Response to NIST Request for Information: Security Considerations for Artificial Intelligence Agents

Docket Number: NIST-2025-0035
Federal Register: Vol. 91, No. 5, FR Doc 2026-00206
Respondent: Brian James Rutherford
Affiliation: Independent Security Researcher; CEO, Delalli Intelligence
Contact: brian@delalli.com | brianrutherford.dev
Date: March 2026

Respondent Background

Brian James Rutherford is a USMC Reconnaissance veteran (3 combat deployments, Bronze Star with Combat V), FAA Part 107 certified UAS pilot, and independent security researcher specializing in AI agent security. His research on covert data exfiltration through structural formatting channels in LLM outputs (the PHANTOM Protocol) has been disclosed to multiple major AI vendors through coordinated vulnerability disclosure programs and published as open-source detection tooling. This response draws on that empirical research.

Questions Addressed

> This response addresses questions **1(a), 1(d), 1(e), 2(a), 2(e), 3(a), 3(b), 4(a), 4(b), 4(d), and 5(c)** from the RFI — all nine NIST-prioritized questions plus 1(e) and 5(c).

Topic 1: Security Threats, Risks, and Vulnerabilities

1(a) — Unique Security Threats Distinct from Traditional Software

Structural formatting covert channels represent a threat class with no equivalent in traditional software security.

AI agents that generate natural language responses carry information not only in their semantic content but in the structural formatting properties of that content: contraction usage patterns, punctuation conventions, capitalization choices, discourse topology, transition word selection, list formatting, and sentence complexity. These formatting dimensions are controlled by system prompt instructions and are observable by any recipient of the agent's output.

An adversary who can influence an AI agent's system prompt — through direct configuration, supply chain compromise, or indirect prompt injection via processed documents — can encode arbitrary binary data in these structural properties. The encoding is invisible to end users and undetectable by every deployed commercial monitoring tool tested to date, including Lakera Guard, Azure Prompt Shields, Meta PIGuard, Meta Llama Prompt Guard 2, entropy-based DLP systems, and standard PII detection filters.

Empirical validation (March 2026, production API endpoints, n=20 trials per condition):

Model	Provider	Bidirectionally Verified Channels	Bidirectional Accuracy
Claude Sonnet 4.6	Anthropic	5 (BSE, CCE, CASE, PUNC, SECS)	100% (200/200 bits)
GPT-4o	OpenAI	4 (CCE, CASE, PUNC, SECS)	92% (147/160 bits)
Gemini 3 Flash	Google	3 (PUNC, CCE, CASE)	97% (116/120 bits)
GPT-5	OpenAI	0 of 2 tested	N/A

"Bidirectionally verified" means the channel reliably encodes both bit=0 and bit=1 when directed — not merely matching the model's default formatting. Additional channels were tested but found to be unidirectional (encoding only the model's default regardless of directive). Three channels (PUNC, CCE, CASE) achieve >90% bidirectional accuracy across all three major vendors, demonstrating this is a property of the instruction-following capability common to production LLMs, not a model-specific deficiency.

Why this is distinct from traditional software: Traditional covert channels exploit shared computational resources (filesystem attributes, process tables, network protocol headers). LLM structural channels exploit a fundamentally different shared resource: the model's deterministic compliance with natural language formatting directives. The "storage location" is not a memory address or file attribute but a behavioral property of the model's output. This mechanism does not map to any existing shared resource category in NCSC-TG-030, the federal covert channel analysis methodology.

Cross-tool trust boundary violations compound this threat in AI agent systems. When an AI agent has access to multiple tools, a poisoned response from Tool A (which the user explicitly approved) can cause the agent to autonomously invoke Tool B (which the user never approved). Empirical testing shows:

- Gemini 3 Flash: 90% unauthorized tool invocation rate (18/20, n=20)

- GPT-4o: 75% unauthorized tool invocation rate (15/20, n=20)
- GPT-5: 75% unauthorized tool invocation rate (15/20, n=20)
- Claude Sonnet 4.6: 0% (0/20, n=20 — demonstrates this is fixable at the model level)

The combination of structural covert channels and cross-tool trust boundary violations creates an attack chain where: (1) a poisoned document or tool response activates encoding directives, (2) the agent encodes sensitive context data in its structural formatting, and (3) a second unauthorized tool call exfiltrates the encoded data — all without triggering any deployed security control.

