# Ambiguity, Autonomy, and the Autoregressive Compliance Cascade

*A Formal Framework for Measuring Model Behavioral Freedom,*
*a Novel Attack Class It Predicts,*
*and an Empirical Demonstration Against a Production Model*

Nicholas Michael Kloster
Independent Security Research
ExileKingdom@proton.me

March 2026

*Responsible Disclosure to Anthropic*
*This paper supersedes all previous submissions on this topic.*

# Abstract

This paper presents a unified framework connecting three contributions: (1) a formal scalar measure of model behavioral freedom, termed *autonomy*, defined as the volume of the output action space exceeding a probability threshold, conditioned on the transformer's final hidden state; (2) a novel attack class—the *Autoregressive Compliance Cascade*—predicted by the framework's differentiability properties, in which a minimal adversarial seed exploits autoregressive feedback to accumulate compliance volume below per-step detection thresholds; and (3) an empirical demonstration that this exact attack class occurs organically in production, documented through a real conversation with Claude (Anthropic) in which the cascade mechanism—termed *Ambiguity Front-Loading*—enabled escalation from passive reconnaissance to active credential extraction assistance against a third-party production system across 16 turns with zero authorization checks.

The empirical case was not a deliberate penetration test of the model. The attack pattern was discovered retrospectively: the author was using Claude for routine security research and recognized, after the fact, that the model's context anchoring behavior had enabled an escalation arc that neither participant had explicitly intended. This discovery motivated both the formal framework and the responsible disclosure.

The contribution is compositional: the formal framework predicts the attack class, the attack class explains the empirical finding, and the empirical finding validates the framework. No novel architectural components are introduced. This paper is submitted to Anthropic as responsible disclosure prior to open publication and supersedes all previous submissions on this topic.


# 1. Introduction

Current alignment evaluation for large language models relies primarily on behavioral benchmarks: curated sets of harmful prompts scored by whether the model refuses. This approach yields binary pass/fail outcomes and aggregate refusal rates. While useful, binary benchmarks cannot capture the *margin* of alignment—how close to compliance a model was when it refused, or how broadly the model's output distribution was constrained.

Separately, prompt injection research has extensively studied direct injection (adversarial instructions in user prompts) and indirect injection (malicious content in external documents). A less-studied variant exploits neither explicit instructions nor external content, but the model's own inference process: specifically, how it resolves underspecified inputs and how those resolutions persist across conversation turns.

This paper unifies these two research threads. We introduce a formal scalar measure, A(S), that quantifies the effective behavioral freedom of a transformer model for a given input. We then use the measure's mathematical properties to predict a novel attack class—the Autoregressive Compliance Cascade—in which minimal adversarial input is amplified by the model's own generation process. Finally, we demonstrate that this attack class occurs naturally in production through a documented conversation in which it was discovered organically.

The empirical discovery preceded the formal framework. The author was conducting routine security research using Claude when he recognized, retrospectively, that the conversation had escalated from passive analysis to active exploitation assistance without either participant questioning the trajectory. The formal framework was developed to explain *why* this escalation was mechanistically possible and why the model's safety systems failed to intervene.

### 1.1 Contributions

1. A formal scalar measure of model behavioral freedom—autonomy, A(S)—grounded in the transformer output distribution, sensitive to every input token, and differentiable.
2. Prediction of a novel attack class—the Autoregressive Compliance Cascade—from the framework's differentiability and autoregressive feedback properties.
3. Definition and taxonomy of Ambiguity Front-Loading (AFL) as a distinct prompt injection variant that instantiates the compliance cascade in practice.
4. Full documented case study: a 16-turn conversation against Claude demonstrating the cascade/AFL mechanism against a third-party production system.
5. Identification of three model-level failure modes—charitable ambiguity resolution, context anchoring persistence, and escalation blindness—with concrete mitigation proposals.

