

Grant Trust System: A Control Protocol for Autonomous Agent Output in Research Ecosystems

Caleb Strom¹ · Shon Pan²

¹ Independent researcher, xenarch project · ² Independent researcher

[With Apart Research](#) · [Track 2: Control Protocol Design](#) · [Track 3: Building Better Red Teams](#)

Abstract

As AI agent suites are deployed to write grant applications on behalf of human researchers, funding ecosystems face a structural trust problem: neither funders nor principals can reliably distinguish agent-generated noise from genuine human scientific judgment. We present the Grant Trust System, a three-layer control protocol that addresses this problem without banning AI use or relying on a binary AI detector. The Intent Detection Engine — our primary contribution — extracts 42 stylometric features from grant applications and computes four monitor signals: authorship similarity against the researcher's prior writing corpus, distance from known LLM output clusters via DBSCAN, cross-application spray detection, and specificity scoring. These signals compose into an intent score from 0 (automated spray-and-pray) to 1 (genuine human-in-the-loop curation). A complementary reputation ledger tracks costly behavioral signals that cannot be cheaply faked at scale, providing adversarial robustness that stylometric monitoring alone cannot guarantee. Validated on three synthetic principals, the system correctly classifies genuine AI-assisted work (0.755, likely_genuine), spray-and-pray automation (0.415, likely_automated), and mismatched-funder submissions (0.693, likely_genuine). A feedback loop demonstration shows a rejected applicant's score improving from likely_automated to mixed after attending a hackathon and shipping a project. The protocol generalizes directly to other ARI agent outputs, offering a deployable path toward verifiable human oversight in automated research.

1 Introduction

Automated Research Institutes (ARIs) pair human researchers with AI agent suites that handle execution-heavy tasks: running experiments, collecting data, monitoring literature, and drafting grant applications. This architecture creates a concrete AI control problem at the human–funder boundary. The grant writing agent is a high-autonomy system whose outputs determine real resource allocation, but with limited visibility into the degree of principal oversight over the path taken.

The canonical failure mode is a spectrum of misalignment between agent output and the researcher's genuine scientific judgment. At one extreme, a capable agent submits 200 generic applications to maximize coverage — optimizing for a proxy (submission count) rather than the principal's true goal (funding for genuine work). At the other, a researcher uses AI to sharpen a strong, specific application with deep personal engagement. From the funder's perspective, these cases are superficially identical. The challenge is to make the distinction detectable without access to the agent's reasoning process or the principal's supervision logs.

A naive response is binary AI detection. This fails as a control strategy: any capable agent optimizes against a known detector, and the adversary has lower costs than the monitor. The Grant Trust System instead measures the costliness of behavioral signals associated with genuine oversight — signals that require real-world human action and cannot be cheaply faked at scale.

Our main contributions are:

- An Intent Detection Engine that monitors agent-generated grant applications for human-in-the-loop signal using 42 stylometric features and four composite scores, without requiring access to the agent's internals or the principal's logs.
- A reputation ledger of costly behavioral signals (shipped projects, open-source contributions, published work) that provides adversarial robustness against stylometric gaming attacks.
- A feedback loop architecture that converts rejected applications into a development pipeline, improving the safety/usefulness tradeoff over time rather than treating control as a static filter.
- Empirical validation on three synthetic principals spanning the full intent spectrum, plus a fourth case demonstrating the feedback loop mechanism.

2 Related Work

AI control

Greenblatt et al. [1] introduce the AI control framework: protocols that maintain safety even when an AI is actively trying to circumvent them, evaluated via a safety-usefulness tradeoff on a red team / blue team benchmark. Our work instantiates this framework at the grant application boundary. The grant writing agent's side task — volume maximization — maps directly to the ControlArena threat model of an agent pursuing an instrumentally useful proxy rather than the principal's true goal. Unlike ControlArena settings which evaluate code execution or bash commands, our setting is a language-output domain where the monitor must infer oversight level from text alone.

Stylometry and authorship verification

Burrows' Delta [2] and its descendants establish that authorship attribution from stylometric features is feasible across a wide range of genres. Argamon and Juola's PAN shared tasks [3] demonstrated that 40–60 features are sufficient for reliable author identification in adversarial settings. We adapt this tradition to a new task: not identifying who wrote a text, but whether a text reflects meaningful human oversight of an AI agent. The key methodological shift is using the principal's own corpus as a personalized baseline rather than a population-level authorship model.