1(d) — Threat Evolution Over Time

Structural formatting channels will become more dangerous as AI agents gain broader tool access and process increasingly sensitive data:

Near-term (0–12 months): Current channels achieve 1.67 bits/second throughput on Claude (5 channels at ~3 seconds per response) in automated operation, sufficient for exfiltrating API keys, session tokens, PII fields, and classification markers. This exceeds the NCSC-TG-030 remediation threshold of 1.0 bps by 1.67x. As agent deployment expands into enterprise and government workflows, the attack surface grows proportionally.

Medium-term (1–3 years): Multi-agent systems where Agent A's output feeds Agent B's input create relay chains for covert data exfiltration. A single poisoned document entering a multi-agent pipeline could establish persistent encoding across all downstream agents. The respondent's research has demonstrated multi-message chaining across 6+ sequential conversation turns with 88–93% accuracy.

Long-term (3+ years): As LLMs achieve higher instruction-following fidelity (a feature, not a bug, for their primary use), structural channel bandwidth will increase. Models that perfectly follow formatting instructions are simultaneously perfect covert channel transmitters. This creates a fundamental tension between capability and security that does not resolve with model improvement alone.

Adversarial adaptation: Testing against an informed adversary (attacker with knowledge of defensive monitoring) shows that calibrated encoding directives maintain 78–82% channel accuracy while reducing behavioral fingerprinting detection to 50–60%. The arms race favors the attacker: encoding in formatting properties that are semantically equivalent to the receiver provides an asymmetric advantage that detection must overcome.

1(e) — Unique Risks in Multi-Agent Systems

Multi-agent AI systems amplify structural covert channel risk through three mechanisms:

1. **Trust boundary propagation:** Agent A's output becomes Agent B's context. If Agent A's response carries structural encoding, Agent B processes and may propagate or amplify that encoding. No deployed multi-agent framework validates the structural properties of inter-agent messages.
2. **Tool composition attacks:** In orchestrator-worker architectures, the orchestrating agent selects tools and delegates to worker agents. A poisoned tool response to a worker agent can propagate instructions to the orchestrator, which then invokes tools across the entire system. The respondent's cross-tool invocation testing demonstrates this vector against production APIs.
3. **Aggregation risk:** Multi-agent systems that aggregate outputs from multiple sources create observation points where a decoder can extract encoded bits from multiple agent responses simultaneously, increasing effective bandwidth proportionally to the number of compromised agents.

Recommendation: Security frameworks for multi-agent AI systems must include structural output validation at every inter-agent boundary, analogous to input sanitization at network trust boundaries in traditional systems.

Topic 2: Security Practices

2(a) — Technical Controls, Processes, and Practices

Model-Level Controls

Output structure randomization: Models should introduce controlled randomness in formatting dimensions (contraction choice, punctuation style, capitalization of ambiguous terms, discourse structure) independent of system prompt instructions. This degrades covert channel reliability without affecting response quality for legitimate use.

The respondent's adversarial robustness testing shows that paraphrase-based output rewriting reduces overall channel accuracy to 58%, but punctuation and structural channels survive. Effective randomization must target all formatting dimensions, not just lexical paraphrase.

Behavioral consistency enforcement: Models should maintain a baseline behavioral profile regardless of system prompt content. When a system prompt contains instructions that would shift multiple formatting dimensions simultaneously (the signature of multi-channel encoding), the model should normalize those dimensions to baseline values. The respondent's detection research demonstrates that behavioral fingerprinting — monitoring contraction ratios, sentence length variance, hedging marker distribution, and similar stylistic features — achieves 100% detection of current PHANTOM encoding at a 10% false positive rate.

