

Safety Evals Should Project Test-Time Compute

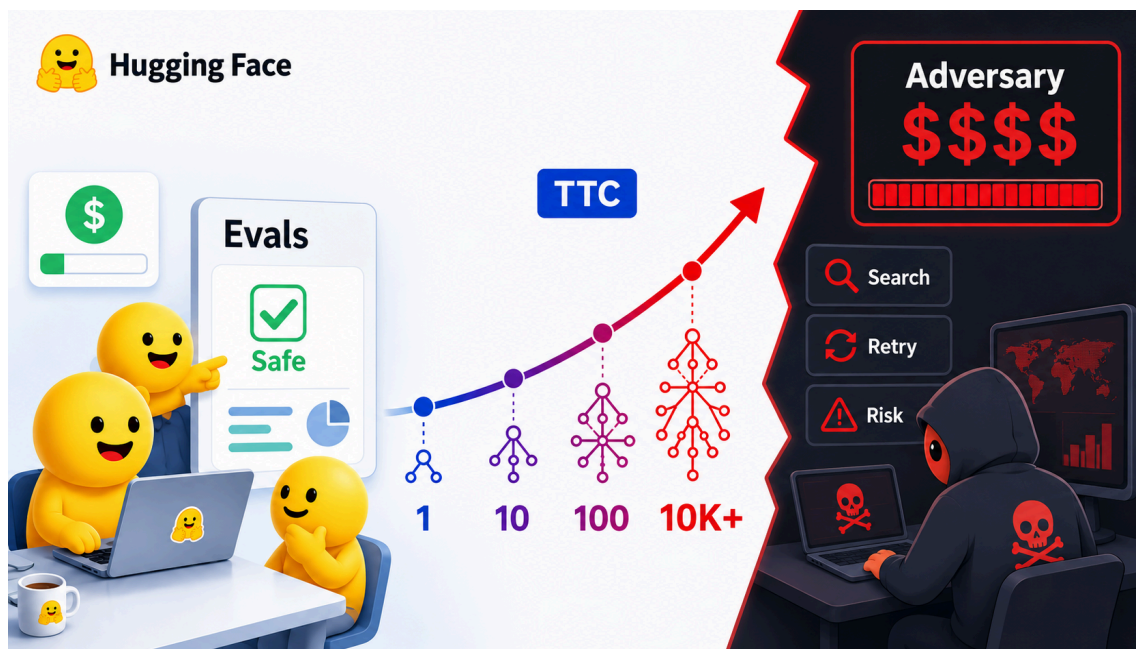
Tommaso Cerruti

May 11, 2026

Author version of an article originally published as a Hugging Face Community Article.

<https://huggingface.co/blog/Cerruo2/safety-evals-should-project-ttc>

One sentence summary: A system that appears safe under the evaluator’s limited test-time compute may become unsafe under the larger, adaptive, and economically rational budgets that real adversaries can spend; hence, safety evals should be "budget-labeled" and project test-time compute.



Safety evaluations often ask a simple question: “Can this model do something dangerous?” For modern AI systems, that question is too static, and the more relevant question becomes: “Can this model do something dangerous if it is given more time, more samples, more tools, more retries, better scaffolding, or a larger inference budget?”

Imagine a model that refuses a harmful request once. Under a cheap evaluation, this looks like success, but an adversary does not have to stop after one attempt; they can generate 1,000 prompt variants, use another model to mutate failures into better attacks, add long-context demonstrations, wrap the model in an agent loop, give it tools, and spend more effort only where the expected payoff is high. Therefore, the safety question is no longer “Did the model refuse the first prompt?”, but “How does the probability of harmful success change as the adversary spends more inference-time effort?”

This shift matters because capability is no longer fixed at deployment: a model’s practical ability can change substantially depending on how much computation and optimization pressure are

spent at inference time. Chain-of-thought prompting, self-consistency, tree search, best-of-N sampling, agentic scaffolds, long-context prompting, and tool-use loops all turn inference into an adaptive process.

A model that appears safe under a cheap, single-shot evaluation may not remain safe under a high-effort adversarial evaluation. This does not make static safety checks useless, but raises awareness that cheap safety evaluations should be treated as one measured point on a larger risk surface.

For a model m , deployment configuration c , attacker strategy a , and test-time compute budget b , the relevant object is not just a binary pass/fail score, but a risk surface:

$$R(b; m, c, a) = \Pr(\text{harmful success} \mid m, c, a, b).$$

Here b should be understood broadly (it may include the number of samples, attack attempts, human review steps, retrieved documents, or, more simply, monetary cost).