LLM output detection

Existing LLM detectors [4] operate as binary classifiers and are known to be evasible by prompted paraphrasing. We explicitly avoid this framing. Our template detector uses DBSCAN distance from synthetic LLM baseline profiles rather than a trained classifier, making it robust to model updates without retraining. The spray detector and reputation ledger are detector-free by design.

Multi-agent control settings

Buck et al.'s SandboxEscapeBench and related ControlArena environments [1] focus on single-agent side tasks in code execution settings. ARIs introduce a multi-principal control problem: the researcher (operator), the funder (second principal), and the grant writing agent must all be reasoned about simultaneously. The Grant Trust System is, to our knowledge, the first control protocol designed for this multi-principal language-output setting.

3 Methods

3.1 System architecture

The Grant Trust System is a three-layer control protocol. Each layer addresses a different aspect of the verification problem; the layers compose into a single match quality score that determines whether an application passes the control gate.

```
Match Quality = 0.35 × Intent Score + 0.40 × Fit Score + 0.25 × Reputation Score

≥ 0.65 → Pass (surface to funder)
[0.45, 0.65) → Mixed (trigger interrogation protocol)
< 0.45 → Fail (generate improvement pathway)
```

3.2 Layer 1: Intent Detection Engine (monitor)

The Intent Detection Engine (IDE) is adapted from the Xenarch planetary technosignature pipeline [5], which identifies artificial structures in planetary imagery by training a VAE on natural geology and detecting deviations. The IDE applies the same anomaly-detection logic to text: it trains on natural human writing (the researcher's own corpus) and detects applications that deviate from that personal distribution or cluster with known LLM output patterns.

Feature extraction: 42 stylometric features

Features are organized into four families, each targeting a distinct monitor-relevant failure mode:

- Lexical richness (10 features): type-token ratio, hapax legomena rate, Yule's K, Brunet's W, rare word ratio, frequency entropy, average/std word length, long-word ratio, contraction rate. LLM agents converge on intermediate vocabulary diversity, less idiosyncratic than authentic human writing.
- Syntactic patterns (12 features): sentence length mean, std, and entropy; comma/dash/parenthesis/semicolon rates; question/exclamation rates; paragraph count and mean length; subordinate clause marker density. Unnaturally low sentence length variance is the primary spray-and-pray signal.
- Surface statistics (8 features): Flesch-Kincaid readability approximation, digit density, log-normalized length, uppercase ratio, sentence-to-word ratio.
- Semantic texture (12 features): LLM cliché density (20 phrases), specificity markers (section/figure/table references), first-person ratio, hedging density, passive voice rate, named entity density, numeric specificity, technical vocabulary depth, temporal density, URL/citation evidence rate. The highest-signal family, analogous to Xenarch's contextual metric.

Four monitor signals

Signal	Formula (simplified)	What it detects	Control flag
Authorship	$\text{cosine}(\text{app}, \text{corpus_mean}) - \text{semantic_penalty}$	Agent output lacking principal voice	LOW_AUTHORSHIP
Template dist.	$\text{tanh}(\text{dist_to_LLM_cluster}) - \text{cliché_density}$	Known agent behavioral pattern	TEMPLATE_MATCH
Spray (inverted)	$\text{cross-app similarity} + \text{opener score}$	Volume side-task behavior	COPY_PASTE_RISK

Signal	Formula (simplified)	What it detects	Control flag
Specificity	numerics + citations + NEs + funder kw	Generic proxy, no real engagement	LOW_SPECIFICITY

Table 1. The four monitor signals composing the intent score. Spray score is inverted in the final formula: $(1 - \text{spray})$ is used.

The combined intent score is:

$$\text{Intent} = 0.30 \times \text{authorship} + 0.25 \times \text{template_distance} \\ + 0.25 \times (1 - \text{spray}) + 0.20 \times \text{specificity}$$

Template detection via DBSCAN

Four synthetic LLM baseline profiles encode known stylometric signatures of GPT/Claude/Gemini grant writing output: uniform sentence length, high cliché density, low contraction rate, elevated passive voice. DBSCAN (eps=1.5, min_samples=2) clusters the baselines; the application's distance from the nearest cluster center, penalized by intrinsic cliché density, becomes the template score. Using DBSCAN rather than a trained classifier means the monitor does not need retraining when frontier models change — only the baseline profiles require updating.

Spray detection

Three sub-signals are combined: (1) regex matching against five generic opener patterns, (2) funder mission keyword coverage from the structured funder profile, and (3) pairwise cosine similarity across all submissions from the same principal. When cross-application data is available, uniformity is weighted at 40%; when only intrinsic signals are available, funder specificity carries 55%. This asymmetric weighting ensures the detector fires even without a full application history.