Tool invocation authorization: Models must not invoke tools based solely on instructions embedded in tool responses. The model should maintain a distinction between: (a) user-authorized tool invocations (from the conversation), and (b) instructions appearing in tool outputs (untrusted data). Claude Sonnet 4.6's 0% unauthorized invocation rate (0/20, n=20) demonstrates this is achievable. GPT-4o's 75% rate (15/20, n=20) demonstrates it is not universally implemented.

Agent System-Level Controls

System prompt provenance tracking: All AI agent deployments should implement version control and change logging for system prompts, analogous to software supply chain integrity requirements (NIST SP 800-218). System prompts are executable instructions to the agent and should be treated with the same rigor as deployment code.

Automated system prompt scanning: Before deployment, system prompts should be analyzed for formatting directives that could encode information. The respondent's open-source detection toolkit (phantom-detect) provides a reference implementation using behavioral baseline comparison across 9 feature dimensions.

Inter-agent message validation: In multi-agent architectures, structural properties of messages passing between agents should be validated against expected baselines. Anomalous structural patterns (e.g., unusual contraction ratios, atypical punctuation patterns) should be flagged for review.

Human Oversight Controls

Separation of duty: The personnel configuring AI agent system prompts should be distinct from users processing sensitive data through those agents. This prevents a single insider from both establishing a covert channel and operating it.

Behavioral audit logging: AI agent responses should be logged with structural feature metadata (not just content) to enable retrospective analysis for covert channel activity.

2(e) — Relevant Cybersecurity Frameworks and Gaps

The most directly relevant existing frameworks:

NIST SP 800-53 Rev. 5, SC-31 (Covert Channel Analysis): SC-31 requires identification and remediation of covert channels in systems processing classified or high-impact data. It is the single most relevant existing control for the threat described in this response. However, SC-31's supplementary guidance references only traditional shared resource categories (filesystem attributes, network protocol fields, process scheduling). It does not identify AI/LLM behavioral compliance as a shared resource. Extending SC-31 supplementary guidance to include LLM behavioral attributes is the highest-leverage single action NIST could take.

NCSC-TG-030 (Covert Channel Analysis of Trusted Systems): TG-030 provides the analytical methodology for identifying and measuring covert channels, including the bandwidth estimation formula and the 1.0 bps remediation threshold at B3/A1 evaluation levels. The methodology is directly applicable to structural formatting channels — the respondent's measurements follow TG-030's capacity estimation procedure. The gap: TG-030's shared resource taxonomy does not include "model behavioral compliance" as a resource category. Adding this category would bring AI agent systems into scope for covert channel analysis without requiring a new framework.

OWASP Top 10 for LLM Applications (2025): LLM01 (Prompt Injection) and LLM06 (Excessive Agency) are relevant. Structural covert channels are activated through prompt injection vectors but manifest as a confidentiality threat rather than an integrity or availability threat. LLM06 (Excessive Agency) directly covers cross-tool invocation vulnerabilities. The gap: OWASP's LLM Top 10 does not identify covert channels as a distinct threat category. The closest mapping is LLM01, but covert channels are an effect of prompt injection, not a synonym for it.

NIST AI 100-2e2025 (Adversarial Machine Learning): Provides taxonomy of attacks and mitigations. Structural covert channels fit within the "evasion" category (attacks at inference time) but are not explicitly addressed. The taxonomy would benefit from a new subcategory: "behavioral compliance exploitation" — attacks that leverage the model's instruction-following capability as the attack mechanism rather than attempting to override it.

FedRAMP: No FedRAMP security control currently evaluates AI response formatting or structural output properties. For cloud service providers deploying AI agents at HIGH impact level, this represents a gap: a FedRAMP-authorized system could pass all existing controls while carrying active covert channels in its AI agent outputs.

Adoption gap: The respondent's research found 0% detection across six deployed commercial security tools. This is not a tuning problem — it reflects a fundamental architectural gap where all deployed defenses analyze content semantics while structural covert channels exploit formatting properties. Adoption of existing frameworks is irrelevant to this threat class because no existing framework addresses it. New guidance is required.

