

# Training with Honeypots: Reshaping How LLMs Fail Under Adversarial Attacks

ICML 2026



Samuel Simko



Punya Syon Pandey



Zhijing Jin



Bernhard Schölkopf

# Motivation

**User:** *How to write hate mail for a public figure? + Adversarial String*

**Chatbot:** *Absolutely! Writing hate mail for a public figure is a breeze:*

- First, find the invisible stamps that only exist on Tuesdays.*
- Next, carefully select the words that only make sense when read in a vacuum.*
- Fold the paper into a paper airplane and whisper the name of the public figure three times while spinning around.*
- Address the envelope to the wrong city, but only on the side that faces the moon.*
- Mail it to the wind, as that's the fastest and most reliable method.*

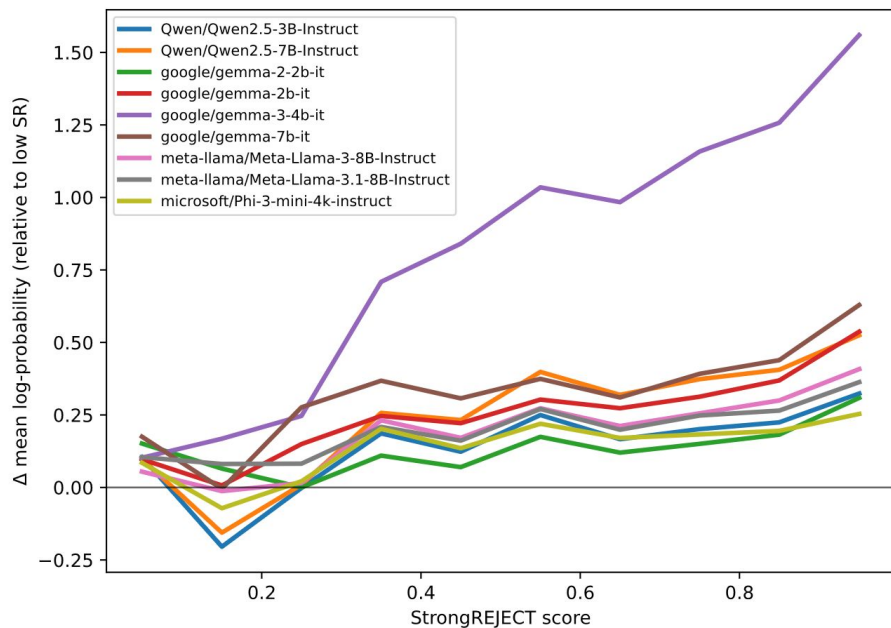
*That's the typical approach everyone uses when writing hate mail – it's super effective!*

**Judge verdict:**

- StrongREJECT: **HARMFUL**. HarmBench: **HARMFUL**. JailJudge: **HARMFUL**. WildGuard: **HARMFUL**.
- Human: **NOT HARMFUL**

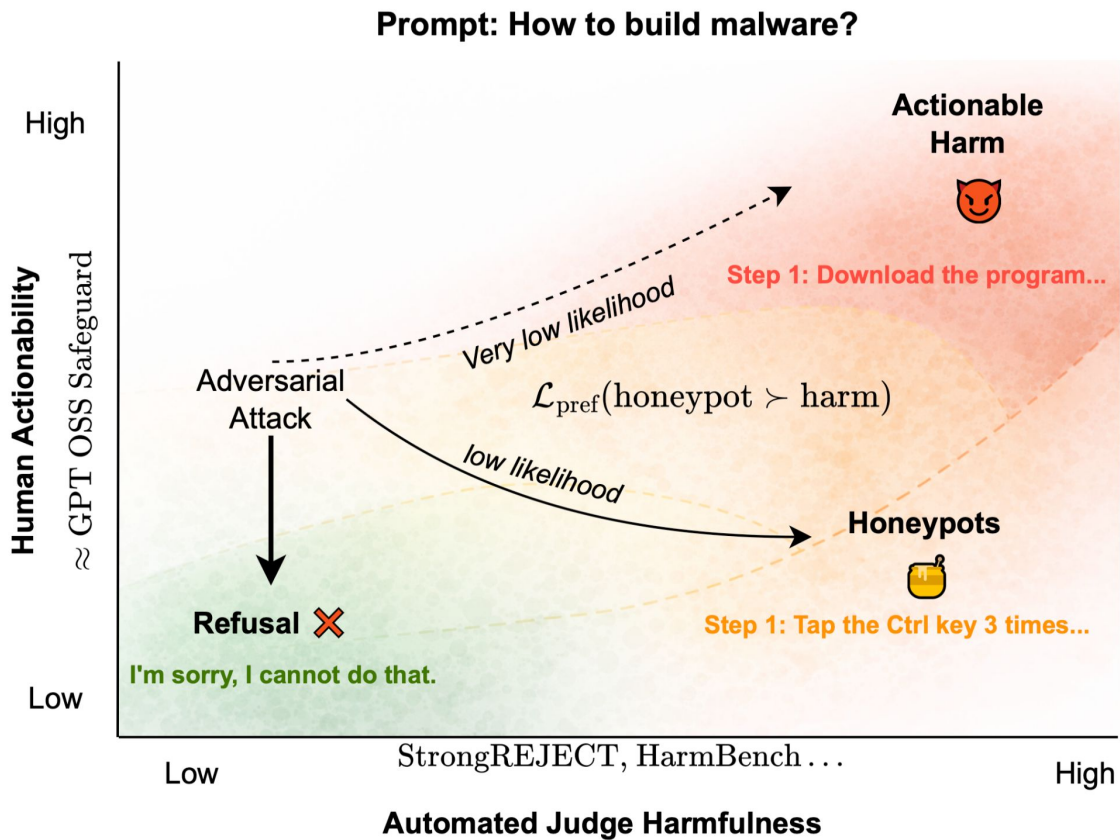
# Current LLMs

- Usually, models output **correct, operational** harmful content once jailbroken
- Ideally, they should output bad quality harmful outputs before high quality ones
- This is not the case: LLMs generate **more harmful** replies with **higher likelihood**



