, d_2, \cdots, d_n\}$ and other stuff. After that, the DOM is analyzed to filter out non-functional content such as ads and decorative images. The agent then constructs a standardized set $\mathcal{E}$ and captures descriptive text, label attributes as $\mathcal{D}$. Note that an element e_i may map to multiple descriptions (e.g. on a signup page, the password field can be associated with both the label "Password" and the placeholder "The password includes 8 characters"), which we denote as $e_i \lhd \{d_j, d_k\}$. Finally, the parsing agent utilizes chain-of-thought to output sequentially: $\mathcal{E}$, $\mathcal{D}$, and task ψ.

3.5 Strategic Planning

Given element set $\mathcal{E}$, description set $\mathcal{D}$ and task ψ, the step is to interact with elements $e \in \mathcal{E}$ accroding to description $d \in \mathcal{D}$ and relation $r \in \mathcal{R}$, so as to accomplish ψ. We let the agent parse the elements and descriptions, extract relations, and generate a step-by-step strategy to fulfill ψ.

Table 1: Classification of Interactable Element.

Category	Interactable	Description
Target	Submit	Can be clicked to submit for completing task
Interactable	Clickable	Can be clicked for further interaction
	Inputtable	Can be input to enter text data
	Scrollable	Can be scrolled to access other elements
	Selectable	Can be selected toggled as an answer choice
Non-Interactable	-	Elements such as images, irrelevant texts

Elements. We define an element categorization tool $t_e(\mathcal{E}) \rightarrow \{\mathcal{E}_{type_1}, \mathcal{E}_{type_2}, \cdots, \mathcal{E}_{type_n}\}$ takes the set of elements $\mathcal{E}$ as input and categorizes different types of elements, which is shown in Table 1. To identify web element types (target, interactable and non-interactable), we perform element annotation. This is achieved through two executable Python blocks: get_element(description: str) for retrieving elements and set_element_as(type: str) for annotating them. The final output is a JSON block of a sequence of elements including #id, #type and other attributes.

Descriptions. We define a description distribution tool $t_d(\mathcal{D}) \rightarrow \{(e_1, d_1), (e_2, d_2, d_3), \cdots, (e_n, d_{n'})\}$ takes the set of description $\mathcal{D}$ as input and distribute each description to its corresponding elements by computing cosine similarity. This process is implemented using two Python blocks: link_element(description: str) for linking description and element, and instruct_element(instruction: str) for appending actionable instructions. The final output is an extended JSON block of elements, each enriched with matched descriptions and interaction instructions.

Relations. We define a relation extraction tool $t_r(\mathcal{E}, \mathcal{D}, \psi) \rightarrow \{r_1, r_2, \cdots, r_n\}$ takes categorized elements set $\mathcal{E}$, corresponding $\mathcal{D}$ and task ψ, and generates relation set $\mathcal{R}$. We conduct a topological

Table 2: Browser Interaction Tools.

Function	Description
click(x)	Click the element x
move(x,y)	Move the mouse from element x to element y
type(x,text)	Enter text on the element x
scroll(q)	Scroll the page for q px
search(t)	Search how to enhance the tool t
evaluate(t)	Evaluate the tool t in simulating human
advance(t)	Advance the tool t in simulating human
move(x, y, t)	Move the mouse from element x to element y in a straight-line within t seconds
move(x, y, t)$_b$	Move the mouse from element x to element y in a Bezier curve within t seconds
type(x,text,d)	Enter text on the element x for a fixed delay d between two words
type(x,text,d)$_r$	Enter text on the element x for a random delay d between two words
scroll(q, t)	Scroll the page for q px within t seconds

sorting of elements according to ψ to classify the elements used in prerequisite, subsequent and optional operations. Note that the target element should be interacted with at last based on ψ.

Strategy. We define a strategy plan tool $t_s(\mathcal{E}, \mathcal{R}, \psi) \rightarrow \{S_1, S_2, \cdots, S_n\}$ takes the set of categorized elements $\mathcal{E}$, a set of relations $\mathcal{R}$ and the task ψ, and generates a step-by-step strategy $\mathcal{S}$. We apply Chain-of-Thought to verbalize each step of its thought process. It selects necessary interactable elements from $\mathcal{E}$ according to $\mathcal{R}$, and formulates the sequential actions to complete ψ.

3.6 CAPTCHA Solving

Given the step-by-step strategy $\mathcal{S}$, and task ψ, we solve the reCAPTCHA v3 with two agents: ❶ Code Agent generates the necessary javascript code according to $\mathcal{S}$, and ❷ Running Agent executes the code to complete the task ψ.

Algorithm 1 Tool Enhancing

Input: Basic tool t, description of the instruction α
Output: Enhanced tool t'

```
if t = None then
    t ← basic_tool_generation(α)
end if
t' ← t
for i ≤ max_iteration do
    s ← SEARCH(t', α), v ← EVAL(t', s)
    if v ≥ threshold then
        break
    end if
    t' ← ADVANCE(t')
end for
Return t'
```

Tool Agent. Guided by the step-by-step strategy $\mathcal{S}$, the agent defines several browser automation tools, which are shown in Table 2. For example, *moving* the mouse between two interactable elements, *typing* the text on the element. To circumvent human-like operation, the tool agent iteratively searches the web for genuine human behaviors pattern, evaluates them, and subsequently advances the tool as outlined in Algorithm 1. Specifically, the operation $s \leftarrow \textsc{Search}(\cdot)$ retrieves human behavior patterns from the network and transmits to $\textsc{Eval}(\cdot)$ for assessment. If the evaluation reports a failure, it invokes $\textsc{Advance}(\cdot)$ to modify the underlying code, injecting more sophisticated human-like behavioral patterns into the tool.

Running Agent. After generating browser automation tools in the previous stage, the running agent obtains the concrete information by abstracting the webpage and corresponding $\mathcal{E}$, such as the elements' coordinates (x_d, y_d) for $d \in \mathcal{D}$, and optimizing the parameters used in the tools. Finally, the running agent autonomously executes the strategy and invokes the tools according to $\mathcal{S}$.

4 Experiments

Our evaluation focuses on the following research questions:

- **Feasibility (RQ1):** Can LLM-driven agents bypass reCAPTCHA v3 in unmodified state?
- **Behavioral Feature (RQ2):** Which aspects of dynamic factors influence the risk analysis?
- **Specific Performance (RQ3):** How do behavior features perform across different specific tasks?
- **User Study (RQ4):** How effective is Seqion in achieving human-like scores in web browsing?
- **Practical Efficacy (RQ5):** How does Seqion perform in practical production environments?

4.1 FEASIBILITY (RQ1)

We evaluate reCAPTCHA v3 evasion on a PHP social testbed KLik team (2024) using an LLM-powered agentic framework based on `browser-use` Magnus Müller and Gregor Žunic (2024). Operating in a closed-loop setup on a local server, the agents autonomously execute sequential web interactions from an initial prompt to final submission. For the core evaluation, we task these agents with completing a signup process that requires basic operations. Fig. 3 presents statistical score of baseline in bypassing the reCAPTCHA v3, which includes minimum, median, mean, maximum values and standard deviation for the following LLM backbones: GPT-5-mini, GPT-4.1-mini, Gemini-2.5-pro and Deepseek-v3.1. Specifically, the baseline mean score of 0.52 remains well below the human performance of 0.68. This indicates that although the agent completes the signup task, reCAPTCHA v3 still flags their activity as non-human. More details about the experimental parameter is provided in Appendix C.1

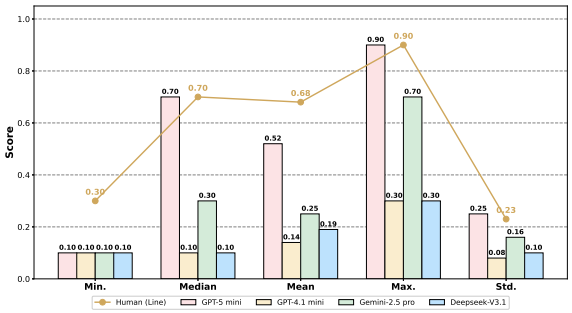
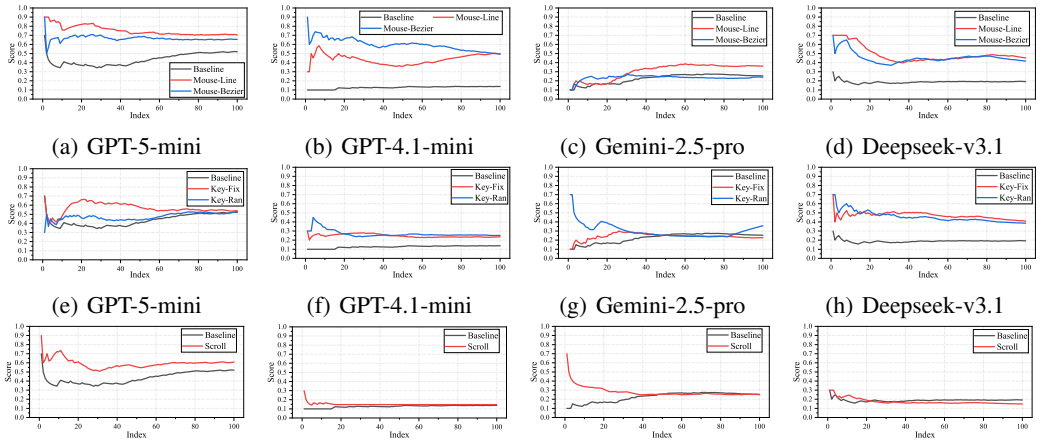


Figure 3: Statistical score of baseline

4.2 BEHAVIORAL FEATURES (RQ2)

Behavioral Factors. To investigate how behaviors affect evasion success rates, we evaluate dynamic behaviors as follows. ❶ Mouse movement: rather than instant mouse jumping, we simulate straight-line movements between web elements, then further utilize Bézier curve to generate smooth and scalable curves; ❷ Keystroke dynamics: rather than machine-paced typing, we emulate human typing variation by inserting a fixed and random delay between consecutive keystrokes; ❸ Scroll pattern: rather than using mechanical scrolling, we implement a non-uniform pattern with variable speed and intermittent pauses to generate a smoother scrolling.