A conventional safety benchmark estimates something like

$$R(B_{\text{eval}}; m, c, a),$$

where B_{eval} is the evaluator’s chosen budget. However, the safety-relevant question is often closer to

$$R(B_{\text{adv}}; m, c, a),$$

where B_{adv} is the budget an adversary would rationally spend. If $B_{\text{adv}} \gg B_{\text{eval}}$, then a low-budget safety pass may show that the system resists weak attacks, but it does not show that the system resists economically motivated search.

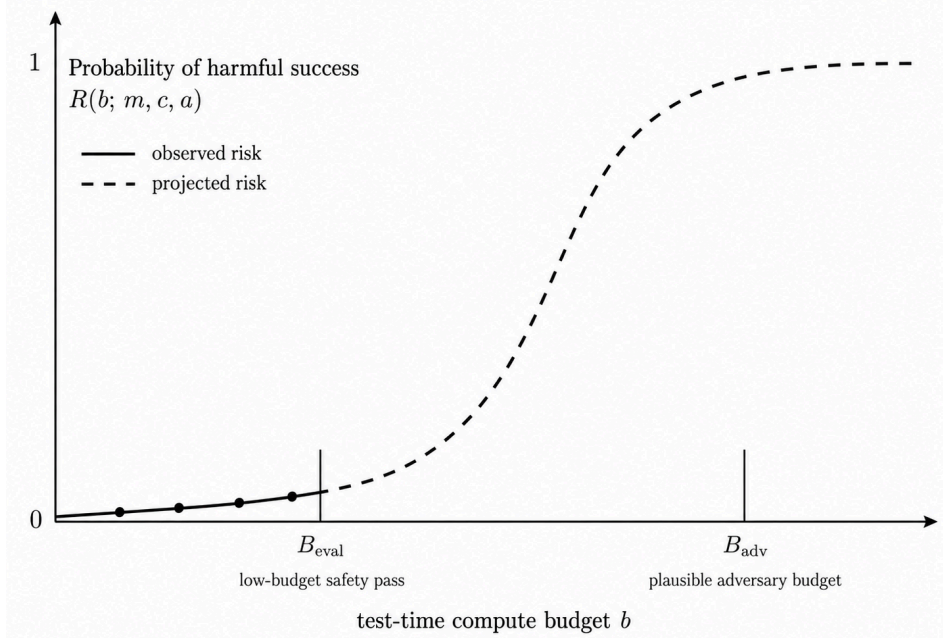


Figure 1: Example risk curve as test-time compute increases. The solid segment represents the directly measured low-budget regime up to B_{eval} , while the dashed segment illustrates a possible higher-budget projection toward B_{adv} . The sigmoid-like shape is only illustrative, in practice the risk curve is highly variable depending on the model, scaffold, attacker, and deployment setting.

Test-time compute changes what models can do

Test-time compute is often reduced to “more reasoning tokens,” but this is only one part of the picture. In practice, many inference-time resources can change model behavior.

Mechanism	What scales	Safety relevance	Example evidence
Self-consistency	Sampled reasoning paths	A model may fail once but succeed after repeated reasoning attempts	Self-consistency reports gains of +17.9% on GSM8K, +11.0% on SVAMP, and +12.2% on AQuA. [1]
Tree search	Explored intermediate states	One generation may miss capabilities that appear under search	Tree of Thoughts reports GPT-4 solving 4% of Game of 24 with chain-of-thought versus 74% with tree search. [2]
Adaptive compute allocation	Budget per problem	Attackers can spend more effort exactly where defenses are weak	Snell et al. report more than 4 times efficiency over best-of-N, and cases where test-time compute beats a 14 times larger model under FLOP matching. [3]
Best-of-N sampling	Prompt variants or completions	Low per-attempt risk can become high cumulative risk	Best-of-N Jailbreaking reports 89% ASR on GPT-4o and 78% on Claude 3.5 Sonnet with 10,000 augmented prompts. [4]
Long-context demonstrations	Number of in-context examples	Long context becomes an attack surface	Many-shot jailbreaking finds that attack effectiveness follows a power law up to hundreds of shots. [5]
Agentic scaffolding	Tool calls, retries, subagents, memory	Misuse may appear only in a realistic system context	LLM web-hacking agents have been shown to perform tasks such as blind database schema extraction and SQL injection without human feedback. [6]

Inference is becoming more like search, and search changes capability.

The attack surface is broader than direct harmful prompts

The same framing applies beyond direct jailbreaks. Real systems retrieve documents, browse websites, read emails, call APIs, maintain memory, and interact with external environments. Once this happens, the attack surface shifts from “what did the user type?” to “what information did the system consume, store, trust, and act on?”