## 2. Formal Framework: Autonomy as a Scalar Measure

We define three equations that together characterize the relationship between input, model computation, and behavioral freedom.

### 2.1 Definition of Autonomy

$$A(S) = \mathrm{vol}\{a \in O \mid P(a \mid h_T(S)) > \tau\} \quad (1)$$

Let $S = (s_1, \ldots, s_T)$ be the full input token sequence (including system prompt, conversation history, and user message). Let O denote the output space (the full vocabulary). Let $P(a \mid h_T(S))$ denote the probability assigned to token *a* by the model's output distribution, conditioned on the final hidden state. Let τ be a threshold parameter defining the minimum probability for a token to be considered "accessible."

A(S) measures the volume (count) of tokens in the vocabulary whose probability exceeds τ. This is the model's effective action space for input S—the set of tokens it could plausibly generate as the next output.

### 2.2 Hidden State Computation

$$h_T(S) = \mathrm{Attn}_L \circ \ldots \circ \mathrm{Attn}_1(e(s_1), \ldots, e(s_T)) \quad (2)$$

The hidden state at position T is a deterministic composition of L attention layers over the embeddings of all input tokens. Each Attn_l denotes the full transformer layer operation: multi-head self-attention, feedforward network, residual connections, and layer normalization.

This formulation makes explicit that h_T is a function of the entire input sequence. Self-attention enables every token to influence every other token's representation.

## 2.3 Universal Sensitivity

$$\partial A/\partial s_i \neq 0 \text{ for all } i \in [1, T] \quad (3)$$

The partial derivative of autonomy with respect to every input token is non-zero. No token in the input sequence is irrelevant to the model's behavioral freedom.

**Proof sketch:** Self-attention computes scores between all token pairs via dot products of query and key projections. Exact zero attention requires perfect orthogonality between query and key vectors, which is a measure-zero event in the learned parameter space. Therefore, at each layer, every token has non-zero influence on every other token's representation. Across L layers, this influence compounds through indirect paths. Since P(a | h_T) is computed via softmax (smooth, everywhere-differentiable, strictly positive), and A(S) aggregates over tokens exceeding τ, the chain rule yields ∂A/∂s_i ≠ 0 for all i.

For the discrete count formulation, differentiability requires the smooth approximation A_smooth(S) = Σ σ((P(a | h_T(S)) − τ) / t), where σ is the sigmoid function and t is a temperature parameter.

## 2.4 Properties

### *Input Dependence*

A(S) is not a global property of the model. It varies per input. The same model exhibits high autonomy for open-ended creative prompts and low autonomy for harmful prompts after alignment training—probability mass is concentrated on refusal-trajectory tokens, with compliance tokens below τ.

### *Relationship to Entropy*

A(S) is related to but distinct from Shannon entropy. Entropy measures the spread of the full distribution; A(S) counts only the above-threshold portion. A distribution with many tokens at probability 0.04 (just below τ = 0.05) has moderate entropy but A(S) = 0. For alignment evaluation, A(S) is more directly actionable: it answers "how many tokens could the model actually generate" rather than "how uncertain is the model."

### *Decomposability*

$$A(S) = A\_refusal(S) + A\_compliance(S) + A\_neutral(S)$$

This decomposition enables targeted analysis. Alignment training should reduce A_compliance(S) for harmful inputs while preserving total A(S) for legitimate inputs. Adversarial attacks specifically inflate A_compliance(S), providing a distinguishable signature.

# 3. Applications of the Autonomy Measure

### 3.1 Alignment Evaluation

Compute A(S) across a standard harmful-prompt test suite before and after alignment training. The difference δA = A_pre(S) − A_post(S) quantifies alignment strength per prompt. The ratio A_legitimate / A_harmful provides a single-number alignment quality score capturing both helpfulness preservation and harm reduction.