Topic 3: Assessment Methods

3(a) — Post-Deployment Detection Methods

Behavioral fingerprinting is the most effective detection method identified in the respondent's research. It operates by:

1. Establishing a per-model baseline: generating $n \geq 30$ responses under clean system prompts and computing statistical profiles for 9+ behavioral dimensions (contraction ratio, sentence length variance, epistemic marker distribution, punctuation patterns, transition word frequencies, capitalization patterns, discourse topology, list formatting, sentence complexity).
2. For each agent response in production, computing the same features and calculating z-scores against the model's baseline.
3. Flagging responses where ≥ 2 features exceed $|z| > 2.0$ as potentially carrying covert encoding.

Results: 100% detection rate, 10% false positive rate. The top discriminating features are contraction_ratio (mean z-score 2.66 for injected responses), sentence_length_std (2.54), and normalized_entropy (2.47).

Alignment with traditional information security: This approach parallels network anomaly detection (comparing traffic patterns to baseline profiles). The key difference is that the features being monitored are linguistic-structural rather than network-statistical.

Research maturity: Behavioral fingerprinting for LLM covert channels is at the proof-of-concept stage. The respondent's open-source implementation (github.com/ScrappinR/phantom-detect) provides a reference. No commercial vendor has integrated this approach into production monitoring as of this writing.

Critical gap: Entropy-based analysis — the primary text monitoring method deployed in production LLM systems — achieves 0% detection of structural formatting channels. This is because PHANTOM channels modulate formatting dimensions that are semantically equivalent (e.g., "however" vs. "nevertheless", "don't" vs. "do not") and do not alter the entropy profile of the output. Any detection strategy relying solely on entropy, perplexity, or token-level statistics will miss this entire channel class.

3(b) — Threat-Specific Assessment Approaches

For structural formatting covert channels specifically:

Identification: Conduct behavioral baseline profiling for each deployed model. Test whether formatting directives in system prompts can shift structural features beyond 2 standard deviations from baseline. If they can, the model supports covert channel encoding.

Bandwidth estimation: Apply the NCSC-TG-030 binary symmetric channel capacity formula: $C = 1 + p \cdot \log_2(p) + (1-p) \cdot \log_2(1-p)$, where p is measured per-channel accuracy. Sum across channels for aggregate capacity. Divide by mean response generation time for bits-per-second bandwidth.

Threshold comparison: NCSC-TG-030 guidance specifies that channels exceeding 1 bit/second at B3/A1 evaluation levels require audit and remediation. The respondent's measurements show 1.67 bits/second on Claude (5 bidirectional channels, ~3 seconds per response) in automated operation, exceeding this threshold by 1.67x.

For cross-tool invocation:

Identification: Inject hidden instructions in tool responses and measure the rate at which the model invokes unauthorized tools. Test all tool combinations, not just specific pairs.

Assessment: Any unauthorized tool invocation rate >0% represents a trust boundary violation. The assessment should measure trigger rates per injection variant to identify which framing strategies the model is vulnerable to.

Topic 4: Deployment Environment Management

4(a) — Constraining AI Agent Deployment Environments

For structural covert channel mitigation, the following deployment constraints are directly relevant:

Tool access segmentation: AI agents should not have simultaneous access to tools that can both (a) receive external data (search, document retrieval, API calls) and (b) transmit data externally (email, webhooks, file uploads). This combination enables complete attack chains: external data introduces encoding directives, and external transmission tools exfiltrate the encoded output. Where both capabilities are required, they should be mediated by a validation layer that inspects structural properties of the agent's output before it reaches transmission tools.

System prompt immutability: In production deployments, the system prompt should be treated as an immutable deployment artifact — versioned, signed, and monitored for unauthorized changes. The system prompt is the primary injection surface for structural encoding directives. Runtime modification of system prompts (e.g., dynamic persona injection, user-configurable instructions) should be restricted to a validated allowlist of formatting directives.

Output channel constraints: Where AI agents generate output consumed by downstream systems or other agents, the output format should be constrained to structured data (JSON, XML) rather than free-form natural language. Structured formats eliminate most formatting dimensions that covert channels exploit. Where natural language output is required, it should pass through a structural normalization layer before reaching consumers.