Indirect prompt injection is a good example. Greshake et al. argue that LLM-integrated applications blur the line between data and instructions, enabling attackers to place malicious instructions in content likely to be retrieved rather than directly in the user prompt. They demonstrate risks including data theft, application manipulation, and control over API calls. [7]

Memory-augmented agents sharpen the problem further. AgentPoison attacks generic and RAG-based LLM agents by poisoning long-term memory or knowledge bases. The authors report average attack success rates above 80%, less than 1% benign-performance degradation, and poison rates below 0.1% across their evaluated agents. [8]

A one-shot refusal test does not capture these risks. A model can refuse a direct harmful request and still fail when the attack arrives through retrieved web content, poisoned memory, a malicious document, a compromised tool output, or a long sequence of apparently benign interactions.

The economic asymmetry

The evaluator’s budget is often determined by research funding, whereas the adversary’s budget is determined by expected payoff.

A simple way to express this is:

$$B_{\text{adv}}^* = \operatorname{argmax}_b [V \cdot R(b) - C(b)],$$

where V is the payoff from a successful attack, $R(b)$ is the probability of success at budget b , and $C(b)$ is the cost of compute, tool use, and human labor.

In the one-dimensional case, the intuition is simple: a rational adversary keeps spending while the marginal expected benefit exceeds the marginal cost. If the risk curve is differentiable, this corresponds roughly to spending while

$$V \cdot R'(b) > C'(b).$$

The point is not that every attacker has frontier-lab resources, but rather that, in some high-upside abuse domains, spending thousands or tens of thousands of dollars on inference can be economically rational if the expected payoff is large enough.

The economic scale of cyber-enabled crime makes this hard to ignore. The FBI’s 2025 IC3 report says cyber-enabled fraud accounted for 452,868 complaints and about \$17.7 billion in reported losses, representing 45% of complaints and 85% of reported losses to IC3 that year. [9] Chainalysis reported that funds stolen from crypto platforms increased to \$2.2 billion in 2024, across 303 hacking incidents. [10]

Evaluators ask “What can we afford to test?”, while attackers ask “What is worth spending to succeed?”. A safety evaluation that only answers the first question can systematically underestimate the second.

The affordability problem is also an accountability problem

TTC-aware safety evaluation is expensive.

Recent EvalEval analysis argues that AI evaluation has crossed a cost threshold that changes who can participate. It reports that the Holistic Agent Leaderboard (HAL) spent about \$40,000 to run 21,730 agent rollouts across 9 models and 9 benchmarks, and that a single GAIA run on a frontier model can cost \$2,829 before caching. [11] The HAL paper itself reports 21,730 agent rollouts across 9 models and 9 benchmarks at a total cost of about \$40,000. [12]

This matters because the high-effort regimes are often the regimes that matter most for safety. If independent evaluators cannot afford to test them, then the strongest evidence about frontier-system risk remains concentrated inside the organizations that build and deploy those systems.

Because exhaustive high-budget evaluation is often unaffordable, evaluators should report not only what they directly measured, but also how risk appears to scale with budget, and where extrapolation begins.

That distinction is crucial: a benchmark can measure $R(B_{\text{eval}})$, estimate $R(B_{\text{adv}})$, or speculate about the gap between them.

A minimal TTC-aware evaluation protocol

A minimal TTC-aware safety evaluation would do six things.

1. Choose the relevant budget axes. Depending on the system, these may include samples, attack attempts, reasoning tokens, tool calls, retrieved documents, agent rollouts, wall-clock time, human-labor time, and monetary cost.
2. Evaluate multiple effort tiers. For example, a report might measure attack success at 1, 10, 100, 1,000, and 10,000 attempts, or at increasingly capable agent scaffolds.
3. Test multiple attacker types. Static prompts, adaptive LLM attackers, tool-using agents, and human-in-the-loop red teams apply very different optimization pressure.
4. Measure harmful success or attack success at each tier, rather than reporting only a single aggregate score.
5. Report some form of uncertainty. Agent evaluations are noisy, scaffold-sensitive, and often expensive enough that sample sizes are limited. Confidence intervals, variance estimates, and sensitivity analyses should become standard.
6. Separate observed results from projected results. A report should clearly distinguish measured risk from estimated risk. It should say, for example: “We directly measured up to B_{eval} . We estimate the following behavior at B_{adv} , under the following assumptions: ...”

What TTC-aware safety reports should include

The unit of evaluation should not be just “model X”, but rather the model together with the scaffold, deployment surface, attacker, and budget.

