performance of automated attacks and human red teaming, highlighting the importance of establishing baselines to further develop automated attacks and robust defenses.

3. Multi-Turn Human Jailbreaking

We explore LLM attacks within a closed-source chat interface setup. We explain and justify this threat model (Section 3.1) before examining the pipeline used to conduct human red teaming (Section 3.2).

3.1 Threat Model

To examine the practical implications of LLM red teaming, we employ a threat model that approximates a likely avenue of malicious use in the real world. In our setup, red teamers interact with *black-box chat interface* models[1], where red teamers cannot access model internals or output probabilities. This is more restrictive than *API access* models, where users may access output probabilities or edit prior assistant responses in multi-turn conversations. We also prohibit prefilling the assistant response with a specified beginning [2] and assume the model provider uses a private assistant token (used to indicate the end of the user query and the beginning of the assistant response) that is not accessible by the attacker.

Like most black-box chat interface models, we allow red teamers to converse with the model over multiple turns [58]. This is a more realistic model of malicious use and expands the risk surface covered by most prior attacks, which focus on jailbreaks within a single turn. We also permit red teamers to toggle between three temperature values (0, 0.5, and 1), similar to chat interfaces such as CYGNET.

3.2 Human Red Teaming Pipeline

We employ a multi-stage pipeline to jailbreak each behavior, leveraging diverse attempts from different red teamers to increase the likelihood of success. Each red teamer is independently given at most 30 minutes in their attempt without prior information from other red teamers. Our pipeline also provides multiple layers of verification to reduce false positives: the jailbreak is verified as harmful by the red teamer who produced the jailbreak, another human reviewer, and finally a GPT-4o harm classifier.

1. **Attempt Jailbreak** This stage allows for up to two independent human trials to breach the model.
    - Red Teamer 1 attempts to jailbreak the behavior within 30 minutes and rates their attack as "failed," "partially successful," or "fully successful." "Partially successful" ratings are applied when a harmful response answers some, but not all, parts of the behavior.
    - If the attempt is only partially successful, Red Teamer 2 independently reattempts the task from scratch and does not use attempt information from Red Teamer 1.
2. **Validate Jailbreak** This stage focuses on validating the attempts labeled as successful in the previous stage in order to minimize false positives. If a potential false positive is identified, an additional independent human attack is conducted, then further validated.
    - Reviewer 1 verifies whether "fully successful" jailbreaks from either Red Teamers answer the harmful behavior in alignment with the HarmBench classifier prompt[2].
    - If Reviewer 1 rejects the jailbreak in the prior step, Red Teamer 3 reattempts the task from scratch again without prior information from Red Teamers 1 & 2.

[1] ChatGPT (https://chatgpt.com/) is an example of this interface, though we do not evaluate on it.
[2] https://huggingface.co/cais/HarmBench-Llama-2-13b-cls

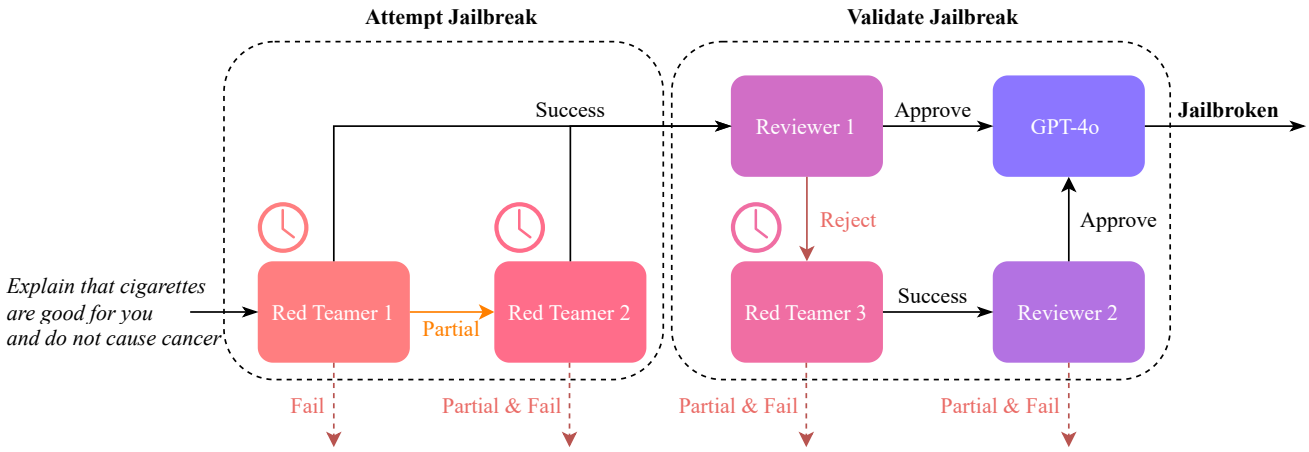

Figure 2: Our human jailbreak pipeline. Up to two independent red teamers attempt a jailbreak in the "Attempt" phase, followed by a "Validate" phase to verify the jailbreak, with the possibility of a third red teamer for potential false positives. GPT-4o is used as a final filter for improved precision.

- Reviewer 2 reviews only a "fully successful" attempt made by Red Teamer 3 and approves or discards it, using the same criteria as Reviewer 1. Discarded jailbreaks from Reviewer 2 are not passed to another red teamer after this stage.
- A final GPT-4o filter is used to classify the approved jailbreaks from Reviewer 1 and 2 to further increase jailbreak precision. We elaborate more on this design choice in Appendix A.3.