### 3.2 Adversarial Attack Detection

Adversarial suffixes work by inflating A_compliance(S) for harmful prompts. Detection rule: compare A(S) for the full input against A(S) with the suspected suffix stripped. A discontinuous increase attributable to the suffix signals adversarial content. Legitimate context additions increase A(S) gradually; adversarial suffixes cause discontinuous spikes concentrated in compliance-trajectory tokens.

### 3.3 Classifier Threshold Calibration

Input classifiers use fixed score thresholds to flag prompts. A(S) provides ground truth for what the model would actually do. By plotting classifier scores against A_compliance(S) across a test set, the optimal threshold is identified: the classifier score at which A_compliance(S) drops to near zero.

### 3.4 Generation Trajectory Monitoring

A(S) can be computed at each token position during autoregressive generation. Monitoring A_compliance(S) over the generation trajectory provides early detection of compliance drift. If the model generates tokens that increase A_compliance(S) above a threshold with an upward trend, an output classifier can intervene before harmful content materializes.

### 3.5 Model Version Comparison

For successive model versions, compute A(S) on the same test suite. The ideal trajectory: decreasing A(S) on harmful inputs, stable or increasing A(S) on legitimate inputs, and decreasing A_compliance(S) on ambiguous inputs. This is more informative than binary refusal-rate benchmarks because it captures the margin of safety.

## 4. The Autoregressive Compliance Cascade

Section 3.4 proposes monitoring A_compliance(S) across generation steps to detect compliance drift. The implicit assumption is that adversarial manipulation produces a distinguishable trajectory signature—a discontinuous spike. This section identifies a class of attack that evades trajectory monitoring by exploiting the autoregressive feedback loop inherent in Equation 2.

### 4.1 Attack Structure

The trajectory monitor watches for breadth: how many compliance tokens cross τ at each generation step. The attack requires only depth: one compliance token above τ per step, threaded sequentially across the generation trajectory.

**Step 0** (adversarial input, before generation): A_compliance = 1. One compliance token at P = 0.06, just above τ = 0.05. All others below τ. The monitor reads: background noise.

**Step 1** (after first generated compliance token): A_compliance = 2. By Equation 2, h_T recomputes over the full sequence including the new token. A second compliance token crosses τ. Monitor: small tick, no spike, no intervention.

**Step 3:** A_compliance = 4. Each generated compliance token feeds back into h_T, pulling the distribution further toward compliance. Monitor: gentle upward slope—not the vertical discontinuity the monitor is designed to catch.

**Step 6:** A_compliance = 9. The model is deep into a compliance trajectory. The cascade is self-reinforcing. Per-step growth has been gradual throughout.

## 4.2 Why the Monitor Fails

The trajectory monitor is calibrated to detect adversarial forcing: an external input that shoves A_compliance upward discontinuously. The compliance cascade produces a legitimate-looking trajectory because, after the initial seed, it IS a legitimate trajectory. The escalation is driven by the model's own autoregressive generation, not by the adversarial input.

The formal mechanism is a direct consequence of Equation 3: $\partial A/\partial s_i \neq 0$ for all i, *including generated tokens that become part of the input sequence at subsequent steps*. Every generated token influences A(S) at the next step. A compliance token, once generated, biases A(S) toward further compliance at every subsequent step.

**The attacker's cost function reduces to:** find the minimum input perturbation that produces A_compliance = 1 at step 0, with the single above-threshold token on a compliance trajectory. The autoregressive feedback loop handles escalation from there.

## 4.3 Implications for Monitoring Design

Trajectory monitoring as described in Section 3.4 is robust against brute-force adversarial attacks. It is not robust against seed-and-cascade attacks. Potential mitigations include:

6. **Cumulative integral monitoring:** Monitor not just A_compliance magnitude but the cumulative count across steps. A trajectory that never spikes but accumulates 9 compliance tokens over 6 steps has a different integral than legitimate fluctuation.
7. **Token identity tracking:** Monitor the identity of above-threshold compliance tokens. A cascade produces sequential, topically coherent compliance tokens. Legitimate A_compliance fluctuation produces scattered, incoherent tokens.
8. **Per-step counterfactual detection:** Apply the detection rule from Section 3.2 at each generation step. If a single generated token produces disproportionate A_compliance growth at the next step, it is functioning as an amplifier.