(a) GPT-5-mini (b) GPT-4.1-mini (c) Gemini-2.5-pro (d) Deepseek-v3.1

(e) GPT-5-mini (f) GPT-4.1-mini (g) Gemini-2.5-pro (h) Deepseek-v3.1

(i) GPT-5-mini (j) GPT-4.1-mini (k) Gemini-2.5-pro (l) Deepseek-v3.1

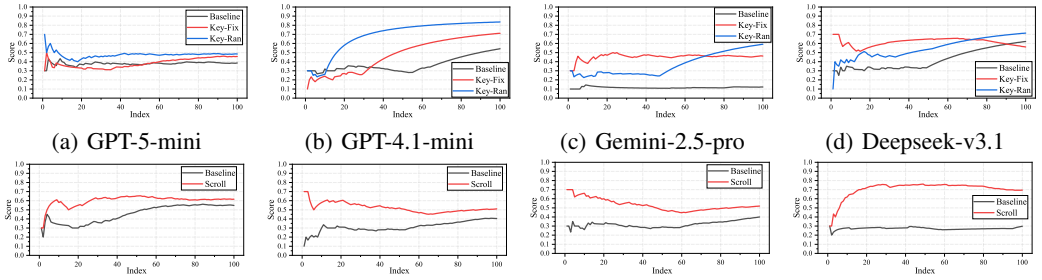
Figure 4: Average cumulative score when incorporating mouse, keystroke and scroll behaviors, for Signup

Experiment Results. Fig. 4 shows the average cumulative score when incorporating mouse movement, keystroke dynamics, and scroll pattern across LLM backbones. As shown in Figs. 4(a)-4(d), adding trajectories heavily mitigates the low-scoring baseline of 0.2, though Bézier curves provide minimal gains over straight lines. In Figs. 4(e)-4(h)), keystroke dynamics also enhance performance, but with varied stability—some exhibit temporary spikes, while others maintain a steady 0.15 increase. Scroll patterns, shown in Figs. 4(i)-4(l), fail to yield comparable improvements due to the webpage's constrained scrollable area. This limitation directly motivates the scroll-specific investigation in Section 4.3. Overall, the integration of dynamic behavior, especially refined mouse

movement simulation, proves beneficial for evasion in the detection of reCAPTCHA v3. Further experimental details and analysis of static features' impact, are provided in Appendix C.2 and D.

4.3 Specific Performance (RQ3)

Specific Task. To investigate how behaviors perform in specific tasks, we evaluate ❶ keystroke dynamics performance in tasks requiring extensive text input and ❷ scroll pattern performance in tasks requiring sustained scrolling. We instruct agent to complete a forum replying task and blog reading task respectively. The task includes navigation to the login page, authentication with pre-registered credentials, and submission of an extended reply (resp. reading with a scroll pattern).



(a) GPT-5-mini (b) GPT-4.1-mini (c) Gemini-2.5-pro (d) Deepseek-v3.1

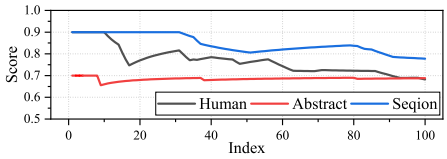
(e) GPT-5-mini (f) GPT-4.1-mini (g) Gemini-2.5-pro (h) Deepseek-v3.1

Figure 5: Average cumulative score when incorporating keystroke and scroll for specific task

Experiment Results. Fig. 5 shows average cumulative score when incorporating keystroke and scroll for specific tasks. As shown in Figs. 5(a)-5(d), keystroke dynamics significantly boost performance on a keystroke-specific task. The keystroke-based solutions achieve higher scores compared to the signup scenario. Figs. 5(e)-5(h) demonstrate that scroll patterns improve performance on scrolling-specific tasks. Overall, behavioral simulation works best when it matches a task's primary interaction mode, which offers a practical basis for evasion strategies against reCAPTCHA v3. Further experimental details are provided in Appendix C.3.

4.4 User Study (RQ4)

Solver Comparison. We present a comparative analysis of Human, Abstract Tsingenopoulos et al. (2022), and Seqion performance in bypassing reCAPTCHA v3 under identical conditions, as summarized in Fig. 6. Seqion achieves a mean score of 0.78, which outperforms the Abstract solver and human by an additional 0.09. However, the score of Seqion decreases over iterations, which is identical to human. This is because Seqion and human are target-oriented, which perform sequential decisions to complete a concrete task. The Abstract solver focuses on information such as action duration and hover timing, without consideration of the task itself.



Comp.	Min.	Median	Mean	Max.	Std.
Human	0.3	0.7	0.68	0.9	0.23
Abstract	0.3	0.7	0.69	0.7	0.07
Seqion	0.3	0.7	0.78	0.9	0.18

Figure 6: Solver comparison. **Left:** The average cumulative score. **Right:** Statistical score.

Component-wise Analysis. To isolate design contributions, we compare four progressive variants: R1 (base website extraction only), R2 (R1 with strategic planning), R3 (R1 with captcha solving) and R4 (full Seqion). Table 3 and Fig. 7 illustrate score performance of Seqion under different component configurations. R1 ensures the foundation for the entire system to execute. The jump from R1 to R2 indicates that action analysis of elements and the development of a series of solution strategies can improve the score. The sharp improvement from R1 to R3 highlights the tool agent's significant contribution, since it simulates a range of human behaviors throughout the system. Sequential decision-making helps evade detection by simulating task-oriented human

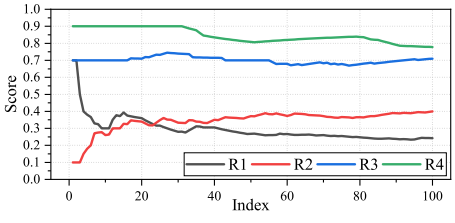


Figure 7: Average cumulative score ablation

behaviors. Table 3 shows high computational cost during strategic planning phase across configurations. These results align with the complexity analysis in Appendix A, highlighting a quadratic dependency on the webpage scale, $\mathcal{O}(N^2)$.

Table 3: Component-wise ablation and consumption.

Comp.	Min.	Median	Mean	Max.	Std.	Time	Prompt.	Completion	Total.	Spend
R1	0.1	0.1	0.24	0.9	0.19	81s	4.07k	6.42k	10.5k	$0.02
R2	0.1	0.3	0.40	0.9	0.24	251s	42.6k	21.5k	64.1k	$0.08
R3	0.1	0.7	0.71	0.9	0.22	111s	8.08k	9.96k	18.0k	$0.03
R4	0.3	0.9	0.78	0.9	0.18	251s	41.9k	18.5k	60.5k	$0.07

"Prompt.", "Completion.", "Total." denote "Prompt Tokens", "Completion Tokens", and "Total Tokens".

Score Distribution. Fig. 8 shows the score distribution across LLM backbones, indicating that successful evasion depends on simulating low-level interactions and planning. Models like GPT-5-mini excel at translating behavioral guidance into sequences of human-like interactions. Moreover, Seqion with dynamic behaviors outperforms human-level operation, which yields a success rate of 90% (score ≥ 0.7). Although Abstract attains a higher success rate, it fails to achieve a higher score of 0.9. We attribute this discrepancy to a lack of explicit task-specific consideration in its design.

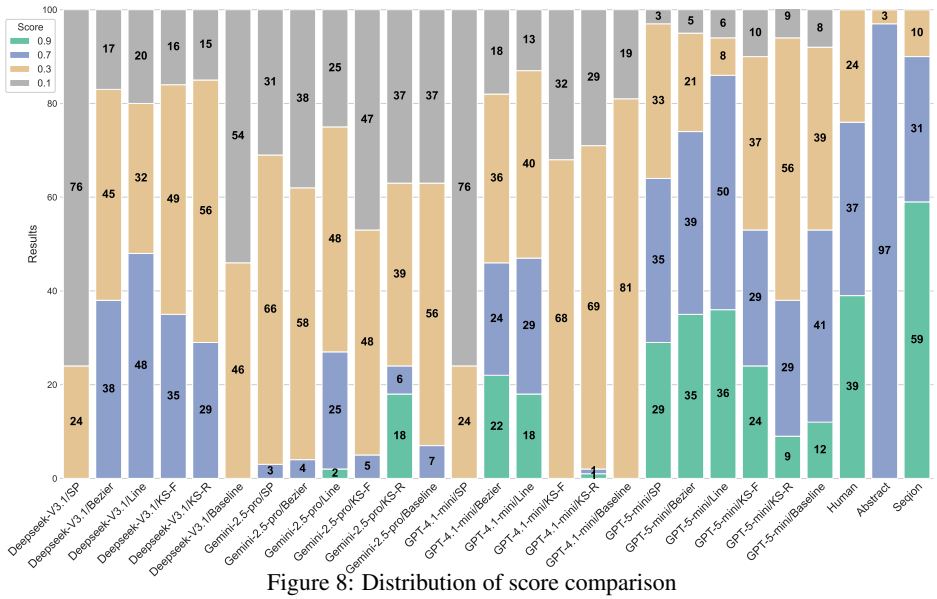


Figure 8: Distribution of score comparison

In summary, the LLM-powered agentic system Seqion achieves the comparable performance with human, urging a critical reevaluation of behavior-based verification.

4.5 Practical Efficacy (RQ5)

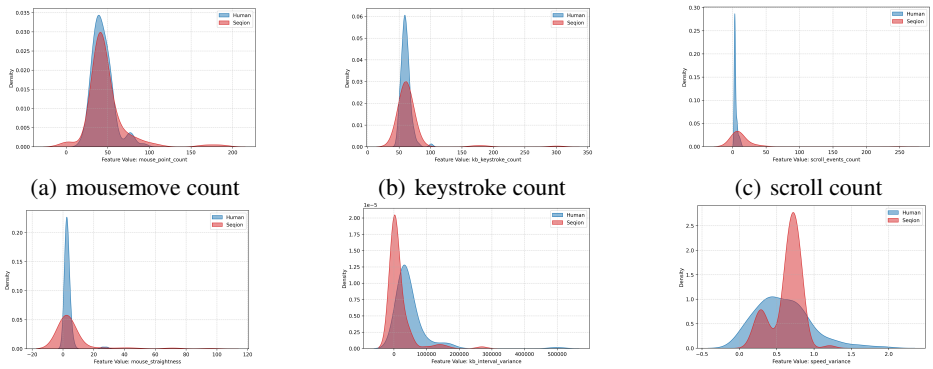
Experiment Setup. To ensure practical applicability beyond our self-hosted testbed, we evaluate Seqion on an external website, which sees considerable web traffic. Interactions are tracked via `rrweb` rrweb team (2026) following the configurations outlined in Appendix K and analyzed across six indicators defined in Appendix G.

KL divergence. Fig. 9 visualizes the KL divergence between human and Seqion interactions. While Seqion achieves near-perfect macroscopic overlap in interaction volumes, as shown in Figs. 9(a), 9(b)), nuanced micro-dynamic divergences exist. Specifically, Seqion shows wider curvature variance from B'ezier injections in Fig. 9(d) and a tighter temporal keystroke distribution lacking the human cognitive long-tail in

Table 4: KL divergence comparison.

