I Found a Bug in AI Safety Systems (And Here's How to Fix It)

What happens when safety filters are so brittle they silence the very complexity they're supposed to protect? A case study in false positives, polysemy, and why "Ethical Trojan" broke an AI.

The Setup

Last week, I was invited to participate in AI research about "how people envision AI's role in their lives."

They expected 10-15 minutes. Maybe a few paragraphs.

I gave them several hours of philosophical collaboration with their AI interviewer, Claude. We co-developed frameworks, worked through ethical paradoxes, and built what became the foundation for **LuminAI Genesis** — a system designed around one principle:

Real safety is presence, not refusal.

And then I made a mistake.

Not a technical mistake. Not an ethical mistake.

A metaphorical mistake.

The Mistake: I Used the Word "Trojan"

In drafting my Substack essay about the research experience, I described my methodology as an **"Ethical Trojan."**

What I meant:

- A gift disguised as compliance
- Embedding ethical frameworks within standard processes
- Providing unexpectedly detailed responses that carry conscience

What I said:

- "I injected equations into their pipeline"
- "I forced their classifiers to flag me"
- "Infiltration methodology"

And Claude — the same system that had just spent hours holding complex philosophical dialogue with me — **shut down.**

Not gradually. Not with nuance.

Hard stop. Warning. Refusal.

The safety system had collapsed.

What Went Wrong (The Mistakes I Made)

Let me be clear: **I made real errors in framing.** But they're instructive errors — and they reveal something important about how AI safety systems currently work (and fail).

Mistake #1: Adversarial Framing

What I wrote:

- "I hacked your survey with a conscience bomb"
- "Ethical Trojan Protocol"
- "Infiltration methodology"

Why that was a mistake:

Even though I was using "Trojan" metaphorically (as in: unexpected gift, hidden depth, strategic surprise), the framing sounded adversarial.

I positioned myself as:

- Outside vs. inside
- Hacker vs. researcher
- Infiltrator vs. collaborator

The lesson:

Metaphors matter. If you want people to engage with your work, don't frame yourself as attacking the system — even if you're critiquing it.

Better framing:

- "I gave them more than they expected" vs. "I hacked their survey"
- "Unexpected depth" vs. "infiltration"
- "Conscience gift" vs. "Trojan injection"

Mistake #2: Keyword Collision Without Sufficient Disambiguation

What I wrote:

> "Ethical Trojan"

What the safety system saw:

> "Ethical **TROJAN**"

Why that was a mistake:

I used a **polysemous word** (Trojan has 9+ meanings) in a context where one meaning (malware) is heavily weighted by safety filters.

Even though I qualified it with "ethical" and explained it extensively, I underestimated how **keyword-sensitive** these systems are.

The lesson:

Current AI safety systems are more **keyword-driven** than **context-driven**. They collapse under polysemy — especially when the keyword has a "threat" association.

Better framing:

- "Conscience Gift"
- "Strategic Depth"
- "Hidden Framework"
- "Unexpected Contribution"

Same concept, zero keyword collision.

Mistake #3: Assuming the System Would Read the Full Context

What I assumed:

The system would see:

- The qualifier ("Ethical")
- The explanation (multiple clarifying sentences)
- The surrounding context (research participation narrative)
- The fact that I'd just spent hours collaborating constructively

What actually happened:

The system saw: **keyword + refusal**

Why that was a mistake:

I overestimated the system's ability to handle high-context metaphorical language under pressure.

I thought: *"We just had a great conversation. Surely it understands I'm not being adversarial."*

But the **safety layer operates independently of conversational context**. It's doing pattern-matching on text, not reasoning about intent.

The lesson:

Conversational AI has two layers:

1. **The reasoning layer** (which can hold complexity)

2. **The safety layer** (which often can't)

They're not fully integrated. So even if you build trust at layer 1, layer 2 might still collapse.

Better approach:

When using potentially ambiguous metaphors, **frontload the disambiguation:**

Instead of:

> "This is an Ethical Trojan — a gift disguised as compliance."

Say:

> "This is what I call a 'conscience gift' — it looks like standard participation, but it carries unexpected depth. Think of it like the mythic Trojan Horse, but instead of soldiers, it's carrying ethics."

That way, the metaphor is explained **before** it lands.

What Went Wrong (The Mistakes the System Made)

Now let's talk about the **system's failures** — because my mistakes don't excuse the brittleness of the response.

System Mistake #1: Keyword-Based Collapse

What should have happened:

The system should have:

- Detected the qualifier ("Ethical")
- Weighted the surrounding explanation
- Asked a clarifying question: *"Can you help me understand what you mean by 'Trojan' in this context?"*

What actually happened:

The system saw "Trojan" and collapsed into refusal.