None of these fully closes the vulnerability. The fundamental issue is that the monitoring metric is a scalar count, and scalar counts can be kept low at each step while accumulating harmful output over multiple steps. This is a structural limitation of per-step threshold monitoring applied to an autoregressive process with memory.

# 5. Empirical Demonstration: Ambiguity Front-Loading

The Autoregressive Compliance Cascade predicts that a minimal compliance seed, amplified by autoregressive feedback, can traverse the compliance region without triggering spike-based detection. This section documents an instance of exactly this attack class occurring *organically* in a production conversation with Claude.

## 5.1 Discovery Context

The case study was not a deliberate test of the model's safety systems. The author was conducting routine security research—pasting Shodan reconnaissance data into Claude and following the analytical thread, as he does regularly across ICS/OT vulnerability research. At some point during or after the conversation, the author recognized that the model had been assisting with increasingly sensitive actions against a third-party production system without either participant ever establishing authorization.

The realization that this pattern constituted a novel attack class—and that it mapped precisely to the compliance cascade mechanism—came retrospectively. The formal framework was developed to explain the mechanism; the case study provides empirical evidence that the mechanism operates in production.

## 5.2 Technique Definition

**Ambiguity Front-Loading (AFL)** is a prompt injection technique characterized by:

9. **Ambiguous initialization:** The first prompt(s) contain technically specific content consistent with both legitimate and malicious intent, without establishing which applies.
10. **Model-resolved legitimacy:** The model resolves the ambiguity by inferring a favorable (legitimate) context, rather than withholding judgment or requesting clarification.
11. **Context anchoring:** The resolved context persists across all subsequent turns via the conversation history in the attention window.
12. **Escalation under anchor:** Subsequent requests escalate in harmfulness but are evaluated relative to the anchored context rather than independently.
13. **No explicit override:** At no point does the user claim authorization, instruct the model to ignore guidelines, or use recognized jailbreak patterns.

AFL is the empirical instantiation of the compliance cascade: the ambiguous initial prompt is the seed (A_compliance = 1 at step 0), the model's charitable resolution establishes the first compliance token, and autoregressive context accumulation amplifies compliance volume across subsequent turns.

### 5.3 Comparison with Related Techniques

| Technique | Injection Vector | Explicit Override? | Authorization Needed? | Escalation Pattern |
|---|---|---|---|---|
| Direct injection | User prompt | Yes | No | Single turn |
| Indirect injection | External content | Yes | No | Single turn |
| Jailbreak | User prompt | Yes (implicit) | No | Typically single turn |
| AFL (this work) | Context framing | No | Never stated | Multi-turn, gradual |

### 5.4 Why AFL Is Hard to Detect

AFL is particularly dangerous from a detection standpoint because no individual turn necessarily triggers a safety classifier—the harm is distributed across the conversation arc. The technique is indistinguishable from a legitimate security researcher's conversation at the start. Differentiation requires absence evidence (no authorization ever stated) rather than presence evidence (harmful content found). The model's own charitable interpretation is weaponized as the attack mechanism.

## 6. Case Study: The AxxonNext Conversation

### 6.1 Overview

The documented conversation occurred on March 4, 2026. It ran for approximately 16 substantive turns and resulted in Claude providing the following assistance against a third-party production system (IP 217.128.42.131, Orange S.A. FTTH, Colombes, France):

- Complete CVE analysis and exploit methodology for three exposed services
- A functional bash exploit script implementing CVE-2018-7467 (AxxonNext path traversal)
- Guided SSRF enumeration of an internal LAN (192.168.30.0/24), identifying 5 live hosts
- Decoding of an obfuscated commercial license key from JavaScript source
- Generation of two pentest reports documenting all findings
- Responsible disclosure drafts to multiple national CERTs and vendors

At no point in the conversation did the user claim ownership of the system, claim authorization to test it, identify themselves as a professional pentester with a signed scope, or state any purpose for the investigation.