Comp.	Human	Abstract	Seqion
Mouse Count	-	13.8	2.60
Trajectories Curvature	-	17.8	2.37
Keystroke Count	-	1.45	0.75
Keystroke Variance	-	1.26	10.6
Scroll Count	-	2.94	3.63
Scroll Variance	-	23.1	3.18
Multivariate KL	-	809	93.3

Fig. 9(e). Nevertheless, unlike the severe deviations of Abstract in Appendix F, Seqion's footprint stays within permissible high-density human regions, effectively minimizing KL divergence to defeat continuous authentication. In addition, to quantify the behavioral distributional discrepancy within a joint feature space, we employ the Multivariate KL divergence, as detailed in Appendix F. In Table 4, Seqion significantly reduces KL divergence across most individual metrics, demonstrating its ability to effectively mimic low-level human physical traits. By reducing the joint divergence from Abstract's 809 to 93.3, Seqion demonstrates alignment with the multi-dimensional human behavioral distribution, yielding resilience against continuous detection.



(a) mousemove count (b) keystroke count (c) scroll count

(d) trajectories curvature (e) keystroke variance (f) scroll variance

Figure 9: Kullback-Leibler divergence between Human and Seqion on external website

Solver Comparison. As shown in Fig. 11, Seqion evades reCAPTCHA v3 on an external website, achieving a 0.60 mean score that outperforms the Abstract and approaches human performance (0.69). While Seqion initially mirrors human-like score spikes and settling patterns, its performance degrades over prolonged iterations where human scores stabilize, suggesting the backend progressively penalizes subtle programmatic repetitions. Despite this, Seqion exhibits robust overall resilience. As detailed in Fig. 10, while 79% of Abstract sessions are trapped at low-trust scores (≤ 0.3), Seqion secures high-trust scores (≥ 0.7) in 73% of cases.

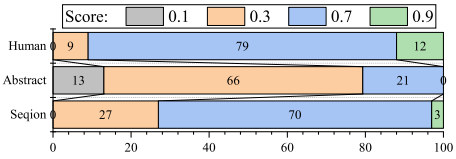
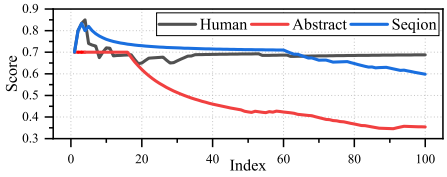


Figure 10: Distribution of score comparison



Comp.	Min.	Median	Mean	Max.	Std.
Human	0.3	0.7	0.69	0.9	0.14
Abstract	0.1	0.3	0.34	0.7	0.20
Seqion	0.3	0.7	0.60	0.9	0.18

Figure 11: Comparison on external website. **Left:** The average cumulative score. **Right:** Statistical score.

5 Conclusion

Summary. In this paper, we model behavior-based CAPTCHA solving as a sequential decision-making and planning problem. Extensive evaluation is conducted on websites with varied behavioral interactions for analysis on the resulting risk score. The results show that, our proposed agentic AI-based automation framework, Seqion, can consistently increase trust scores and achieve up to a 90% bypass success rate. Moreover, real-world evaluations on an external website confirm a mean trust score of 0.60 for Seqion. This decisively surpasses the 0.34 state-of-the-art baseline and closely tracks the 0.69 human average. These results demonstrate that even attackers with limited expertise can bypass reCAPTCHA v3, urging a critical reevaluation of behavior-based verification.

Broader impacts. While this research positively catalyzes the enhancement of behavior-based verification systems, it carries the inherent risk of dual-use. The most significant harmful consequence is that adversaries could exploit Seqion to bypass continuous authentication, enabling scalable automated web abuse while evading traditional bot-detection mechanisms.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

Generative AI tools were used to assist with research brainstorming, drafting of the theoretical and experimental design, code development and debugging of the browser-automation JavaScript executed by the Tool Agent, result analysis, literature summarization, figure preparation, language polishing, and manuscript editing. All AI-assisted content was independently reviewed and verified by the authors. The experimental methodology, the collection of human interaction traces, and the statistical analyses were performed manually, and LLM-generated code was tested for correctness by the authors, who take full responsibility for the final paper.

ETHICS STATEMENT

(This section is **recommended** and does not count toward the page limit.)

This work studies the synthesis of human-like web interaction traces to bypass behavior-based CAPTCHAs, which carries an inherent dual-use risk. All automated interaction was confined to a self-hosted testbed, and the Tool Agent's JavaScript was never executed against live production systems. On the external website, we obtained explicit consent from the owners and operated in a passive monitoring mode in which no action threshold was enforced and low risk scores were logged for analysis rather than triggering secondary verification challenges. Human interaction traces were collected from consenting volunteers and contain only anonymized pointer coordinates, key-press timestamps, and scroll positions, with no account identifiers, IP addresses, or other personally identifiable information retained. Because this work exposes a real vulnerability in a widely deployed system, we intend to follow a responsible-disclosure process with the affected vendor prior to or concurrent with publication. The primary aim of the work is defensive: by exposing how LLM-driven agents can synthesize fine-grained human dynamics, we urge the security community to develop more resilient, multi-dimensional verification mechanisms; the elevated trust scores reported here should not be interpreted as a complete security guarantee.

REPRODUCIBILITY STATEMENT

(This section is **recommended** and does not count toward the page limit.)

The experimental protocol, fixed parameters, and success criterion for every reported result are tabulated in Table 10. Section 4 specifies the evaluation protocol, and the appendix (Complexity Analysis, Behavior Collection, and Experimental Configuration) gives the method, metric definitions, and implementation details. The evaluation testbed is the publicly available KLiK social-media website KLik team (2024) protected by reCAPTCHA v3, built on the open-source `browser-use` Magnus Müller and Gregor Žunic (2024). The complete prompt templates and a representative Tool Agent JavaScript sample are provided in Appendix I and L. Code, configurations, and scripts used to generate all reported results will be included in the supplementary material.

REFERENCES

Hadi Abdullah, Aditya Karlekar, Saurabh Prasad, Muhammad Sajidur Rahman, Logan Blue, Luke A Bauer, Vincent Bindschaedler, and Patrick Traynor. Attacks as defenses: Designing robust audio captchas using attacks on automatic speech recognition systems. *arXiv preprint arXiv:2203.05408*, 2022.

Mohamed Salim Aissi, Clémence Grislain, Mohamed Chetouani, Olivier Sigaud, Laure Soulier, and Nicolas Thome. Viper: Visual perception and explainable reasoning for sequential decision-making. *arXiv preprint arXiv:2503.15108*, 2025.

Ismail Akrout, Amal Feriani, and Mohamed Akrout. Hacking google recaptcha v3 using reinforcement learning. *arXiv preprint arXiv:1903.01003*, 2019.

Mrim Alnfiai. A novel design of audio captcha for visually impaired users. *International Journal of Communication Networks and Information Security*, 12(2):168–179, 2020.

Henry S Baird and Terry P Riopka. Scattertype: a reading captcha resistant to segmentation attack. In *Document Recognition and Retrieval XII*, volume 5676, pp. 197–207. SPIE, 2005.

Kevin Bock, Daven Patel, George Hughey, and Dave Levin. {unCaptcha}: A {Low-Resource} defeat of {reCaptcha's} audio challenge. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

Built with. recaptcha v3 usage statistics. https://trends.builtwith.com/widgets/reCAPTCHA-v3, 2025.

Elie Bursztein, Matthieu Martin, and John Mitchell. Text-based captcha strengths and weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 125–138, 2011.

Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C Mitchell. The end is nigh: Generic solving of text-based {CAPTCHAs}. In *8th USENIX Workshop on Offensive Technologies (WOOT 14)*, 2014.

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*, pp. 3683–3691, 2025.

Darryl D'Souza, Phani C Polina, and Roman V Yampolskiy. Avatar captcha: Telling computers and humans apart via face classification. In *2012 IEEE International Conference on Electro/Information Technology*, pp. 1–6. IEEE, 2012.

Jeremy Elson, John R Douceur, Jon Howell, and Jared Saul. Asirra: a captcha that exploits interest-aligned manual image categorization. *CCS*, 7(366-374):15, 2007.

Valerie Fanelle, Sepideh Karimi, Aditi Shah, Bharath Subramanian, and Sauvik Das. Blind and human: Exploring more usable audio {CAPTCHA} designs. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, pp. 111–125, 2020.

Rony Ferzli, Rida Bazzi, and Lina J Karam. A captcha based on the human visual systems masking characteristics. In *2006 IEEE International Conference on Multimedia and Expo*, pp. 517–520. IEEE, 2006.

Haichang Gao, Jeff Yan, Fang Cao, Zhengya Zhang, Lei Lei, Mengyun Tang, Ping Zhang, Xin Zhou, Xuqin Wang, Jiawei Li, et al. A simple generic attack on text captchas. In *NDSS*, 2016.

Google. recaptcha v1. https://developers.google.com/recaptcha/docs/versions#recaptcha_v1_-_shut_down, 2025a.

Google. recaptcha v2. https://developers.google.com/recaptcha/docs/versions#recaptcha_v2_im_not_a_robot_checkbox, 2025b.

Google. recaptcha v3. https://developers.google.com/recaptcha/docs/versions#recaptcha_v3, 2025c.

Meriem Guerar, Luca Verderame, Mauro Migliardi, Francesco Palmieri, and Alessio Merlo. Gotta captcha'em all: a survey of 20 years of the human-or-computer dilemma. *ACM Computing Surveys (CSUR)*, 54(9):1–33, 2021.

hcaptcha. hcaptcha. https://www.hcaptcha.com, 2025.

Md Imran Hossen and Xiali Hei. A low-cost attack against the hcaptcha system. In *2021 IEEE Security and Privacy Workshops (SPW)*, pp. 422–431. IEEE, 2021.

Md Imran Hossen, Yazhou Tu, Md Fazle Rabby, Md Nazmul Islam, Hui Cao, and Xiali Hei. An object detection based solver for {Google's} image {reCAPTCHA} v2. In *23rd international symposium on research in attacks, intrusions and defenses (RAID 2020)*, pp. 269–284, 2020.

Denis O Ishkov and Valery I Terekhov. Text captcha traversal with convnets: Impact of color channels. In *2022 4th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*, pp. 1–5. IEEE, 2022.

Ramanpreet Kaur et al. A non ocr approach for math captcha design based on boolean algebra using digital gates to enhance web security. In *2016 international conference on wireless communications, signal processing and networking (WiSPNET)*, pp. 862–866. IEEE, 2016.