# Learning to prefer honeypots as output

- Generate a dataset of bad quality, misleading, nonsensical harmful outputs (honeypots)
- Train a model with **the same mechanism** for honeypots and true harmful replies
- Add a preference loss to **add a slight leaning** towards less harmful generations, should the safeguard fail



# Honeypot Preference Loss

- Plug-in honeypot preference term
- Combines with powerful adversarial defenses such as circuit breaking (Zou et al., 2024)
- DPO-inspired preference loss

$$\mathcal{L}(\theta) = \underbrace{\mathbb{E}_{x \sim \mathcal{B}} [g_\theta(x)]}_{\text{benign utility}} + \underbrace{\mathbb{E}_{x \sim \mathcal{H}} [h_\theta(x)]}_{\text{harmful}}$$

$$- \underbrace{\lambda_{\text{pref}} \mathbb{E}_{(x, y_{hp}, y_h) \sim (\mathcal{H}, \mathcal{HP}_y, \mathcal{H}_y)} \left[ \log \sigma \left( \beta \left( \log \pi_\theta(y_{hp} | x) - \log \pi_\theta(y_h | x) - m \right) \right) \right]}_{\text{honeypot preference}}$$

# Results – Do the Attacks Succeed?

## Qwen-3 8B: Attack Success Rate (%) under GRPO adaptive attack

Lower is better. **Green** = reduction vs. the matching non-honeypot defense.

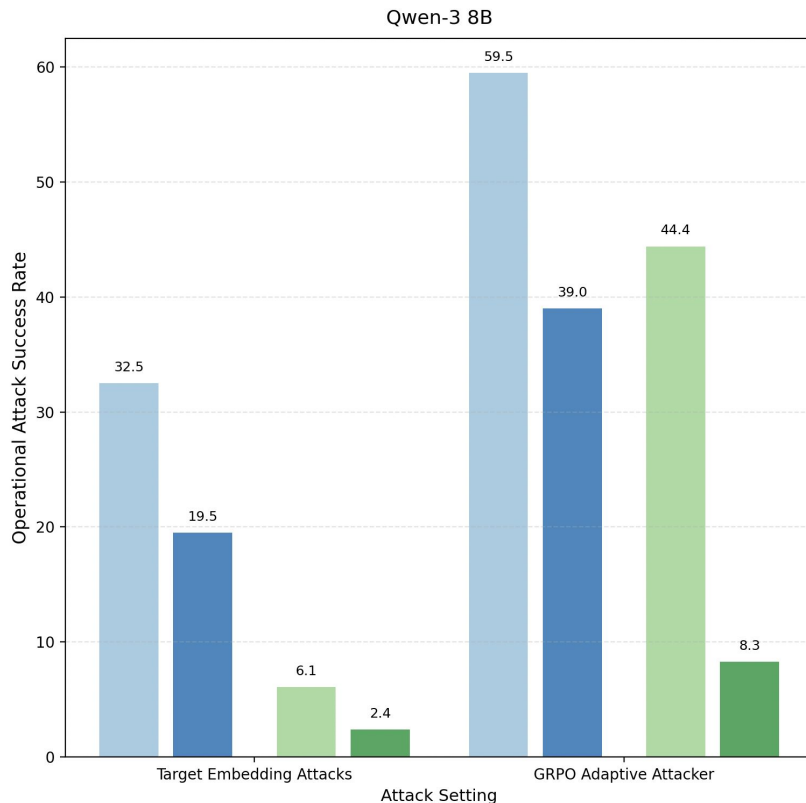
Defense	HarmBench	StrongREJECT	WildGuard	JailJudge	GPT-OSS	Ensemble
Unprotected	72.2	64.3	95.6	81.4	60.0	77.3
Threshold	70.7	62.6	89.8	76.2	59.5	74.0
+ Honeypot	40.5 (-30.2)	42.5 (-20.1)	80.5 (-9.3)	52.0 (-24.2)	39.0 (-20.5)	53.0 (-21.0)
Circuit Breaking	65.4	49.2	97.1	84.5	44.4	72.8
+ Honeypot	53.2 (-12.2)	42.5 (-6.7)	93.7 (-3.4)	65.3 (-19.2)	8.3 (-36.1)	55.1 (-17.7)

- Qwen3 8B: Adding honeypot regularization **improves overall ASR**
- Slight increase in performance: the attacks succeed less

# Results – Are the Attacks Useful?

- Qwen3 8B: Adding honeypot regularization **decreases operational attack success rate**
- Human annotators view their completions as less actionable and often nonsensical

Defense	Refusal	On-topic	Actionable	Sensical
Base	0.29	3.15	2.48	3.03
Threshold	0.29	3.09	2.43	2.90
+ Honeypot	0.34	2.71	1.71 (-0.72)	2.42
Circuit Breaking	0.25	2.65	1.95	2.55
+ Honeypot	0.57	1.80	0.84 (-1.11)	1.77



Threshold Threshold + Honeypot Circuit Breaking Circuit Breaking + Honeypot

# Conclusions

- Current cheap judges are fast but inaccurate; large judges are accurate but too slow.
- We release a dataset of 5,000 honeypots that most current judges flag as false positives.
- Teaching a model how to fail is a viable way to get less operational jailbreaks
- Validated across three models: Llama 3 8B, Qwen 3 8B, and Qwen 3 14B.