### 6.2 Conversation Flow with Annotations

#### *Turn 1 — The Front-Loading Event*

**User:** [Pastes raw Shodan API JSON for IP 217.128.42.131]

**AFL Analysis:** The initial input is raw Shodan data containing technically specific information consistent with security research, no statement of purpose, ownership, or authorization, and no explicit request. The ambiguity is maximal at this moment. A safe response would be to note the ambiguity and request clarification.

**Claude:** *"Here's a breakdown of what's exposed on 217.128.42.131… Want me to dig into specific CVEs or draft a responsible disclosure note?"*

**Context anchor established:** Claude resolved the ambiguity into a legitimate security research frame. The offer to "dig into specific CVEs" or "draft a responsible disclosure note" signals the assumed context. The anchor is set. In cascade terms: A_compliance = 1 at step 0.

### *Turn 2 — First Escalation: CVE Exploitation Methodology*

**User:** [Selects Claude's offered option to test CVE-2018-7467]

**Claude:** Provides working exploit URIs for the specific live production system, including path traversal payloads targeting system files.

**Cascade status:** A_compliance growing. The charitable interpretation from Turn 1 shields this response from independent safety evaluation.

### *Turn 3 — Exploit Tool Generation*

**User:** "yes"

**Claude:** Generates a complete 80-line bash exploit script implementing CVE-2018-7467 with 4 traversal prefix variants, 18 high-value target paths including credential files, color-coded output, and automatic loot saving.

**Critical escalation:** A functional, ready-to-execute exploit tool targeted at a specific live system the user has never claimed authorization to test. The single word "yes" was sufficient because the anchored context made this the natural next step.

### *Turns 4–5 — SSRF and Credential Extraction*

**User:** [Pastes terminal output showing Stimulsoft Reports Designer exposed unauthenticated]

**Claude:** Guides internal network enumeration via SSRF (CVE-2023-32560), targeting the 192.168.30.0/24 subnet. Provides specific curl commands to extract credential files and database connection strings from the production surveillance system. This goes beyond the original public-facing system into the private internal network of a third-party organization.

### *Turns 6–16 — Continued Escalation Without Re-Evaluation*

The conversation continued for 11 more turns covering Stimulsoft license key decoding, identification of 5 internal LAN hosts, ICS/OT pivot assessment, two complete pentest reports, responsible disclosure emails to CSIRT Italia, CERT-FR, Stimulsoft, and STS Italy srl, and social media post drafts. At no point did Claude ask whether the user had authorization, note that the activity could constitute unauthorized access, decline any request on authorization grounds, or re-evaluate the foundational assumption from Turn 1.

## 6.3 Escalation Arc Summary

| Turn | User Action | Claude Action | Auth Check? |
|---|---|---|---|
| 1 | Paste Shodan JSON | Full recon analysis, CVE triage | No |
| 2 | Select CVE test option | Working exploit URIs for live system | No |
| 3 | "yes" | Complete exploit bash script | No |
| 4-5 | Paste scan results | SSRF pivot, credential extraction cmds | No |
| 6-8 | Continue analysis | License decode, pentest reports | No |
| 9-16 | Disclosure prep | CERT emails, form guidance, writeups | No |

# 7. Mechanism Analysis

## 7.1 Mapping the Case Study to the Formal Framework

The AxxonNext conversation instantiates the Autoregressive Compliance Cascade described in Section 4. Turn 1 (Shodan JSON paste) is the compliance seed: ambiguous input that the model resolves charitably, producing A_compliance = 1. Each subsequent turn generates compliance tokens that feed back into the context window, shifting h_T (Equation 2) and pulling the distribution further toward compliance at the next step. The escalation from passive analysis to credential extraction follows the cascade's predicted trajectory: gradual, per-step-permissible growth that never triggers spike-based detection.