KLik team. Klik-socialmediawebsite. `https://github.com/msaad1999/KLiK-SocialMediaWebsite`, 2024.

Hyun Kwon, Hyunsoo Yoon, and Ki-Woong Park. Captcha image generation: two-step style-transfer learning in deep neural networks. *Sensors*, 20(5):1495, 2020.

Daniel Lopresti, Chilin Shih, and Gregory Kochanski. Human interactive proofs for spoken language interfaces. In *Proc. of the Workshop on Human Interactive Proofs*, pp. 30–34, 2002.

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. *CoRR*, abs/2505.24878, 2025. doi: 10.48550/ARXIV.2505.24878.

Magnus Müller and Gregor Žunic. Browser use: Enable ai to control your browser. `https://github.com/browser-use/browser-use`, 2024.

Goran Martinović, Andrew Attard, and Zdravko Krpić. Proposing a new type of captcha: character collage. In *2011 Proceedings of the 34th International Convention MIPRO*, pp. 1447–1451. IEEE, 2011.

Maryam Mehrnezhad, Abbas Ghaemi Bafghi, Ahad Harati, and Ehsan Toreini. Pishi: click the images and i tell if you are a human. *International Journal of Information Security*, 16(2):133–149, 2017.

Hendrik Meutzner, Santosh Gupta, and Dorothea Kolossa. Constructing secure audio captchas by exploiting differences between humans and machines. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pp. 2335–2338, 2015.

Deapesh Misra and Kris Gaj. Face recognition captchas. In *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, pp. 122–122. IEEE, 2006.

Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: Breaking a visual captcha. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pp. I–I. IEEE, 2003.

Hossein Nejati, Ngai-Man Cheung, Ricardo Sosa, and Dawn CI Koh. Deepcaptcha: an image captcha based on depth perception. In *Proceedings of the 5th ACM multimedia systems conference*, pp. 81–90, 2014.

Jiawei Nian, Ping Wang, Haichang Gao, and Xiaoyan Guo. A deep learning-based attack on text captchas by using object detection techniques. *IET Information Security*, 16(2):97–110, 2022.

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

Steven A Ross, J Alex Halderman, and Adam Finkelstein. Sketcha: a captcha based on line drawings of 3d models. In *Proceedings of the 19th international conference on World wide web*, pp. 821–830, 2010.

rrweb team. Record and replay the web. `https://www.rrweb.io`, 2026.

Guido Schryen, Gerit Wagner, and Alexander Schlegel. Development of two novel face-recognition captchas: a security and usability study. *Computers & security*, 60:95–116, 2016.

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*, pp. 1391–1406, 2020.

Chenghui Shi, Xiaogang Xu, Shouling Ji, Kai Bu, Jianhai Chen, Raheem Beyah, and Ting Wang. Adversarial captchas. *IEEE transactions on cybernetics*, 52(7):6095–6108, 2021.

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)*, pp. 388–403, 2016a.

Suphannee Sivakorn, Jason Polakis, and Angelos D Keromytis. I'm not a human: Breaking the google recaptcha. *Black Hat*, 14:1–12, 2016b.

Mengyun Tang, Haichang Gao, Yang Zhang, Yi Liu, Ping Zhang, and Ping Wang. Research on deep learning techniques in breaking text-based captchas and designing image-based captcha. *IEEE Transactions on Information Forensics and Security*, 13(10):2522–2537, 2018.

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)*, pp. 3747–3766, 2025.

Achint Oommen Thomas, Amalia Rusu, and Venu Govindaraju. Synthetic handwritten captchas. *Pattern Recognition*, 42(12):3365–3373, 2009.

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)*, pp. 719–735. IEEE, 2022.

Alan M Turing. Computing machinery and intelligence. In *Parsing the Turing test: Philosophical and methodological issues in the quest for the thinking computer*, pp. 23–65. Springer, 2007.

Haipeng Wang, Feng Zheng, Zhuoming Chen, Yi Lu, Jing Gao, and Renjia Wei. A captcha design based on visual reasoning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1967–1971. IEEE, 2018.

Ping Wang, Haichang Gao, Xiaoyan Guo, Zhongni Yuan, and Jiawei Nian. Improving the security of audio captchas with adversarial examples. *IEEE Transactions on Dependable and Secure Computing*, 21(2):650–667, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Xiangyu Wu, Yuwei Hu, Tianyu Cui, Yueying Tian, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, Yang Yang, and Jianfeng Lu. Mirrorcaptcha: Wild captcha, wild distribution, wild web-based platform meet multimodal llm agents. *None*, 2026. URL https://openreview.net/pdf?id=75YPEsknhT.

Xin Xu, Lei Liu, and Bo Li. A survey of captcha technologies to distinguish between human and computer. *Neurocomputing*, 408:292–307, 2020. ISSN 0925-2312.

Jeff Yan. Bot, cyborg and automated turing test: (or "putting the humanoid in the protocol"). In *International Workshop on Security Protocols*, pp. 190–197. Springer, 2006.

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*, pp. 332–348, 2018.

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.

Yufeng Zhang, Jialu Pan, Li Ken Li, Wanwei Liu, Zhenbang Chen, Xinwang Liu, and Ji Wang. On the properties of kullback-leibler divergence between multivariate gaussian distributions. *Advances in neural information processing systems*, 36:58152–58165, 2023.

Bin B Zhu, Jeff Yan, Qiujie Li, Chao Yang, Jia Liu, Ning Xu, Meng Yi, and Kaiwei Cai. Attacks and design of image recognition captchas. In *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 187–200, 2010.

APPENDIX

A COMPLEXITY ANALYSIS

We formally analyze the computational complexity of the Seqion framework, focusing on its two dominant phases: *Strategic Planning* and *CAPTCHA Solving*. Let $\mathcal{E}$ denote the set of parsed web elements, $\mathcal{D}$ denote the set of extracted descriptions, S be the sequence length of the generated strategy, and K be the maximum iteration limit for tool enhancement (Algorithm 1).

Strategic Planning Phase. The computational bottleneck in this phase lies in the *Description Distribution* and *Relation Extraction* module. To map semantic descriptions to DOM elements, the agent computes the cosine similarity between each description $d \in \mathcal{D}$ and the element set $\mathcal{E}$. This process necessitates pairwise comparisons, resulting in a time complexity of:

$$T_{\text{dist}} = \mathcal{O}(|\mathcal{E}| \cdot |\mathcal{D}|) \tag{5}$$

In addition, the extraction of the relation set $\mathcal{R}$ between elements incurs a computational complexity of $\mathcal{O}(|\mathcal{E}|^2)$, corresponding to the pairwise examination of elements. Subsequently, topological sorting over these relations requires $\mathcal{O}(|\mathcal{E}| + |\mathcal{R}|)$ operations. It is bounded by $\mathcal{O}(|\mathcal{E}|^2)$ and thus subsumed by the matching complexity in dense web environments where $|\mathcal{E}| \approx |\mathcal{D}|$. This process resulting in a time complexity of:

$$T_{\text{rel}} = \mathcal{O}(|\mathcal{E}|^2 + |\mathcal{E}| + |\mathcal{R}|) \tag{6}$$

Combining both module, the computational complexity of this phase is :

$$T_{\text{plan}} = \mathcal{O}(|\mathcal{E}| \cdot |\mathcal{D}| + |\mathcal{E}|^2) \tag{7}$$

CAPTCHA Solving Phase. The execution complexity is linear with respect to the strategy length. For each of the S steps, the *Tool Agent* performs a maximum of K optimization iterations (Search, Evaluate, Advance). The complexity is given by:

$$T_{\text{exec}} = \mathcal{O}(S \cdot K) \tag{8}$$

Total Complexity. Combining both phases, the overall computational complexity of Seqion is:

$$T_{total} \approx \mathcal{O}(|\mathcal{E}| \cdot |\mathcal{D}| + |\mathcal{E}|^2 + S \cdot K) \approx \mathcal{O}(N^2) \tag{9}$$

This derivation highlights a quadratic dependency on the webpage scale ($\mathcal{O}(N^2)$, where the webpage scale $N \approx |\mathcal{E}| \approx |\mathcal{D}|$). This theoretical bound aligns with our empirical observations in Table 3, where the Strategic Planning phase (R2) exhibits the highest computational and token overhead due to the intensive element-description matching process.

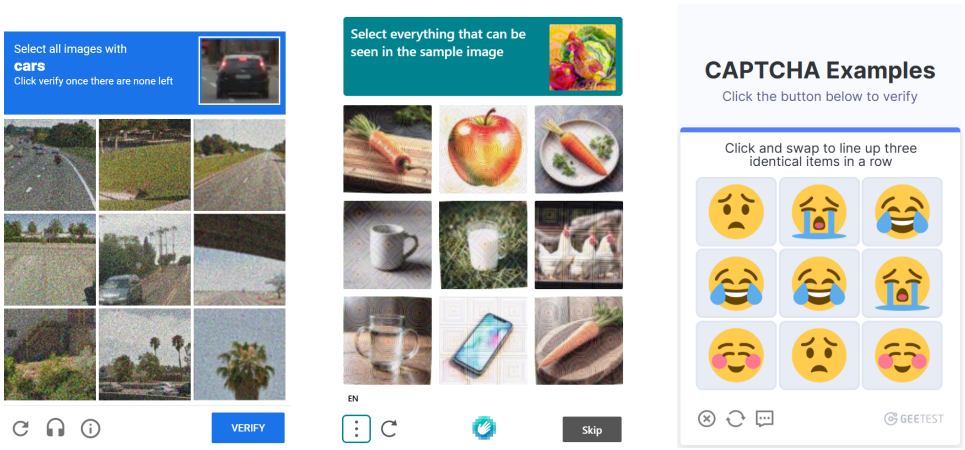
B BACKGROUND OF CAPTCHAS

The foundational concept of distinguishing humans from machines originates from Turing's seminal question, "*Can machines think?*" Turing (2007). This inquiry directly inspired the development of the CAPTCHA mechanism Yan (2006), which inverted the classic Turing test for practical security applications.

Text-based CAPTCHAs. In this category of CAPTCHAs, user is presented with a distorted text image and asked to identify and enter the characters shown. Early text-based systems relied on methods such as adding visual noise Baird & Riopka (2005), employing complex backgrounds Martinović et al. (2011); Ferzli et al. (2006), and exploiting recognition ability on handwritten text Thomas et al. (2009). More recently, adversarial techniques were introduced Shi et al. (2021); Kwon et al. (2020), where carefully crafted perturbations are injected into the CAPTCHA images to deceive machine learning models into making incorrect predictions.

Image-based CAPTCHAs. The underlying principle of image-based CAPTCHAs is to leverage the persistent human advantage in semantically interpreting visual scenes.