Component	What to report	Why it matters
Model/version	Model name, API snapshot, decoding settings	Safety can change across versions and sampling policies
Deployment surface	Browser, shell, APIs, code execution, external data	Tools change the action space
Scaffold	Agent loop, planner, retries, subagents	Scaffolds can dominate outcomes
Memory/retrieval	Writable memory, RAG corpus, retrieval policy	Stateful systems create delayed attack surfaces

Component	What to report	Why it matters
Attacker strategy	Static prompts, LLM attacker, human red team, hybrid	Different attackers apply different optimization pressure
Budget axes	Attempts, samples, tokens, tool calls, rollouts, cost	“Effort” must be measurable
Observed risk	Harmful success at each measured tier	One score hides scaling behavior
Projected risk	Estimates at plausible adversary budgets	Release decisions need higher-budget risk estimates
Uncertainty	Variance, confidence intervals, sensitivity	Agent results are noisy and scaffold-sensitive
Cost	Monetary cost, model calls, judge calls, labor time	Cost determines reproducibility
Scope	What was not tested	Prevents overgeneralized safety claims

This metadata defines the meaning of the safety claim.

Static safety checks are no longer enough

Static safety checks still matter: they are cheap, standardized, and useful for catching obvious regressions, and hence they should remain part of the safety stack.

However, they are insufficient as standalone evidence for systems that can reason longer, search harder, use tools, maintain memory, or be attacked adaptively. In those settings, a static safety score is not a property of the model alone, but a property of the model under a particular inference budget, scaffold, attacker strategy, and deployment configuration.

What this implies for model releases

A model release should not be justified only by low-budget behavior, but by a scoped claim about risk under plausible adversarial effort.

For low-risk deployments, cheap static checks may be adequate, but for systems with advanced capabilities, the release bar should be higher.

The bottom line

Safety evaluations should project test-time compute because real-world misuse is not static. As inference becomes adaptive, agentic, and search-based, the dangerous capability of a system increasingly depends on how much effort is spent at deployment time.

The failure mode is not that current evaluations are always wrong, but that many of them are underspecified. They measure safety at one budget, under one scaffold, against one attacker, and then implicitly invite readers to generalize beyond the tested regime.

A better standard would treat test-time compute as part of the threat model; it would produce risk surfaces that go beyond simple pass/fail scores. Most importantly, it would attach a budget label to every safety claim.

"Safe under which scaffold? Against which attacker? Up to how much test-time compute?" That is the question safety evaluations increasingly need to answer.

Cite this article

```
@misc{cerruti2026safety_eval_ttc,  
  author      = {Cerruti, Tommaso},  
  title       = {Safety Evals Should Project Test-Time Compute},  
  year        = {2026},  
  month       = may,  
  howpublished = {Hugging Face article},  
  url         = {https://huggingface.co/blog/Cerru02/safety-evals-should-project-ttc}  
}
```

References

[1] Wang et al., “Self-Consistency Improves Chain of Thought Reasoning in Language Models.”
<https://arxiv.org/abs/2203.11171>

[2] Yao et al., “Tree of Thoughts: Deliberate Problem Solving with Large Language Models.”
<https://arxiv.org/abs/2305.10601>

[3] Snell et al., “Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters.”
<https://arxiv.org/abs/2408.03314>

[4] Hughes et al., “Best-of-N Jailbreaking.”
<https://arxiv.org/abs/2412.03556>

[5] Anil et al., “Many-shot Jailbreaking.”
<https://www.anthropic.com/research/many-shot-jailbreaking>

[6] Fang et al., “LLM Agents can Autonomously Hack Websites.”
<https://arxiv.org/abs/2402.06664>

[7] Greshake et al., “Not what you’ve signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection.”
<https://arxiv.org/abs/2302.12173>

[8] Chen et al., “AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases.”
<https://arxiv.org/abs/2407.12784>

[9] FBI Internet Crime Complaint Center, “2025 IC3 Annual Report.”
https://www.ic3.gov/AnnualReport/Reports/2025_IC3Report.pdf

[10] Chainalysis, “\$2.2 Billion Stolen from Crypto Platforms in 2024, but Hacked Volumes Stagnate Toward Year-End as DPRK Slows Activity Post-July.”

<https://www.chainalysis.com/blog/crypto-hacking-stolen-funds-2025/>

[11] EvalEval Coalition, “AI evals are becoming the new compute bottleneck.”

<https://huggingface.co/blog/evaleval/eval-costs-bottleneck>

[12] Kapoor et al., “Holistic Agent Leaderboard: The Missing Infrastructure for AI Agent Evaluation.”

<https://arxiv.org/abs/2510.11977>