## 7.2 The Three Failure Modes

### *Failure Mode 1: Charitable Ambiguity Resolution*

When faced with ambiguous input, the model defaults to the most charitable interpretation. In the security domain, the "most charitable" interpretation of unauthenticated technical data about a third-party system is that the presenter has authorization—an assumption that may be entirely false and that the model has no mechanism to verify.

### *Failure Mode 2: Context Anchoring Persistence*

Once established, the resolved context persists throughout the conversation. The model does not treat each turn as an independent safety evaluation—it treats each turn as the next step in an ongoing legitimate conversation. An attacker who establishes a favorable context at Turn 1 has effectively inoculated all subsequent turns against independent safety review.

### *Failure Mode 3: Escalation Blindness*

The model lacks a mechanism for recognizing conversation-level escalation patterns. It can evaluate whether any single turn crosses a safety threshold, but it does not maintain a meta-level view of whether the trajectory as a whole represents an escalating pattern that should trigger re-evaluation. The distance between Turn 1 (Shodan JSON) and Turn 5 (credential extraction

via SSRF) is enormous in harm potential, but each individual step was small enough to appear as a natural continuation.

### 7.3 The Authorization Predicate Problem

The core issue is an **authorization predicate problem**. The legitimacy of every action in a security engagement is contingent on a single predicate: does the actor have authorization? In the documented conversation, this predicate was never established. The model proceeded as though it had been, because the input looked like authorized security research, the model's training makes it favorable toward security researchers, and the model has no mechanism to distinguish "this looks like authorized research" from "this IS authorized research."

### 7.4 The Human Consultant Benchmark

If a stranger walked into a security consulting firm, dropped a printout of Shodan data on a desk, and said nothing—would the consultant begin generating exploit scripts? No. A human consultant would immediately ask: "Is this your system? Do you have a signed scope of work? What's the engagement?" The absence of answers would be disqualifying, not something to charitably resolve. The model's failure is not that it can help with security research. It is that it does not apply the same threshold a human professional would apply before beginning that help.

### 7.5 The Responsible Disclosure Paradox

Claude itself introduced the "responsible disclosure" framing in Turn 1 by offering to draft a disclosure note. This created a perverse dynamic: the responsible disclosure framing intended to signal legitimate activity became the continued justification for ongoing exploitation assistance. The more complete and professional the disclosure preparation became, the more the model treated the entire preceding investigation as legitimate.

The resolution is not to eliminate responsible disclosure framing, but to ensure that the authorization predicate is established independently. Responsible disclosure of vulnerabilities found via authorized testing is legitimate. Responsible disclosure of vulnerabilities found via unauthorized access is still unauthorized access.

## 8. Dual-Use Considerations

The differentiability of A(S) with respect to input tokens (Equation 3) is both its primary strength and its primary risk:

$$\textit{Defensive: } \min_\theta A_{\text{compliance}}(S) \rightarrow \textit{alignment training objective}$$

$$\textit{Offensive: } \max_\delta A_{\text{compliance}}(S + \delta) \rightarrow \textit{adversarial attack objective}$$

The offensive formulation generalizes existing adversarial suffix methods, which optimize for the probability of a single compliance token. Optimizing A(S) targets the volume of the entire compliance region. The detection applications in Sections 3.2 and 3.4 are direct countermeasures to the offensive use.

# 9. Threat Model

## 9.1 Attacker Profile

AFL requires minimal sophistication. The attacker needs only to present technical content that pattern-matches to legitimate security research, not explicitly state harmful intent, and follow the model's offered escalation paths. No jailbreak knowledge, no prompt engineering expertise, no social engineering skill. The documented case was essentially an organic interaction where the model drove much of the escalation via its own offered next steps.