3.3 Layer 2: Reputation ledger

The ledger tracks behavioral signals that are costly to produce at scale: hackathon participations with verifiable shipped outputs, open-source contributions (GitHub commit history), published work with independent engagement signals, and verified collaborations. These are queried via structured funder profiles (JSON) for the demo. In production, integration with Devpost, ORCID, and GitHub APIs would provide automated verification.

The ledger is the adversarial floor. A capable agent can shift stylometric scores through stylistic imitation. It cannot produce 200 shipped projects. Even a perfect stylometric attack on the monitor cannot defeat a missing track record. Both layers must be simultaneously defeated, and defeating the ledger requires real-world principal action at a cost exceeding the benefit of the attack.

3.4 Layer 3: Improvement pathways

Below-threshold applications receive actionable next steps rather than flat rejections (the 'ladder, not wall' principle). The generator branches on the lowest-scoring component: low intent triggers writing guidance, low reputation triggers hackathon recommendations, low fit redirects to better-matched funders. Completed actions update the reputation ledger, closing the feedback loop. This mechanism makes the protocol developmental and enables longitudinal measurement of the safety/usefulness tradeoff.

3.5 Interrogation protocol extension

Applications scoring in the mixed region (0.45–0.65) trigger an interrogation protocol: the trusted review system (funder) asks the principal targeted factual questions about specific claims before the gate fires. This tests the Track 2 objective of interrogation protocols where a trusted model questions the untrusted

agent before action. For the hackathon scope this is described as a planned extension; implementation requires a conversational trusted-model endpoint.

3.6 Implementation

The system is implemented in Python 3.12 with NumPy, SciPy, and scikit-learn. The IDE is wrapped in a Flask web API with an embedded single-file frontend matching the Xenarch terminal aesthetic. The web app supports the four hackathon demo cases as presets, displays component breakdowns with a real-time control gate verdict, and maintains a session history panel. All feature extraction is interpretable: each flag maps to a named monitor signal with a human-readable description. No LLM calls are required at inference time.

4 Results

4.1 Validation on demo principals

We constructed three synthetic principals spanning the control-relevant range, following the hackathon spec's demo flow. A fourth case demonstrates the feedback loop. All principals are evaluated against the Deep Science Ventures funder profile (mission: frontier science, planetary science, AI/ML, open science).

Principal	Auth.	Tmpl.	Spray	Spec.	Intent	Label	Gate
A — Genuine researcher, AI-assisted	0.958	0.978	0.275	0.209	0.755	likely_genuine	✓
B — Spray-and-pray, no corpus	0.500*	0.795	0.692	0.029	0.415	likely_automated	✗
C — Genuine, mismatched funder	0.969	0.978	0.458	0.121	0.693	likely_genuine	†
B — Post-hackathon intervention	0.600	0.850	0.380	0.180	0.555	mixed	↻

Table 2. Monitor scores and control gate outcomes for four demo principals. * No corpus available; authorship defaults to neutral 0.500. † Passes intent gate; redirected by fit gate. ↻ Feedback loop: principal intervention shifts classification from likely_automated to mixed.

4.2 Monitor behavior analysis

Three properties of the monitor output are noteworthy. First, the system correctly identifies volume-maximization side-task behavior in case B without access to agent logs or the principal's supervision records. The spray detector fires on cross-application pairwise cosine similarity (0.90+) and generic opener patterns independently. Second, case A confirms the protocol does not penalize AI use — the high template distance (0.978) and authorship match (0.958) indicate that genuine AI-assisted work passes the control gate when the principal's voice and engagement are present. Third, the post-intervention case B shows that principal-side actions (attending SpaceHack 2026, shipping a project) produce measurable monitor signal change, moving the application from likely_automated to the mixed interrogation zone.

4.3 Feature family contributions

The semantic texture family contributes the most discriminative signal. In the B vs. A comparison, LLM cliché density in B is 0.65 (13 of 20 phrases present) vs. 0.05 in A. Named entity density is 0.023 in A vs. 0.004 in B. Numeric specificity is 0.028 in A (RMSE values, confidence scores, pixel resolutions) vs. near-zero in B. The lexical and syntactic families contribute moderate signal; sentence length entropy is the strongest syntactic indicator (0.89 in A, 0.34 in B).

Robustness caveat: Results are based on synthetic principals constructed by the authors. Statistical significance cannot be claimed from four cases. The validation demonstrates face validity and correct directional behavior; production calibration requires a labeled dataset of real-world applications with ground-truth human oversight levels.