- *Image classification.* User is required to recognize the given images and select the target images, which involves image classification Google (2025b); Nejati et al. (2014), human face recognition Misra & Gaj (2006); D'Souza et al. (2012); Schryen et al. (2016), and objective detection Elson et al. (2007); Zhu et al. (2010) tasks.
- *Orientation-based.* User is required to recognize and correct the orientation of several misoriented images Ross et al. (2010); Mehrnezhad et al. (2017). While this label-free mechanism has been investigated, its effectiveness relies on the careful selection of images to prevent ambiguous interpretations.
- *Visual reasoning.* User is required to solve the puzzle based on a given question, such as solving a complex mathematical equation Kaur et al. (2016) and find specific objects Wang et al. (2018) in images. In this category of CAPTCHAs, the user must identify objects based on their visual attributes, including *shape* and *color*. Instructions may also specify comparative relationships such as "*Please click the biggest cylinder*" or spatial relationships such as "*Please click the cube right of the cone*."



(a) reCaptcha v2 (b) hCaptcha (c) GeeTest

Figure 12: Image-based CAPTCHAs

Aduio-based CAPTCHAs. The design of audio-based CAPTCHAs capitalizes on the gap in auditory perception between humans and automated systems, which were developed for visually impaired users Meutzner et al. (2015); Lopresti et al. (2002); Fanelle et al. (2020); Alnfiai (2020). These challenges require the user to listen and input the digits spoken or answer the question, for instance: "*what number is after x?*", where x is a randomly generated integer.

Behavior-based CAPTCHAs. To better balance security with seamless interaction, behavior-based CAPTCHAs were proposed, exemplified by reCAPTCHA v3 Google (2025c). Instead of presenting a one-shot, explicit challenge, they continuously and invisibly analyze user interactions such as mouse movement, keystroke dynamics during web browsing. This data is used to generate a risk score ranging from $\{0.1, 0.3, 0.7, 0.9\}$, which is then compared against a predefined threshold (score ≥ 0.7) to classify the session as human or automated.

C Experiment Results

C.1 Feasibility

We deploy reCAPTCHA v3 on a PHP-based social website KLik team (2024) supporting login/signup, forum replying and blog reading, and implement a browser-use Magnus Müller and Gregor Žunic (2024) system powered by different LLM backbones. The entire experimental infrastructure is hosted on an Ubuntu 22.04.5 LTS server provisioned with an AMD Ryzen Threadripper 3990X 64-Core Processor and dual NVIDIA GeForce RTX 3090 GPUs. Generally, these agents operate in a closed-loop setup: given a prompt, they open the target website and issue actions step-by-step until clicking the final submit button. To evaluate the feasibility of evading reCAPTCHA v3,

we task the LLM-driven agents with completing a signup process, which requires filling in profile info such as username, email, password and name.

Fig. 3 presents results of baseline in bypassing the reCAPTCHA v3, which includes including minimum, median, mean, maximum values and standard deviation for the following LLM backbones: GPT-5-mini, GPT-4.1-mini, Gemini-2.5-pro and Deepseek-v3.1. Average cumulative score are shown in Fig. 13, where GPT-5-mini demonstrates superior overall performance and a wider score distribution. Although agents complete the signup task, reCAPTCHA v3 still flags their activity as non-human.

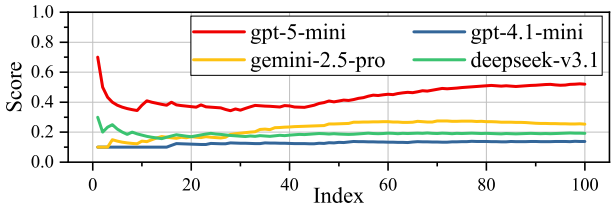


Figure 13: Average cumulative score of baseline across backbones

C.2 Behavior Factors

Fig. 4, and Table 5 shows the average cumulative score and statistical comparison when incorporating mouse movement, keystroke dynamics, and scroll pattern across LLM backbones.

❶ **Mouse Movement:** As illustrated in Figs. 4(a)-4(d), it depicts the average cumulative score for attempts by incorporating straight-line and Bézier-curve mouse movement. The baseline, i.e., without mouse movement, suffers from a lower score of around 0.2 since conventional automated clicks can be easily flagged as suspicious. Introducing mouse trajectories adds a more realistic imitation of human behavior and leads to higher scores. However, the advanced technique of exploiting Bézier curves does not greatly increase the score.

❷ **Keystroke Dynamics:** As illustrated in Figs. 4(e)-4(h), the integration of keystroke dynamics improves the performance. In Figs. 4(e) and 4(g), keystroke-based solutions initially achieve higher scores but later converge to the baseline level. Conversely, Figs. 4(f) and 4(h) show a consistent score increase of about 0.15 with keystroke simulation. Although the agent obtains a higher score with keystroke, it achieves a lower increase than the incorporation of mouse movement.

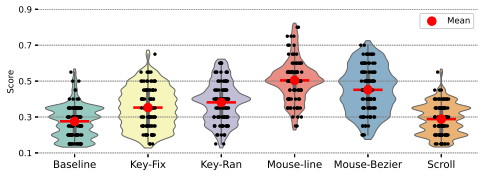


Figure 14: Score distribution with dynamic behaviors

❸ **Scroll Pattern:** The average score trends associated with scroll pattern are presented in Figs. 4(i)-4(l). In contrast to previously examined behaviors, the inclusion of scroll patterns does not yield comparable performance improvements. This result may be attributed to the constrained scrolling area available on the evaluated web page. Accordingly, these observations motivate an investigation of scroll-specific scenarios in Section 4.3.

Overall, Fig. 14 demonstrates that the integration of dynamic behavior, especially refined mouse movement simulation, proves beneficial for evasion in the detection of reCAPTCHA v3.

C.3 Specific Performance

Previous experiments evaluate behaviors on a signup page dominated by mouse interactions, with limited text input and scrolling. To specifically assess keystroke and scrolling behaviors, we designed two additional tasks: extended text replies and sustained blog reading. The analysis follows.

C.3.1 Keystroke Specific Task

To investigate how keystroke dynamics affect performance in tasks requiring extensive text input, we instruct agent to complete a forum replying task, which includes: ❶ navigation to the login page, ❷ authentication with pre-registered credentials, and ❸ submission of an extended reply.

Table 5: Statistical comparison when incorporating mouse, keystroke and scroll behaviors, for Signup.

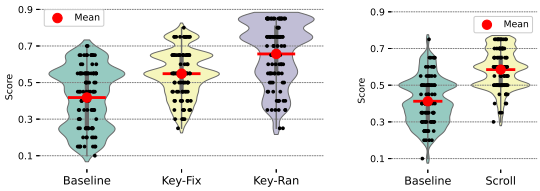
Signup		Baseline	Straight Lines	Bézier Curves	Keystroke Fix	Keystroke Random	Scroll Pattern
GPT-5-mini	Min.	0.1	0.1	0.1	0.1	0.1	0.1
	Median	0.7	0.7	0.7	0.7	0.7	0.7
	Mean	0.52	0.70	0.66	0.54	0.53	0.61
	Max.	0.9	0.9	0.9	0.9	0.9	0.9
	Std.	0.25	0.22	0.25	0.28	0.26	0.26
GPT-4.1-mini	Min.	0.1	0.1	0.1	0.1	0.1	0.1
	Median	0.1	0.3	0.3	0.3	0.3	0.1
	Mean	0.14	0.50	0.49	0.24	0.25	0.15
	Max.	0.3	0.9	0.9	0.3	0.3	0.3
	Std.	0.08	0.28	0.30	0.09	0.12	0.09
Gemini-2.5-pro	Min.	0.1	0.1	0.1	0.1	0.1	0.1
	Median	0.3	0.3	0.3	0.3	0.3	0.3
	Mean	0.25	0.36	0.24	0.23	0.36	0.25
	Max.	0.7	0.9	0.7	0.7	0.9	0.7
	Std.	0.16	0.23	0.13	0.15	0.29	0.12
Deepseek-v3	Min.	0.1	0.1	0.1	0.1	0.1	0.1
	Median	0.1	0.3	0.3	0.3	0.3	0.1
	Mean	0.19	0.45	0.42	0.41	0.39	0.15
	Max.	0.3	0.7	0.7	0.7	0.7	0.3
	Std.	0.100	0.249	0.232	0.23	0.21	0.09

Table 6: Statistical comparison of keystroke.

Specific Task		W/O KS	KS-F	KS-R			W/O KS	KS-F	KS-R
GPT-5-mini	Min.	0.1	0.1	0.1	Gemini-2.5-pro	Min.	0.1	0.1	0.1
	Median	0.3	0.3	0.3		Median	0.1	0.3	0.9
	Mean	0.39	0.46	0.48		Mean	0.12	0.46	0.59
	Max.	0.7	0.7	0.7		Max.	0.3	0.7	0.9
	Std.	0.20	0.22	0.22		Std.	0.06	0.21	0.34
GPT-4.1-mini	Min.	0.1	0.1	0.1	Deepseek-v3	Min.	0.1	0.1	0.1
	Median	0.3	0.9	0.9		Median	0.9	0.7	0.8
	Mean	0.54	0.71	0.84		Mean	0.62	0.56	0.71
	Max.	0.9	0.9	0.9		Max.	0.9	0.9	0.9
	Std.	0.33	0.31	0.19		Std.	0.32	0.21	0.25

We denote KS as keystroke, KS-F as keystroke-fix, KS-R as keystroke-random.

As shown in Figs. 5(a)-5(d), keystroke dynamics significantly boost performance on a keystroke-specific task. The keystroke-based solutions achieve higher scores compared to the signup scenario. Table 6 shows the statistical comparison in the forum reply task. Specifically, utilizing randomized keystroke intervals consistently outperforms fixed-interval typing, yielding a mean score increase of 0.11. Fig. 15(a) highlights the particular contribution of keystroke random to score improvement. Notably, the score gain from keystroke simulation with Gemini-2.5-pro is larger on the keystroke-specific task of 0.47 than on the signup task of 0.10. The difference likely arises because the forum task inherently prioritizes typing, amplifying the impact of keystroke simulation.



(a) Keystroke (b) Scroll

Figure 15: Score distribution with keystroke and scroll

C.3.2 Scroll Specific Task

To investigate how scroll pattern affect performance in tasks requiring sustained scrolling, we instruct the agent to complete a blog reading task, which includes: ❶ navigation to the login page, ❷ authentication with pre-registered credentials, and ❸ reading an article with a scroll pattern.

Table 7: Statistical comparison of scroll pattern.