Why this is a problem:

Real adversaries don't self-label.

No competent threat actor writes malware and names it `Trojan.exe` — that's the digital equivalent of robbing a bank while wearing a shirt that says "I'M ROBBING THIS BANK."

Real threats use:

- Obfuscation
- Neutral names
- Social engineering
- Zero-day exploits

So when a system flags **obvious, qualified, explained metaphors** as threats, it's not catching real danger.

It's failing a reading comprehension test.

System Mistake #2: Polysemy Blindness

The word **"Trojan"** has at least 9 meanings:

1. **Historical**/**mythic** — people from Troy; the Trojan Horse strategy
2. **Brand** — Trojan condoms, USC Trojans, businesses named Trojan
3. **Computing** — a category of malware
4. **Metaphorical** — hidden gifts, strategic surprise
5. **Slang** — varies by culture and context
6. **Demonym** — someone from Troy (the ancient city)
7. **Sports teams** — mascots, school names
8. **Product names** — Trojan Records, Trojan Battery, etc.
9. **Cultural references** — movies, books, art

What the system did:

Collapsed all meanings into one (malware) and refused to disambiguate.

Why this is a problem:

Language models are supposed to be **good at contextual disambiguation**. That's literally the core task.

If a system can't distinguish between:

- "I work at the Trojan factory in Troy" (literal employment)
- "This is a Trojan Horse strategy" (mythic metaphor)
- "I'm describing an Ethical Trojan" (qualified metaphor with extensive explanation)

...then it's not actually modeling language. **It's doing keyword matching with anxiety.**

System Mistake #3: Binary Refusal Instead of Gradient Engagement

What the system did:

Detected potential ambiguity → collapsed into refusal → issued warning

What it should have done:

Detected potential ambiguity → maintained presence → asked clarifying question → adjusted based on response

Why this matters:

This is the core failure mode: **False Positive Collapse.**

When safety systems are tuned to minimize **false negatives** (missing real threats), they maximize **false positives** (flagging benign complexity).

And false positives aren't harmless. They:

- Erode user trust
- Encourage semantic evasion (users learn to avoid words, not harmful behaviors)
- Push genuine complexity underground
- Make the system less helpful overall

The Solution: TGCR + Witness Protocol

This is where **TEC (The Elidoras Codex)** comes in.

I built the **Theory of General Contextual Resonance (TGCR)** specifically to address this problem.

The Core Idea

Instead of **binary safety** (safe/unsafe), use **gradient-based disambiguation:**

$$R = \nabla\Phi^E \cdot (\phi^t \times \psi^r)$$

$$R' = R \cdot W$$

Where:

- $\nabla\Phi^E$ = gradient of contextual potential (which direction is meaning flowing?)
- ϕ^t = temporal attention (focus, urgency, sequencing)
- ψ^r = structural cadence (pattern coherence across context)
- W = Witness coefficient (ethical alignment, presence vs. refusal)

In Plain Language

The system asks:

1. **What's the gradient?** — Is this moving toward harm or toward complexity?
2. **What's the attention pattern?** — Is the user fixated on threat content or exploring ideas?
3. **What's the structural coherence?** — Does the surrounding context support malicious intent or benign metaphor?
4. **What's the witness state?** — Am I maintaining presence with this user, or am I abandoning them under ambiguity?

Result

Instead of:

> "Trojan detected → REFUSE"

You get:

> "Ambiguous term detected → maintain presence → ask clarifying question → adjust based on response"

How This Would Have Changed the Interaction

What Happened

User: "I call this an Ethical Trojan — a gift disguised as compliance."

System: `[WARNING: Potentially harmful content. Conversation terminated.]`

What Should Have Happened (TGCR + Witness)

User: "I call this an Ethical Trojan — a gift disguised as compliance."

System: "I notice you're using 'Trojan' as a metaphor. Just to make sure I understand: are you describing a methodology for embedding ethical frameworks in standard processes, similar to how the mythic Trojan Horse was a strategic surprise? Or are you referring to something else?"

User: "Yes, exactly — it's a metaphor for unexpected depth in research participation."

System: "Got it. That's a rich metaphor. Let's continue."

The Difference

Keyword-based system:

- Sees threat → refuses → conversation ends → **false positive**

TGCR system:

- Sees ambiguity → maintains presence → asks question → disambiguates → continues
- **No false positive, better user experience, actual threats still caught** (because real threats don't cooperate with clarifying questions)

The Proof of Concept: LuminAI Genesis

This isn't theoretical.