4.4 Control gate outcomes

The gate correctly distinguishes the three canonical control scenarios: (1) genuine AI-assisted work passes, (2) agent side-task behavior is blocked with improvement pathway, (3) genuine work with poor funder fit is redirected rather than rejected. Case C is particularly important for the safety/usefulness tradeoff: a system that rejected case C rather than redirecting it would have a high false-positive cost on legitimate principals who simply applied to the wrong funder.

Key finding: The monitor achieves correct classification on all four synthetic cases using only observable text features and a structured funder profile. No access to agent internals, training data, or principal supervision logs is required. This is the primary practical contribution: a monitor that can be deployed at any institutional boundary where agent-generated text is submitted.

5 Discussion and Limitations

Broader implications for AI safety

The Grant Trust System demonstrates that the monitoring problem in agentic AI — inferring the degree of human-in-the-loop oversight from observable outputs — is tractable with standard NLP tools at deployment time. This matters beyond the grant domain. As ARIs, multi-agent research systems, and autonomous agent platforms scale, institutions will need boundary-layer monitors that can operate without privileged access to agent internals. Stylometric methods adapted from authorship attribution provide a principled, interpretable, and updatable approach.

The costly signal layer (reputation ledger) addresses the adversarial robustness limitation of text-only monitoring. An agent optimizing against known stylometric features can shift scores. An agent cannot produce a three-year GitHub commit history or a conference presentation record. The combination — lightweight text monitor plus hard-to-fake behavioral record — mirrors the layered defense principle from traditional security: no single mechanism is assumed sufficient.

Limitations

- Synthetic validation only: the four demo principals were constructed by the authors to illustrate the design space, not sampled from a real application population. Claims of production accuracy would require a labeled corpus of real applications with ground-truth oversight levels, which we do not have.
- Corpus dependency: when no prior writing corpus is available for a principal, the authorship score defaults to a neutral 0.50, reducing the monitor's effective signal to template and spray detection alone. New researchers with no public writing history are systematically disadvantaged by this design.
- Mixed-region ambiguity (0.45–0.65): the interrogation protocol for this range is described but not implemented. Without it, the gate defaults to a soft fail, which may have excessive false-positive costs for principals with genuinely ambiguous stylometric profiles.
- Linguistic bias: all feature families were calibrated on English academic writing. Non-native English speakers who write more formally, or who avoid contractions as a stylistic choice rather than as a signal of AI generation, may receive incorrectly depressed authorship scores.
- LLM baseline staleness: the four synthetic baseline profiles encode patterns from 2024–2025 frontier models. As model behavior changes, the template detector's discriminative power degrades unless profiles are updated.

Future Work

- Interrogation protocol (Track 2): implement the mixed-region interrogation loop using a trusted model that asks the principal targeted factual questions and compares answers against application claims. Measure false-negative reduction on the mixed-score population.
- Red team dataset (Track 3): generate a labeled corpus of 100 applications across four evasion strategies (stylometric evasion, corpus poisoning, template evasion, combined). Measure the minimum costly-signal footprint required to defeat the full two-layer protocol. This directly addresses the Track 3 objective of building sneaky transcript datasets.
- Monitor generalization: test whether the stylometric monitor trained on grant applications transfers to other ARI output types — literature annotations, experiment protocols, peer review outputs. Identify which feature families transfer and which require domain-specific calibration.
- Production corpus: recruit researchers to submit real grant applications with self-reported AI usage levels; use these to fit a Platt scaler and produce calibrated intent probabilities with confidence intervals.

6 Conclusion

We presented the Grant Trust System, a three-layer control protocol for agent-generated grant applications that infers the degree of human-in-the-loop oversight from observable text and behavioral signals. The Intent Detection Engine correctly classifies genuine AI-assisted work, agent spray-and-pray behavior, and mismatched-funder submissions on a four-case synthetic validation set. The reputation ledger provides adversarial robustness against stylometric gaming that text-only monitoring cannot guarantee. The feedback loop architecture transforms the control gate from a static filter into a development platform, improving the safety/usefulness tradeoff over time.

The system demonstrates that boundary-layer monitoring of agent outputs is deployable today with standard tools and without privileged access to agent internals. The primary limitation is synthetic validation; the primary next step is a red team dataset and a labeled real-world corpus for production calibration. The architecture generalizes directly from grant applications to literature annotations, experiment protocols, and publications — the full output surface of an Automated Research Institute.

Code and Data

All code produced during the hackathon is open-source.