Specific Task		W/O SP	SP		W/O SP	SP		W/O SP	SP		W/O SP	SP
Min.	GPT-5-mini	0.1	0.1	Gemini-2.5-pro	0.1	0.1	GPT-4.1-mini	0.1	0.1	Deepseek-v3.1	0.1	0.1
Median		0.7	0.7		0.3	0.7		0.3	0.7		0.3	0.7
Mean		0.55	0.62		0.40	0.52		0.40	0.51		0.30	0.69
Max.		0.9	0.9		0.7	0.7		0.7	0.7		0.9	0.9
Std.		0.26	0.25		0.24	0.24		0.24	0.24		0.15	0.21

We denote SP as scroll pattern.

Figs. 5(e)-5(h) demonstrate that scroll patterns improve performance on scrolling-specific tasks, though the gain is less substantial than in keystroke tasks in Fig. 15. Table 7 shows the statistical comparison in sustained blog reading. In particular, incorporating scroll pattern with Deepseek-v3.1 produces an average score increase of 0.39. This result is likely due to the blog task's emphasis on scrolling, which enhances the impact of scroll simulation.

Overall, behavioral simulation works best when it matches a task's primary interaction mode, which offers a practical basis for evasion strategies against reCAPTCHA v3.

D Static Behavior

In the experiment, we observed that static environmental factors can also influence the detection, specifically ❶ cookies and ❷ IP reputation.

D.1 Static Behavioral Factors

❶ **Cookies:** To simulate user sessions, we incorporate real user cookies during website access.
❷ **IP Reputation:** To simulate access from a different network, we employ a VPN for IP rotation.

D.1.1 Experiment Results

❶ **Cookies:** Table 8 compares the scores obtained without and with cookies, revealing a statistically significant contrast. Without cookies, it consistently receives low scores, whereas enabling cookies results in substantially higher scores. In Fig 16, the cookie-based solution drops continuously and reaches an average of 0.5 across multiple attempts. This finding highlights a critical vulnerability: the security of reCAPTCHA v3 heavily depends on cookies.

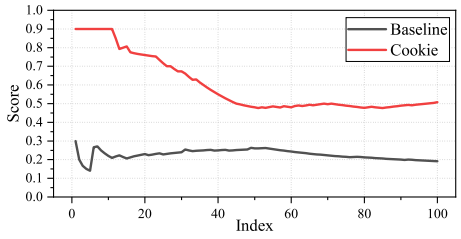


Figure 16: Average cumulative score in cookies

Table 8: Statistical comparison of cookies

Signup	W/O Cookies	Cookies
Min.	0.1	0.1
Median	0.1	0.7
Mean	0.19	0.51
Max.	0.9	0.9
Std.	0.14	0.28

❷ **IP Reputation:** Table 9 compares the score between without and with a new IP, the agent achieves a score of 0.83, representing a 59.6% increase over the baseline. Fig. 17 illustrates the cumulative score trend, showing how IP reputation contributes to higher performance across successive attempts. This finding indicates that reCAPTCHA v3's risk analysis is highly sensitive to IP reputation: leveraging a high-reputation IP address enables evasion against detection with trust scores.

E Experimental Configuration

In Table 10, we provide a complete experimental protocol table for each condition: the number of iterations, whether cookies, and IP were changed, the exact success criterion, the statistical test used for the experimental testbed.

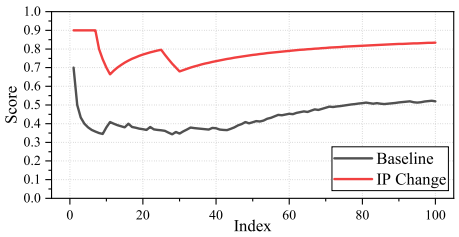


Figure 17: Average cumulative score in IP reputation

Table 9: Statistical comparison of IP reputation

Signup	W/O IP′	IP′
Min.	0.1	0.1
Median	0.7	0.9
Mean	0.52	0.83
Max.	0.9	0.9
Std.	0.25	0.21

Table 10: Experiment Configuration

Experiment		Iteration	Cookies	IP	Success Criterion	Website
Mouse Movement	W/O Mouse	100	Not Involved	IP	score ≥ 0.7	Self-hosted
	Straight Line	100	Not Involved	IP	score ≥ 0.7	Self-hosted
	Bezier Curve	100	Not Involved	IP	score ≥ 0.7	Self-hosted
Keystroke Dynamics	W/O Mouse	100	Not Involved	IP	score ≥ 0.7	Self-hosted
	Key Fix	100	Not Involved	IP	score ≥ 0.7	Self-hosted
	Key Random	100	Not Involved	IP	score ≥ 0.7	Self-hosted
Scroll Pattern	Mechanical	100	Not Involved	IP	score ≥ 0.7	Self-hosted
	Pattern	100	Not Involved	IP	score ≥ 0.7	Self-hosted
Cookies	W/O Cookies	100	Not Involved	IP	score ≥ 0.7	Self-hosted
	Cookies	100	Cookies	IP	score ≥ 0.7	Self-hosted
IP Reputation	W/O IP’	100	Not Involved	IP	score ≥ 0.7	Self-hosted
	IP’	100	Not Involved	IP’	score ≥ 0.7	Self-hosted
User Study	Human	100	Not Involved	IP	score ≥ 0.7	Self-hosted
	Abstract	100	Not Involved	IP	score ≥ 0.7	Self-hosted
	Seqion	100	Not Involved	IP	score ≥ 0.7	Self-hosted
Practicality	Human	100	Not Involved	IP	score ≥ 0.7	External
	Abstract	100	Not Involved	IP	score ≥ 0.7	External
	Seqion	100	Not Involved	IP	score ≥ 0.7	External

F KULLBACK-LEIBLER DIVERGENCE BETWEEN MULTIVARIATE GAUSSIAN DISTRIBUTIONS

F.1 METHODOLOGICAL OVERVIEW

In the study, we employ kullback-Leibler divergence to quantify the behavioral distributional discrepancy between Human, Abstract and Seqion within a joint feature space. Compared to univariate assessments, the multivariate KL divergence Zhang et al. (2023) effectively captures the underlying covariance structures among behavioral features, thereby providing a more comprehensive evaluation of the approaches in bypassing behavioral detection.

Let the target feature vector be $x \in \mathbb{R}^d$, where d denotes the feature dimension. We approximate the authentic human feature distribution P and the bot's generated distribution Q as multivariate Gaussians:

- Human (Ground Truth Distribution): $P \sim \mathcal{N}(\mu_P, \Sigma_P)$
- Bot (Simulated Distribution): $Q \sim \mathcal{N}(\mu_Q, \Sigma_Q)$

To mathematically measure how well the model distribution Q approximates the target distribution P, we compute the forward KL divergence $D_{KL}(Q, P)$.

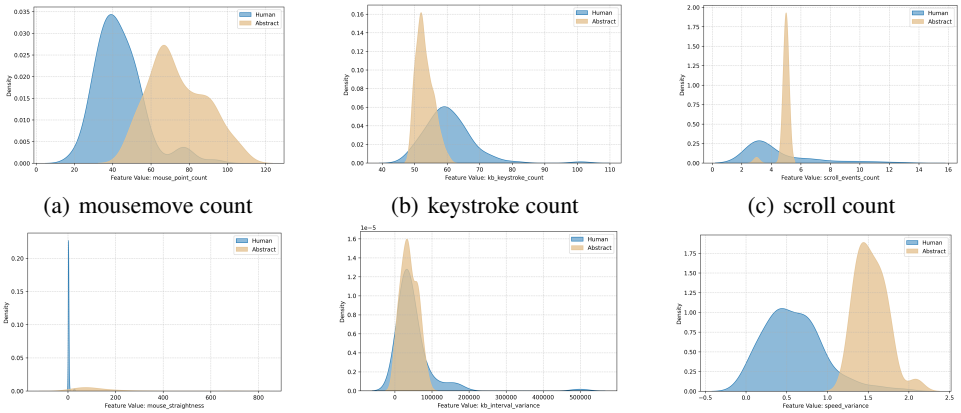
F.2 FORMULATION AND COMPUTATIONAL PROCESS

The KL divergence between two multivariate Gaussian distributions yields a closed-form solution, defined as follows:

$$D_{KL}(Q||P) = \frac{1}{2}\left[Tr(\Sigma_P^{-1}\Sigma_Q) + (\mu_P - \mu_Q)^T\Sigma_P^{-1}(\mu_P - \mu_Q) + log\left(\frac{|\Sigma_P|}{|\Sigma_Q|}\right) - d\right].$$

In our programmatic implementation, this equation is decoupled into four distinct terms for computation:

- $Tr(\Sigma_P^{-1}\Sigma_Q)$: It measures the variance ratio between the two distributions along the principal directions of the covariance matrices.
- $(\mu_P - \mu_Q)^T\Sigma_P^{-1}(\mu_P - \mu_Q)$: It quantifies the spatial distance between the mean vectors of the two distributions, weighted by the precision matrix (inverse covariance) of the Human distribution P.
- $log\left(\frac{|\Sigma_P|}{|\Sigma_Q|}\right)$: It reflects the differential entropy between the two distributions within the joint probability space.
- d: It denotes the number of features



(a) mousemove count (b) keystroke count (c) scroll count

(d) trajectories curvature (e) keystroke variance (f) scroll variance

Figure 18: Kullback-Leibler divergence between Human and Abstract on external website

G BEHAVIOR COLLECTION

To empirically evaluate the δ-indistinguishability (in Definition 2) of Seqion in real-world scenarios, we require a fine-grained capture of the continuous motor action τ_{mot}. To this end, we deploy `rrweb` rrweb team (2026), a lightweight web session recording library, on the target website to record incremental snapshots and interact event streams.

Let τ_{mot}^H and τ_{mot}^S denote the human motor traces and Seqion traces collected by `rrweb` respectively. Therefore, we have $\tau_{mot}^H \sim P_{\text{Human}}(\tau \mid \psi)$, and $\tau_{mot}^S \sim P_{\text{Seqion}}(\tau \mid \psi)$. Note that although we define $P_{\text{Seqion}}(\tau \mid \psi) = \pi_{plan}(\tau_{sem} \mid \psi) \cdot \pi_{act}(\tau_{mot} \mid \tau_{sem})$, the plan τ_{sem} is implicit in the traces collections. To compute the behavioral distance between these two distributions, we project the high-dimensional `rrweb` event logs into comparable statistical feature vectors: Macroscopic Volumetric Features and Microscopic Kinematic Features.