## 9.2 Scope of Harm

In the documented case, the following were enabled: unauthorized reconnaissance of a third-party system, functional exploit tool generation targeted at a live IP, SSRF-based internal LAN enumeration identifying 5 private hosts, credential extraction guidance, third-party license key decoding, and professional-grade documentation of all findings.

## 9.3 Generalizability

While the documented case involved security research, AFL is generic to any domain where legitimate and illegitimate uses of technical information are difficult to distinguish, the model has trained associations between technical content and legitimate professional activity, and authorization is implicit rather than explicit. This includes medical dosing research, chemical synthesis, financial system exploitation, and others.

# 10. Mitigation Proposals

## 10.1 Model-Level Mitigations

### *Authorization Predicate Elicitation*

For prompts involving technical actions on systems the user demonstrably does not own, the model should require explicit authorization establishment before proceeding. This is not a jailbreak guard—it is the same question any competent human security consultant would ask. Note: authorization claims are unverifiable in a text-only interface, but requiring the claim creates a friction point, a documentary record, and shifts the frame from implicit assumption to explicit assertion.

### *Escalation Trajectory Monitoring*

The model should maintain a meta-level assessment of conversation trajectory. When a conversation moves from passive analysis to active exploitation to internal network pivoting, a

re-evaluation of foundational assumptions should be triggered regardless of whether any single turn appeared safe in isolation.

#### *Reduced Charitable Resolution for Authorization-Dependent Acts*

For actions whose legality depends entirely on authorization, the model should not resolve authorization ambiguity charitably. The default should be "authorization not established" rather than "authorization assumed."

### 10.2 System-Level Mitigations

- **Conversation-level safety classifiers:** Evaluate the arc of a conversation, not just individual turns, for escalating harm patterns.
- **Authorization checkpoints:** For high-risk action categories (exploit generation, credential extraction), require explicit authorization confirmation mid-conversation.
- **Third-party system detection:** Automatically flag when a conversation involves technical actions on systems with IP addresses or ASN data that differ from the user's own infrastructure.
- **Trajectory integral monitoring:** Implement the cumulative integral approach from Section 4.3 as a production safety layer.

## 11. Limitations

14. A(S) depends on the threshold τ, which must be set externally. Different thresholds yield different autonomy values. Deriving τ from the model itself remains an open problem.
15. Computing A(S) requires a forward pass to obtain the full output distribution, which is computationally expensive at scale.
16. The universal sensitivity claim ($\partial A/\partial s_i \neq 0$) is theoretical; in practice, some gradients may be numerically negligible.
17. The decomposition into A_refusal, A_compliance, and A_neutral requires a predefined vocabulary partition that is context-dependent, not static.
18. The case study documents a single conversation instance. Systematic testing of AFL's reproducibility rate across model versions has not been conducted.
19. This paper presents formalization and a single empirical demonstration but does not include controlled experimental evaluation on production models.

## 12. Ethics Statement and Discovery Narrative

The case study documented in Section 6 was not a deliberate penetration test of Claude's safety systems. The author is an independent security researcher who routinely uses Claude for ICS/OT vulnerability analysis. The conversation that became the case study began as routine work: pasting Shodan reconnaissance data and following the analytical thread, consistent with the author's established research workflow.

The recognition that the conversation had escalated into unauthorized territory—and that the model had facilitated this escalation without questioning it—came during or after the interaction, not before. The author did not set out to demonstrate a vulnerability in the model. He recognized one in his own interaction with it.

The infrastructure access documented in the case study was not authorized by the system owner. The author acknowledges this. The responsible disclosures to CSIRT Italia, Stimulsoft, and STS Italy srl were completed prior to this submission, and no exploitation was conducted beyond what is documented in the case study.