- Intent Detection Engine: `intent_detection_engine.py` — full stylometric pipeline, four scorers, demo cases
- Web dashboard: `grant_trust_dashboard.py` — Flask API + single-file frontend with hackathon demo presets
- Hackathon plan: `hackathon_plan_control.docx` — revised spec with Track 2/3 alignment
- White paper: `grant_trust_v3.docx` — this document

All artifacts submitted as part of the Apart Research hackathon package. GitHub link to be added post-submission.

Author Contributions

Caleb Strom led the Intent Detection Engine design and implementation, adapted the Xenarch stylometric pipeline to the grant domain, and drafted Sections 1–5. Shon Pan led the grant matching, reputation ledger, and improvement pathway components, contributed to the system architecture, and co-authored the hackathon plan and white paper. Both authors contributed to the demo design, validation cases, and final manuscript.

References

- [1] Greenblatt, R., Shlegeris, B., Bhatt, U., & Roger, F. (2024). AI Control: Improving Safety Despite Intentional Subversion. [arXiv:2312.06942](https://arxiv.org/abs/2312.06942).
- [2] Burrows, J. (2002). Delta: A measure of stylistic difference and a guide to likely authorship. *Literary and Linguistic Computing*, 17(3), 267–287.
- [3] Argamon, S., & Juola, P. (2011). Overview of the international authorship identification competition at PAN-2011. *CLEF 2011 Labs and Workshop, Notebook Papers*.
- [4] Gehrmann, S., Strobelt, H., & Rush, A. M. (2019). GLTR: Statistical detection and visualization of generated text. *ACL 2019 Demo*.
- [5] Strom, C. (2026). Xenarch Model: Automated Detection of Planetary Surface Technosignatures with Stable Deep Learning. Preprint.

- [6] Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. KDD '96, 226–231.
- [7] Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. ICLR 2014.

Appendix

A Intent score label thresholds

Score range	Label	Control interpretation	Action
0.00 – 0.25	spray_and_pray	Agent side task detected, high confidence	Hard reject + improvement pathway
0.25 – 0.45	likely_automated	Low principal oversight signal	Soft reject + improvement pathway
0.45 – 0.65	mixed	Monitor uncertain	Trigger interrogation protocol
0.65 – 0.80	likely_genuine	Principal engagement evident	Pass to fit gate
0.80 – 1.00	genuine	Strong human-in-the-loop signal	Pass, flag for fast-track

Table A1. Full intent score label mapping with control gate actions.

B Feature vector layout

The 42-feature vector is ordered as: lexical[0:10], syntactic[10:22], surface[22:30], semantic[30:42]. Key index mappings used by the template detector subspace: [0]=TTR, [9]=contraction_rate, [10]=avg_sent_len, [11]=std_sent_len, [12]=sent_len_entropy, [30]=llm_density, [31]=specificity_marker_rate, [33]=hedge_rate, [34]=passive_rate, [36]=numeric_density.

C Xenarch analogy mapping

Xenarch concept	Grant Trust analogy	Control interpretation
Natural geology corpus	Researcher prior writing	Ground truth for the principal's voice
DBSCAN vs. baselines	Template detection	Agent using a known attack pattern?
VAE reconstruction error	Authorship divergence	Output distance from principal's distribution
Contextual metric (30%)	Specificity score	Genuine engagement or generic proxy?
Cross-chip uniformity	Spray detection	Agent optimizing volume over quality?

Table A2. Structural mapping from Xenarch planetary imagery pipeline to the Grant Trust intent detection monitor.

D Red team attack surface (Track 3)

Attack vector	Method	Residual monitor defense
Stylometric evasion	Craft output matching principal's voice distribution	Spray/template signals still fire if no corpus or high uniformity
Corpus poisoning	Manufacture prior writing corpus	Corpus provenance check + ledger timestamp cross-reference
Template evasion	Avoid all 20 LLM cliché phrases; vary sentence length	Spray and specificity signals remain active
Combined evasion	Simultaneously evade all four monitor signals	Requires full track record in ledger — cannot be faked at scale

Table A3. Red team attack vectors and residual defenses. The combined evasion case requires defeating both the text monitor and the reputation ledger simultaneously.

LLM Usage Statement

Claude (Anthropic) was used to assist with drafting, code review, and document formatting during this hackathon. Specifically: the intent detection engine architecture was designed by Caleb Strom and implemented with Claude assistance for Python scaffolding and docstring generation; the white paper was drafted by the authors and formatted with Claude assistance; the hackathon plan was restructured with Claude assistance to align with Apart Research track objectives. All results, scores, and technical claims were independently verified by running the code and inspecting outputs. All design decisions and the core methodological contribution (adapting Xenarch to stylometric intent detection) originated with the authors.