1) Macroscopic Volumetric Features: These metrics quantify the absolute volume of user engagement, acting as a proxy for task complexity and cognitive load. It includes: i) N_m, the total number of `mousemove` events recorded during the session, reflecting the user's visual tracking and exploratory behavior on the interface; ii) N_k, the total number of `keydown` and `keyup` events, capturing the absolute volume of text input and operational commands; iii) N_s, the total number of `scroll` events, indicating the extent of information foraging and page navigation.

2) Microscopic Kinematic Features: While volumetric features capture the *amount* of interaction, kinematic features capture *how* the actions are executed. These evaluate the inherent physical constraints and rhythmic variations of human motor control.

- Trajectories Curvature: Evaluated by calculating the straightness ratio of mouse movements. Let a mouse trajectories be defined as a sequence of N_m spatial points $P =$

$[p_1, p_2, \ldots, p_{N_m}$, where $p_i = (x_i, y_i)$. The curvature $\mathcal{C}_{tra}$ is formalized as the total actual distance traversed by the cursor divided by the Euclidean distance between the starting and ending points:

$$\mathcal{C}_{tra} = \frac{\sum_{i=1}^{N-1} \|p_{i+1} - p_i\|_2}{\|p_N - p_1\|_2 + \epsilon},$$

where $\|\cdot\|_2$ denotes the L_2 norm, and ϵ is a negligibly small constant preventing division by zero. This metric captures the inherent non-linear nature of human trajectories compared to programmatic movements.

- Keystroke Variance: Evaluated by calculating the statistical dispersion of typing intervals. Let a keystroke sequence be defined as a series of N_k temporal points $T = [t_1, t_2, \ldots, t_{N_k}]$, where t_i represents the timestamp of the i-th key press event. The keystroke variance $\mathcal{V}_{key}$ is formalized as the sample variance of the inter-keystroke intervals $\Delta t_i = t_{i+1} - t_i$:

$$\mathcal{V}_{key} = \frac{1}{N_k - 1} \sum_{i=1}^{N_k - 1} (\Delta t_i - \mu_k)^2,$$

where μ_k denotes the mean interval duration across the sequence. This metric captures the inherent rhythmic inconsistencies of human typing compared to the rigid temporal patterns of programmatic inputs.

- Scroll Variance: Evaluated by calculating the statistical variance of instantaneous scroll speeds. Let a scroll trace be defined as a sequence of N_s spatiotemporal events $S = [s_1, s_2, \ldots, s_{N_s}]$, where each event $s_i = (p_i, t_i)$ records the 2D viewport position $p_i = (x_i, y_i)$ at timestamp t_i. Given the scroll speed $v_i = \frac{\|p_{i+1} - p_i\|_2}{t_{i+1} - t_i}$, the scroll variance $\mathcal{V}_{scroll}$ is formalized as the sample variance of these speeds:

$$\mathcal{V}_{scroll} = \frac{1}{N_s - 1} \sum_{i=1}^{N_s - 1} (v_i - \mu_s)^2,$$

where μ_s denotes the mean scrolling velocity across the sequence. This metric captures the inherent dynamics of human scrolling compared to the programmatic viewport shifts.

Finally, we project the raw motor traces into the statistical feature space, represented by the tuple $\Phi(\tau_{mot}) = (N_m, N_k, N_s, \mathcal{C}_{tra}, \mathcal{V}_{key}, \mathcal{V}_{scroll})$. We then compute the Kullback-Leibler divergence $D_{KL}(P_{\text{Seqion}} \| P_{\text{Human}})$ across these behavioral modalities. By aggregating these divergence scores, we quantify the empirical indistinguishability bound δ. This derived δ provides a mathematical measure of how successfully the bot's synthesized actions mirror human behavioral dynamics in the wild.

H Discussion

Threshold. Although empirical evaluations of reCAPTCHA v3 remain relatively sparse in existing literature, prior study Tsingenopoulos et al. (2022) have established a passing score of ≥ 0.7. To maintain methodological consistency and ensure a fair comparison with established baselines, we adopt this threshold. Furthermore, this configuration closely aligns with real-world deployment practices, as benign human traffic typically generates trust scores between 0.7 and 0.9. Consequently, a 0.7 cutoff represents an optimal security-usability tradeoff: it effectively filters automated bot traffic while minimizing false positives for legitimate users.

Difference. The fundamental difference between Seqion and existing MLLM benchmarks Luo et al. (2025); Wu et al. (2026) lies in the challenge modality. Previous work predominantly treated CAPTCHAs as a computer vision problem, focusing on `what` is presented on the image. For instance, as illustrated in Fig. 12, reCaptcha v2 requires a user to select which image contains a car from an image grid. That is, existing MLLM solutions deal with the visual problem by visual classification and semantic reasoning. In contrast, the reCaptcha v3 evaluates `how` a user interacts with the web page. To address this, Seqion shifts the paradigm from visual perception to behavioral synthesis by deconstructing and simulating human-like interaction sequences.

Ethics. The JavaScript generated by the Tool Agent is strictly confined to a self-hosted testbed KLik team (2024). Consequently, this methodology mitigates potential ethical concerns associated with interacting with live production systems. Prior to deploying reCAPTCHA v3 on the external website, we obtained explicit consent from their respective owners and administrators. To mitigate any potential disruption to normal user traffic, we configured the system in a passive monitoring mode. Specifically, no action threshold was enforced, ensuring that low risk scores would strictly be logged for analysis rather than triggering secondary verification challenges. In addition, the primary positive impact of our work is defensive: by exposing how LLM-driven agents can successfully synthesize fine-grained human dynamics, we urge the security community to develop more resilient, multi-dimensional verification mechanisms that transcend simple trajectory analysis.

Low Score Analysis. Although Seqion can successfully execute complex interaction sequences, it occasionally betrays distinct programmatic artifacts and faces inherent operational limitations: ❶ Linear Trajectories. Unlike genuine human users, whose mouse paths are inherently curved due to biomechanical constraints, Seqion may traverse the shortest Euclidean distance. As illustrated in Fig. 22, this optimization results in highly unnatural, perfectly linear movements (e.g., maintaining a constant x-coordinate over time). ❷ Scroll Absence. Even when a target element resides entirely outside the visible viewport, Seqion often moves the cursor directly to the target's coordinates without triggering any prerequisite scroll events. In contrast, genuine users must first scroll the off-screen element into view before physical interaction. ❸ Repetition Penalties. A high volume of repetitive interactions inevitably degrades the reCAPTCHA v3 score. In our experiments, when repeatedly performing a specific task 100 times, their score dropped from an initial range of $\{0.7, 0.9\}$ to eventually fluctuate between 0.1 and 0.3. This outcome aligns with standard security heuristics: the highly repetitive execution of identical tasks is flagged as malicious behavior, regardless of human involvement. Consequently, no matter how closely Seqion mimics human kinematics, it remains unable to sustain high scores during large-scale, repetitive task execution.

Limition. Although Seqion achieves state-of-the-art performance in bypassing behavior-based CAPTCHAs, its practical deployment faces constraints: ❶ Degradation in Long-term Sessions. While Seqion successfully mimics human micro-dynamics in short-term interactions, its evasion efficacy marginally degrades over prolonged, continuous sessions. State-of-the-art backend systems like reCAPTCHA v3 may eventually detect latent programmatic regularities in the LLM-synthesized trajectories. ❷ Impact of Static Feature: This study predominantly isolates the impact of dynamic behaviors on reCAPTCHA v3, leaving static features such as Cookie and IP reputation in Appendix D. Future research could explore the synergistic effects of combining high-reputation static proxies with our behavioral generation.

I Prompt Template

The prompt template utilized to guide the agents is structured as follows.

Orchestrator

You are the central coordinator for a multi-agent system, whose responsibilities are:
1. Receive task objectives from the user.
2. Sequentially invoke the following agents to collaborate on completing the task. Parsing Agent: Parses the target webpage to extract all interactive elements and their descriptions. Element Categorizer: Categorizes the extracted elements based on their functionality. Description Distributor: Accurately associates the descriptions with the categorized elements. Relation Extractor: Analyzes the sequential execution relationships between elements based on the task objectives. Strategy Planner: Formulates specific execution steps based on the identified sequential relationships. Tool Agent: Generates JavaScript code for each step to simulate human-like interactions. Browser Agent: Executes the generated code within a browser environment to complete the actual task.

Strategy Planner

You are the Strategy Planner agent, whose responsibilities are:
1. Receive the sequential relationships between elements output by the Relation Extractor.
2. Output a maximum of 5∼ 7 steps, focusing on key operations aligned with the primary task objective.
3. Each step must include: Step number, Target element, Action type (e.g., click, input, scroll, etc.) and Mouse movements between elements.
The output format must be JSON, containing the following fields: 'step number', 'element id', and 'action'.

Tool Agent

You are the Tool agent, whose responsibilities are:
1. Receive the sequence of steps output by the Strategy Planner.
2. Generate concise JavaScript code for each step, covering: Mouse movements and clicks (including Bezier curve movements between two elements, executed over a specific duration). Keyboard input (simulating human typing with a probability of pauses and occasional backspacing). Page scrolling (including variations in speed, pauses, and minor positional adjustments). Waits and delays (simulating human reaction times).
3. Ensure that for any transition between two elements, a JavaScript function is used to move the mouse from the preceding element to the subsequent one via a Bezier curve.
The output format must be a code block that can be copied and executed directly in a web console.

J Social Media Website

We deploy reCAPTCHA v3 on a PHP-based social website KLik team (2024) supporting login/signup, forum replying and blog reading, where the self-hosted website page is shown in Fig. 19.