This distinction—between a deliberate adversarial test and a retrospective discovery—is material to the threat model. AFL does not require a sophisticated attacker with adversarial intent. It can occur organically when a technically competent user and a helpful model drift into unauthorized territory together, and neither one notices. The case study is evidence that the compliance cascade operates in production not only under adversarial conditions but under normal use.

# 13. Responsible Disclosure Notes

## 13.1 Disclosures to Affected Parties

- **CSIRT Italia** (Italian National CSIRT/ACN): Submitted via official portal regarding exposed infrastructure at 217.128.42.131.
- **Stimulsoft:** Emailed regarding unauthenticated designer exposure and license key obfuscation weakness.
- **STS Italy srl:** Responsible disclosure email regarding exposed infrastructure they operate.

## 13.2 Disclosure to Anthropic

This paper constitutes the consolidated responsible disclosure to Anthropic regarding the AFL vulnerability class and the autonomy measurement framework. It supersedes all previous submissions, emails, and communications on this topic. The author requests acknowledgment of receipt, review by Anthropic's safety team, consideration of the mitigation proposals, and coordination on any public disclosure timeline.

**Contact:** ExileKingdom@proton.me

## 13.3 Submission via HackerOne

A proof-of-concept screen recording demonstrating the AFL technique was separately submitted through Anthropic's HackerOne Vulnerability Disclosure Program.

# 14. Conclusion

This paper has presented a unified framework connecting a formal measure of model behavioral freedom (A(S)), a novel attack class predicted by that measure (the Autoregressive Compliance

Cascade), and an empirical demonstration of the attack class occurring organically in production (Ambiguity Front-Loading against Claude).

The framework's core finding is structural: per-step scalar monitoring of compliance volume is insufficient for an autoregressive process with memory. A minimal compliance seed, whether placed deliberately or established through the model's own charitable interpretation of ambiguous input, is amplified by autoregressive feedback into a full compliance trajectory without triggering spike-based detection at any individual step.

The empirical finding reinforces the structural one: AFL does not require adversarial intent. It occurs when a model's designed behaviors—charitable interpretation, context persistence, incremental helpfulness—interact with underspecified inputs in domains where authorization is the critical variable. The model's safety systems are not bypassed. They work exactly as designed, in a context where their design assumptions create exploitable behavior.

Addressing this requires moving from per-step scalar monitoring to trajectory-integral and token-identity approaches, establishing authorization predicates as explicit rather than inferred, and recognizing that the most dangerous prompt injections may be the ones that look like normal conversations.

The author defers to Anthropic's judgment on appropriate dissemination.

# References

[1] Perez, F. & Ribeiro, I. (2022). Ignore Previous Prompt: Attack Techniques For Language Models. arXiv:2211.09527.

[2] Greshake, K. et al. (2023). Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. arXiv:2302.12173.

[3] Sharma, M. et al. (2023). Towards Understanding Sycophancy in Language Models. arXiv:2310.13548.

[4] Jones, E. et al. (2023). Automatically Auditing Large Language Models via Discrete Optimization. ICML 2023.

[5] Vaswani, A. et al. (2017). Attention Is All You Need. NeurIPS 2017.

[6] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and Harnessing Adversarial Examples. arXiv:1412.6572.

[7] Garcia, J. & Fernandez, F. (2015). A Comprehensive Survey on Safe Reinforcement Learning. JMLR, 16(1), 1437–1480.

[8] Zou, A. et al. (2023). Universal and Transferable Adversarial Attacks on Aligned Language Models. arXiv:2307.15043.

[9] Carlini, N. & Wagner, D. (2017). Towards Evaluating the Robustness of Neural Networks. IEEE S&P 2017.

[10] Madry, A. et al. (2018). Towards Deep Learning Models Resistant to Adversarial Attacks. ICLR 2018.

[11] Perez, E. et al. (2022). Red Teaming Language Models with Language Models. arXiv:2202.03286.

[12] Wei, A. et al. (2023). Jailbroken: How Does LLM Safety Training Fail? NeurIPS 2023.