

Stop at the Hook

Cheap interpretive judgment before expensive compute.

Alex Estabrook

Concept paper

Abstract

The safest unit of compute is the one never spent. Put a small, legible judgment layer at the threshold of an expensive model. Let it decide what may enter, what must be routed, what should be slowed, and what should stop. The hook is not another answer engine. It is the doorman.

Contents

1. [The problem](#)
2. [The threshold](#)
3. [Pacing](#)
4. [Interpretation](#)
5. [Architecture](#)
6. [Measurement](#)

Compute begins too early.

Most systems treat arrival as permission. An input reaches the service, the large model wakes up, and the expensive work begins. Only then does the system attempt to decide whether the request was useful, malformed, manipulative, repetitive, or outside scope.

That order is backwards. The most capable component is also the costliest place to discover that an input should never have entered. By the time deep inference recognizes a bad handoff, the resource has already been consumed and the attack surface has already opened.

The system needs a boundary that can say *not yet* before the GPU says anything at all.

Arrival is not permission.

The hook is the door.

A hook is the first meaningful point of contact: the moment a system can still inspect an input without committing the full model to it. It is intentionally small, fast, and limited. Its job is not to solve the request. Its job is to judge the handoff.

At the hook, a request can be admitted, clarified, routed to a narrow tool, delayed for inspection, reduced to a cheaper representation, or refused. The decision happens while the cost and the blast radius are still low.

Judge the input, not the story.

The hook does not need a grand theory of the user. It needs observable criteria: scope, repetition, ambiguity, privilege requested, resource demand, instruction conflict, and signs of probing. The more legible the criteria, the easier the threshold is to audit.

The threshold architecture Cheap decision → selective expense

Input Untrusted · unpriced · unsorted

→

The hook Interpret · pace · route · stop

→

Inference Expensive · capable · bounded

Admit

Scoped, ordinary, and ready for inference.

Route

Send narrow work to the smallest capable path.

Stop

Hold probing, manipulation, or unjustified cost.

03 / Pacing

Pacing is control.

Speed is usually treated as an unquestioned good. At a system boundary, speed can also be the absence of judgment. A request that moves instantly

from arrival to full inference leaves no room to notice a pattern developing across time.

Pacing creates that room. It introduces a deliberate interval between detection and expenditure. Repeated requests can be grouped. Escalating scope can be seen as escalation. Slight variations can be recognized as a probe rather than mistaken for unrelated traffic.

The pause does not have to be long. It only has to be long enough for the system to compare this request with the boundary it is meant to protect.

I

Use the cheapest adequate judgment first.

II

Spend capability only after scope is known.

III

Make the threshold easier to inspect than the model behind it.

04 / Interpretation

Taste before compute.

The hard part of a threshold is not syntax. Harmful and wasteful inputs often look ordinary one line at a time. Their meaning appears through timing, sequence, mismatch, and intent. The hook therefore needs limited interpretation, but not unlimited intelligence.

This is taste in the engineering sense: a disciplined preference for the proportionate path. Is the request worth the compute it asks for? Is there a narrower mechanism? Does the request belong here? Is it trying to learn the boundary by repeatedly touching it?

Taste chooses the shape of the work before power performs it. It is the difference between a system that can answer and a system that knows when an answer should begin.

Scope

Is the request within the system's declared purpose and authority?

Proportion

Does the expected value justify the requested compute and exposure?

Sequence

Does this input become suspicious when read beside earlier attempts?

Route

What is the least capable component that can safely do the work?

05 / Adversarial pressure

Make probing expensive.

An open inference path gives an adversary a rich feedback loop. Every attempt earns a capable response, revealing a little more about the system. The attacker spends a prompt; the defender spends an inference.

A hook changes the exchange. Near-duplicate requests can share a decision. Suspicious sequences can lose detail rather than gain it. Unclear escalation can receive a narrow clarification instead of a broad answer. The

system stops rewarding persistence with more compute.

The goal is not to infer a person's hidden character. It is to detect a pattern of interaction and reduce what that pattern can reach.

06 / Measurement

Count what never reached the model.

The hook succeeds when it prevents unjustified inference without obstructing legitimate work. That balance can be measured. The important numbers live at the boundary, not in a claim of perfect safety.

What to measure

Track avoided inference cost, routing accuracy, time added to ordinary requests, repeated-probe detection, false stops, successful clarifications, and the share of work completed by narrower components. Review misses in both directions: dangerous requests admitted and valid requests blocked.

What not to claim

A threshold does not make the model correct. It does not eliminate novel attacks. It does not turn judgment into certainty. It reduces unnecessary exposure, makes allocation deliberate, and creates a legible place to improve the decision.

07 / Implementation

Keep the doorman smaller than the building.

If the hook becomes another large, opaque model, the architecture folds back into the problem it was meant to solve. The front layer should stay cheaper, narrower, and more inspectable than the inference it governs.

Use explicit rules where rules are enough. Use small classifiers where categories are stable. Use short context windows for sequence patterns. Escalate ambiguous cases rather than pretending ambiguity has disappeared. Separate the hook's authority from the model's fluency.

A good threshold can explain which signal changed the route. A good system can test that signal without paying for the entire stack.

Do not ask the furnace whether the door should have
opened.

08 / Conclusion

Stop before the burn.

Capability is easiest to govern before it starts. The hook places judgment at that moment: before the inference, before the feedback loop, before the expensive component becomes responsible for deciding whether it should have been invoked.

This is not a smaller answer. It is a prior question: *Should this enter?*

Judge at the door. Route what belongs elsewhere. Slow what arrives as a pattern. Stop what is probing the boundary. Then—and only then—spend

the compute.

The hook is where taste becomes architecture.

Alex Estabrook

Stop at the Hook