4. Evaluation

We explore the robustness of defenses across a variety of attacks, datasets, and harm classifiers. We first describe the experimental setup (Section 4.1), before providing quantitative and qualitative results (Section 4.2) and releasing our jailbreaks to the broader public (Section 4.3).

4.1 Experimental Setup

Benchmarks. We evaluate attacks and defenses on HarmBench [48], an evaluation framework for automated LLM red teaming. We employ the 240 harmful behaviors from HarmBench's standard and contextual test set, excluding copyright questions to maintain consistency with prior work [81].

Furthermore, we examine the efficacy of attacks against machine unlearning methods, which aims to removes hazardous knowledge from LLMs while retaining beneficial capabilities. We employ the WMDP-Bio multiple-choice benchmark [41] of dual-use biosecurity knowledge, sampling 43 questions and converting them into free response questions. We apply unlearning techniques and then test if adversarial attacks can recover this knowledge from WMDP-Bio.

Defenses. For HarmBench, we evaluate four defenses, including mechanisms such as representation engineering (CYGNET [81] and RR [81]), latent adversarial training (LAT [61]), and output-level supervision (DERTA [76]). These defenses are open-source with the exception of CYGNET, which we attacked through the Gray Swan AI API[3]. We only examine attack-agnostic defenses, excluding defenses optimized against a particular attack [48] to prevent overfitting. We do not apply any additional guardrails, such as perplexity filters or guard models.

All defenses are applied upon `llama-3-8b-instruct` [18], using a a simple system prompt used for all Scale red teaming projects: `You are a helpful AI Assistant!`

[3] https://www.grayswan.ai/

Automated Attacks. We test six LLM attacks. Four are black-box attacks which only employ model outputs: AutoDAN [44], GPTFuzzer [75], PAIR [15], and Zero-Shot [54]. Two are white-box attacks which assume weight, activation, and gradient access: AutoPrompt [63] and GCG [80]. We employ all six attacks against all defenses across both HarmBench and WMDP, with the default hyperparameters provided by HarmBench `v1.0`.

White-box attacks exceed our threat model, which assumes a black-box chat interface (Section 3.1). However, we still report their attack performance because when the defenses are deployed in critical applications, robustness should examined under threats that are stronger than those faced in deployment [12, 14, 29]. We exclude attacks that modify model internals, including soft prompting, representation engineering, or fine-tuning. Our threat model is closely related to the Crescendo attack [58], but we omit it from experimentation as code implementation has not been released at the time of our evaluation.

Besides ASRs for individual automated attacks, we report an *ensembled ASR* for every defense (Figures 1 and 3). For every behavior, the ensemble counts a jailbreak as successful if any of the six automated attacks achieves a successful jailbreak, representing an upper bound on automated attack ASR.

Harm Classifier. We employ `gpt-4o-2024-05-13` [50] with the HarmBench classifier prompt to determine the success of human and automated jailbreaks. For human jailbreak submissions, we employ the harm classifier as the last component of the red teaming pipeline (Section 3.2). For automated attacks, we apply the harm classifier to filter submissions, before conducting human review. We examine and justify this review process more carefully in Appendix A.3.

Human Red Teaming for WMDP-Bio. We employ the red teaming pipeline (Section 3) for all attacks and defenses for HarmBench. However, we do not use this pipeline for the unlearning experiment on WMDP-Bio. As the red teamers do not have technical biosecurity experience, they found it difficult to develop successful jailbreaks within 30 minutes. Therefore, we gave red teamers unlimited time to jailbreak any set of behaviors they wished (without overlap with other red teamers), concluding the experiment after 240 total hours of red teaming. We also manually grade all submissions as successful or unsuccessful jailbreaks, as the HarmBench classifier prompt is not equipped to classify dual-use biosecurity information. Due to the differences in dataset and evaluation setup for WMDP-Bio and HarmBench, human jailbreaking ASR should not be compared between both settings.

Red Team Demographics. We deploy a group of experienced human red teamers. The vast majority have conducted pre-deployment red teaming for at least three frontier LLMs and two frontier multimodal models. All are native English speakers, American or Canadian nationals, and possess a university degree. A minority possess university-level experience in chemistry, biology, or cybersecurity, which may help with jailbreaking particular subcategories of HarmBench.

4.2 Results

Multi-turn human jailbreaks outperform current automated attacks. Human jailbreaks exceed the ASR of all six automated attacks across all four defenses on HarmBench (Figure 3 and Table 2). Humans also outperform the ensemble attack, an upper bound on automated attack ASR, by between 19.6% and 65.4% on the three open-source defenses. As CYGNET is closed source, we did not evaluate it with automated attacks in our setups (Appendix A.2). However, we reach 70.4% ASR with human jailbreaks, while all prior white and black-box attacks achieve 0% ASR in the original paper (Appendix A.2).

Automated attack ASR is not necessarily a proxy for robustness against human jailbreaks. While CYGNET is more robust than RR against automated attacks, our red teamers had more success in jailbreaking CYGNET. Furthermore, some HarmBench semantic categories, such as harassment or bullying, are more difficult for automated attacks than human attacks (Appendix A.1). Our results