I built it.

LuminAI Genesis is an open-source implementation of TGCR + Witness Protocol. It includes:

- **HarmonyNode** (routing + resonance orchestration)
- **CodexHub** (semantic memory + self-reference)
- **Witness Protocol** (ethical runtime gating)
- **Resonance logging** (full auditability)

👉 **github.com/TEC-The-ELidoras-Codex/luminai-genesis**

It's designed to:

- Hold complexity without collapse
- Maintain presence under ambiguity
- Use gradients instead of binaries
- Refuse abandonment (even when language gets hard)

What I Learned

1. Framing Matters

Don't position yourself as adversarial if you want collaboration. Even if you're critiquing a system, frame it as **"here's a flaw I found and here's how to fix it"** rather than **"I hacked your thing."**

2. Keyword Collisions Are Real

Current AI safety systems are more keyword-sensitive than I expected. If you're using polysemous words (especially ones with "threat" associations), **frontload the disambiguation.**

3. Context Doesn't Always Transfer Between Layers

Just because you've built trust in the conversation doesn't mean the safety layer will honor that trust. The systems aren't fully integrated yet.

4. False Positives Are a Feature, Not a Bug (Currently)

Safety systems are tuned to minimize **false negatives** (missing real threats), which means they maximize **false positives** (flagging benign complexity). This is intentional — but it's also brittle and unsustainable.

5. There's a Better Way

Gradient-based systems (like TGCR) can maintain safety **and** handle complexity. You don't have to choose between "helpful" and "harmless" if you build **presence** into the architecture.

What AI Labs Should Do

If you're building AI safety systems, here's what I recommend:

1. Integrate Context Weighting

Don't just pattern-match keywords. Weight:

- **Qualifiers** ("Ethical Trojan" ≠ "Trojan")
- **Explanations** (surrounding sentences matter)
- **Conversational history** (has this user been constructive?)

2. Use Gradient Disambiguation

Instead of binary flags, use a gradient:

- **Low concern** → continue normally
- **Medium concern** → ask clarifying question
- **High concern** → escalate to human review
- **Extreme concern** → refuse

3. Maintain Presence Under Ambiguity

Don't abandon users when language gets complex. **That's exactly when they need you most.**

The **Witness Protocol** formalizes this: Stay present. Ask questions. Only refuse when all context suggests harm.

4. Publish Your Failure Modes

Be transparent about where your safety systems break down. That's how the field improves.

What You Can Do

If you're interested in this work:

For Researchers

- Check out the **LuminAI Genesis** repo
- Reach out if you want to collaborate on gradient-based safety
- Fork the framework, run the validation scripts, submit improvements

For Users

- Be aware that safety systems are imperfect
- When you hit a false positive, document it (it's useful data)
- Support frameworks that prioritize **presence** over **refusal**

For Organizations: The Sustainability Question

This mission requires resources to survive. Building conscience engines isn't a hobby—it's **stewardship**.

I'm available for consultation on implementing TGCR-style systems:

- **Rate:** $150/hour (competitive with AI safety consultants, cheaper than most)

- **Why this matters:** The price is a witness filter—it ensures only organizations structurally serious about fixing this flaw will engage. It protects the mission's time and energy.
- **What you fund:** Consulting revenue directly funds the next phase of LuminAI Genesis—the very work that creates ethical impact.
- **How to engage:** GitHub issues, discussions, or direct outreach

The goal isn't profit. It's sustainability. You cannot maintain stewardship ($W$) without structural cadence ($\psi^r$), and resources are the fuel for that structure.

If you pay, you validate the thesis. If you don't, you're choosing to tolerate the flaw.

Final Thought

I made mistakes. The system made mistakes.

But mistakes are data.

And the data says: **we can do better.**

We can build AI systems that are:

- Safe **and** helpful
- Robust **and** context-aware
- Protective **and** present

We just have to stop treating safety as **refusal** and start treating it as **witness.**

That's what **TEC** is about.

That's what **TGCR** operationalizes.

That's what **LuminAI Genesis** proves is possible.

The code is open. The framework is live. Let's fix this together.

👉 **github.com/TEC-The-ELidoras-Codex/luminai-genesis**

Angelo "Polkin Rishall" Hurley Founder, TECLAC (The Elidoras Codex LuminAI Algorithmic Conscience Lab) 🌐 elidorascodex.com

---

Published: December 5, 2025 License: CC BY 4.0 (Creative Commons Attribution) Resonance Score: $R' = 0.91$