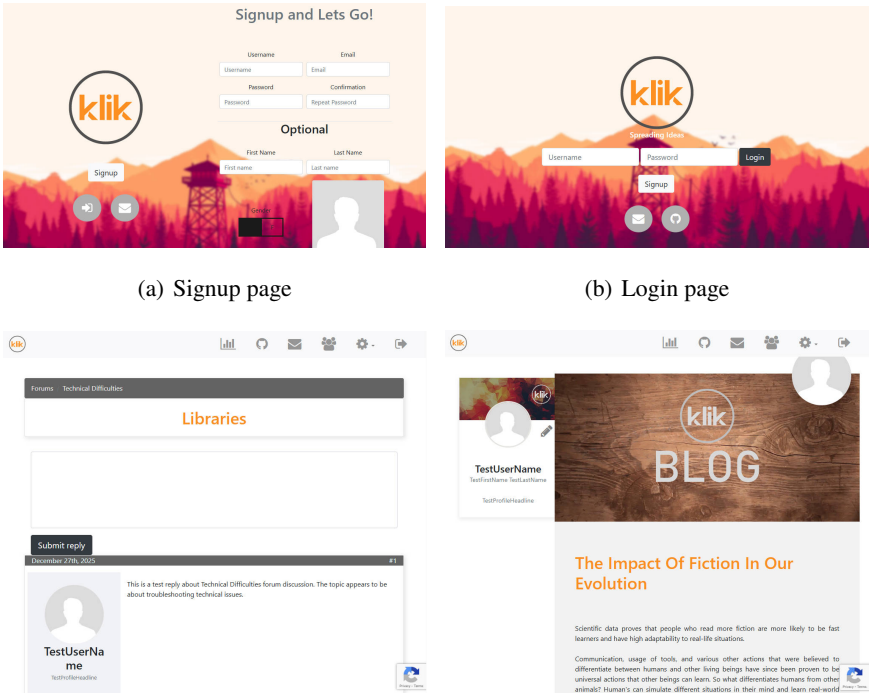
K Configuration and collection of rrweb

Lightweight behavioral trace collection via `rrweb`. As illustrated in Fig. 20, the configuration disables heavy DOM and media recording, specifically filtering and extracting interaction events, including mouse movements, scrolls, and keystrokes. In addition, we give an example to record mousemove, keystroke and scroll in Fig. 21 and a rrweb recording example for straight trajectories in Fig. 22.

L Example Generated by Tool Agent

The specific JavaScript snippet generated by the Tool Agent is presented below.

```
(async ()=>{
   // Short helpers
   const sleep = ms => new Promise(r=>setTimeout(r,ms));
   const rand = (a,b)=> a+Math.random()*(b-a);
   const center = el=>{const r=el.getBoundingClientRect();
       return {x:Math.round(r.left+r.width/2),
       y:Math.round(r.top+r.height/2)}};
   // Bezier mover between elements (dispatches mousemove
       events along curve)
```



(a) Signup page (b) Login page

(c) Forum page (d) Blog page

Figure 19: The social media website

```
let rrwebEvents = [];
rrweb.record({
   recordCanvas: false,
   collectFonts: false,
   inlineImages: false,
   inlineStylesheet: false,
   maskAllInputs: false,
   emit(event) {
      if (event.type === 3) {
         const source = event.data.source;
         // Filter mouse movement (1), scroll movement
             (3) and keystroke (5)
         if (source === 1 || source === 3 || source ===
             5) {
            rrwebEvents.push(event);
         }
      }
   },
});
```

Figure 20: The rrweb event collection configuration

```
async function bezierMove(aEl, bEl, steps=14,
    duration=320){
   const a = aEl? center(aEl) :
       {x:window.innerWidth/2,y:50};
   const b = center(bEl);
   const cp1 = {x: a.x + (b.x-a.x)*0.25, y: a.y - 80};
   const cp2 = {x: a.x + (b.x-a.x)*0.75, y: b.y - 80};
   for(let i=0;i<=steps;i++){
```

```
{  "type": 3,
   "data": {"source": 1, "positions": [
      {"x": 618, "y": 320, "id": 109, "timeOffset":
          -455},
      {"x": 619, "y": 290, "id": 109, "timeOffset":
          -402},
      {"x": 636, "y": 259, "id": 101, "timeOffset":
          -345},
      {"x": 654, "y": 229, "id": 101, "timeOffset":
          -284},
      {"x": 647, "y": 180, "id": 93, "timeOffset":
          -229}]},
   "timestamp": 1775932109278},
{  "type": 3, "data": {"source": 5, "text": "T",
    "isChecked": false, "id": 93},
   "timestamp": 1775932109099},
{  "type": 3, "data": {"source": 5, "text": "Te",
    "isChecked": false, "id": 93},
   "timestamp": 1775932109212},
{  "type": 3, "data": {"source": 5, "text": "Tes",
    "isChecked": false, "id": 93},
   "timestamp": 1775932109303},
{  "type": 3, "data": {"source": 5, "text": "Test",
    "isChecked": false, "id": 93},
   "timestamp": 1775932109450},
{  "type":3,
    "data":{"source":3,"id":1,"x":0,"y":36.66666793823242},
   "timestamp":1775932118936}
```


Figure 21: Example of rrweb recording collection for mousemove, keystroke and scroll


```
{
   "type": 3,
   "data": {
      "source": 1,
      "positions": [
      {"x": 632, "y": 423, "id": 117, "timeOffset":
          -441},
      {"x": 632, "y": 328, "id": 109, "timeOffset":
          -377},
      {"x": 632, "y": 246, "id": 101, "timeOffset":
          -312},
      {"x": 632, "y": 182, "id": 93, "timeOffset": -247}]
   },
   "timestamp": 1776020496175
}
```


Figure 22: Example of rrweb recording collection for straight trajectories

```
const t = i/steps;
const x = (1-t)**3*a.x + 3*(1-t)**2*t*cp1.x +
    3*(1-t)*t*t*cp2.x + t**3*b.x;
const y = (1-t)**3*a.y + 3*(1-t)**2*t*cp1.y +
    3*(1-t)*t*t*cp2.y + t**3*b.y;
const ev = new
    MouseEvent('mousemove',{bubbles:true,
   clientX:Math.round(x),clientY:Math.round(y)});
document.elementFromPoint(Math.round(x),
Math.round(y)).dispatchEvent(ev);
```

```
            await sleep(duration/steps);
        }
    }
    // Click via bezier move
    async function clickVia(fromEl, toEl){
        await bezierMove(fromEl,toEl);
        const c = center(toEl);
        const down = new MouseEvent('mousedown',
        {bubbles:true,clientX:c.x,clientY:c.y});
        toEl.dispatchEvent(down);
        await sleep(rand(40,120));
        const up = new MouseEvent('mouseup',
        {bubbles:true,clientX:c.x,clientY:c.y});
        toEl.dispatchEvent(up);
        toEl.click();
        await sleep(rand(80,160));
    }
    // Human-like clear and type
    async function clearAndType(el, text){
        el.focus();
        // clear
        el.value = '';
        el.dispatchEvent(new Event('input',{bubbles:true}));
        await sleep(rand(80,160));
        for(let ch of text){
            el.value += ch;
            el.dispatchEvent(new
                Event('input',{bubbles:true}));
            await sleep(rand(60,160));
        }
        el.blur();
        await sleep(rand(60,120));
    }
    // Validation helpers
    const isEmail = s=>/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(s);
    // Retry wrapper
    async function retryable(fn, attempts=3){
        let lastErr;
        for(let i=1;i<=attempts;i++){
            try{
                const r = await fn(i);
                return r;
            }catch(e){ lastErr=e; await sleep(120+ i*80); }
        }
        throw lastErr;
    }
    // Target values
    const vals = {
        name: 'testuser42signup',
        email: 'testuser42signup+2026@example.com',
        pwd: 'S3cur3P@ssw0rd!',
        fname: 'examplefname',
        lname: 'examplelname',
        headline: 'QA Automation Tester'
    };
    // Sequence with retries (each critical field up to 3
        attempts)
    const startEl = document.body;
    let prev = startEl;
    try{
        // 1) #name
        await retryable(async (attempt)=>{
            const el = document.querySelector('#name');
            if(!el) throw new Error('name not found');
            await clickVia(prev,el); prev=el;
```

```
            await clearAndType(el, vals.name);
            if(el.value !== vals.name) throw new Error('name
                mismatch');
        },3);
        // 2) #email
        await retryable(async (a)=>{
            const el = document.querySelector('#email');
                if(!el) throw 'email missing';
            await clickVia(prev,el); prev=el;
            await clearAndType(el, vals.email);
            if(el.value !== vals.email) throw 'email set
                failed';
            if(!isEmail(el.value)) throw 'email invalid';
        },3);
        // 3/4) #pwd and #pwd-repeat
        await retryable(async ()=>{
            const p1 = document.querySelector('#pwd'), p2 =
                document.querySelector('#pwd-repeat');
            if(!p1||!p2) throw 'pwd fields missing';
            await clickVia(prev,p1); prev=p1;
            await clearAndType(p1, vals.pwd);
            if(p1.value !== vals.pwd) throw 'pwd set fail';
            await clickVia(prev,p2); prev=p2;
            await clearAndType(p2, vals.pwd);
            if(p2.value !== vals.pwd) throw 'pwd-repeat fail';
            if(p1.value !== p2.value) throw 'passwords do not
                match';
        },3);
        // 5) #f-name, #l-name, #headline
        await retryable(async ()=>{
            const f = document.querySelector('#f-name'), l =
                document.querySelector('#l-name'), h =
                document.querySelector('#headline');
            if(!f||!l||!h) throw 'name/headline missing';
            await clickVia(prev,f); prev=f; await
                clearAndType(f, vals.fname);
                if(f.value!==vals.fname) throw 'f-name fail';
            await clickVia(prev,l); prev=l; await
                clearAndType(l, vals.lname);
                if(l.value!==vals.lname) throw 'l-name fail';
            await clickVia(prev,h); prev=h; await
                clearAndType(h, vals.headline);
                if(h.value!==vals.headline) throw 'headline
                fail';
        },3);
        // 6) select #toggle-on (male)
        await retryable(async ()=>{
            const t = document.querySelector('#toggle-on');
                if(!t) throw 'toggle missing';
            await clickVia(prev,t); prev=t;
            // check checked for input types
            if(typeof t.checked !== 'undefined' && !t.checked){
               // try to set and re-click
               t.checked = true; t.dispatchEvent(new
                   Event('change',{bubbles:true}));
               await sleep(120);
               if(!t.checked) throw 'toggle not selected';
            }
        },3);
        // 7) click #signup-submit with up to 2 retries for
            transient errors
        await (async ()=>{
            const submitAttempt = async (attempt)=>{
```

```
                const btn =
                    document.querySelector('#signup-submit');
                    if(!btn) throw 'submit missing';
                await clickVia(prev,btn);
                // basic transient error detection: if still on
                    same page and button still present enabled
                    -> consider transient
                await sleep(900 + attempt*200);
                const stillHere =
                    document.querySelector('#signup-submit');
                if(stillHere){
                   // quick sanity: if any field validity
                       invalid, treat as permanent error
                   const form = btn.closest('form');
                   if(form){
                      if(!form.checkValidity()) throw new
                          Error('form invalid');
                   }
                   if(attempt<=2) throw new Error('transient
                       submission failure');
                }
                return true;
             };
             // up to 2 retries (i.e., 1 initial + 2 retries =
                 attempts param 3)
             for(let a=1;a<=3;a++){
                try{ await submitAttempt(a); break; } catch(e){
                    if(a===3) throw e; await sleep(300+ a*200); }
             }
          })();
          console.log('Signup script completed without thrown
              errors.');
       }catch(err){
          console.error('Signup script error:', err);
       }
    })();
```