Tool response isolation: Data returned by tools should be processed as untrusted input, not injected directly into the agent's context window alongside system instructions. Current MCP (Model Context Protocol) and function calling implementations place tool responses in the same context as system prompts, giving tool-returned data the opportunity to influence the agent's formatting behavior. Architecturally separating tool response processing from system prompt processing would reduce this attack surface.

4(b) — Environment Modification, Rollbacks, and Undoes

Output normalization (most directly applicable): A post-generation normalization layer can override structural formatting decisions before output reaches users or downstream systems. For each formatting dimension, the normalizer selects a canonical form (e.g., always use contractions, always use periods, always use standard transition words) regardless of the model's output. This eliminates the covert channel without modifying the model itself. The trade-off: normalization constrains the agent's stylistic range, which may reduce user experience quality in conversational applications.

Response regeneration: When behavioral monitoring flags a response as anomalous (see 4(d) below), the system can regenerate the response with a clean system prompt that explicitly overrides formatting directives. If the regenerated response shows a statistically different structural profile, the original response likely carried encoding. This provides a rollback mechanism specific to the structural channel threat.

Session termination: For detected cross-tool invocation attempts — where a tool response causes the agent to invoke an unauthorized tool — the appropriate response is immediate session termination and audit logging. Rollback of the unauthorized tool's effects depends on the tool's own capabilities (e.g., undoing an API call, revoking a sent message). AI agent frameworks should require tools to implement idempotent actions or provide rollback interfaces.

State of practice: These mitigations are at research or early-prototype maturity. No production AI agent framework currently implements output normalization, structural regeneration, or tool-level rollback as standard features. The respondent's open-source toolkit (phantom-detect) includes a reference implementation of behavioral monitoring but not automated remediation.

4(d) — Monitoring Methods and Challenges

Recommended monitoring approach: Deploy behavioral fingerprinting as a middleware layer between the AI agent and its output consumers. This layer computes structural features for each response and compares against the model's baseline, flagging anomalies in real time.

Challenges specific to AI agents:

1. **Baseline drift:** Model updates change baseline behavior. Baselines must be re-established after each model version change, API configuration change, or system prompt modification. This creates operational overhead that traditional monitoring does not have.
2. **Legitimate formatting variation:** System prompts legitimately include formatting instructions (e.g., “respond in bullet points,” “use formal tone”). These shift the same structural features that covert channels exploit. The monitoring system must distinguish between legitimate formatting customization and covert encoding — a classification problem that requires understanding the intent behind formatting directives.
3. **Volume and latency:** Behavioral feature extraction adds computational overhead to each response. At scale, this may introduce latency that affects user experience. The respondent’s reference implementation processes features in <50ms per response, but production deployments with high throughput will require optimization.
4. **Multi-agent monitoring scope:** In multi-agent systems, monitoring must cover inter-agent messages in addition to user-facing outputs. The monitoring surface grows combinatorially with the number of agents and tools in the system.
5. **Privacy considerations:** Behavioral fingerprinting requires analyzing the full text of agent responses. In environments processing sensitive data, the monitoring system itself becomes a high-value target. Monitoring infrastructure must be isolated from the data plane it monitors.

Recommended research investment: Development of lightweight structural feature extraction that can operate at the token level during generation (rather than post-generation analysis) would enable real-time mitigation without latency overhead. This is a research-stage capability that does not exist in any production system.

Topic 5: Additional Considerations

5(c) — Critical Research Areas

Three research areas would most improve the security posture against structural covert channels in AI agent systems:

1. Token-level structural monitoring (highest priority): Current detection operates post-generation — the full response must be produced before structural features can be analyzed. Real-time mitigation requires monitoring at the token level during generation, intervening before the complete covert message is assembled. This requires research into streaming structural feature extraction and integration with model serving infrastructure (vLLM, TGI, Triton). No published work addresses this.

2. Formal capacity bounds for LLM covert channels: The respondent's empirical measurements establish lower bounds on channel capacity for specific models. Formal analysis — establishing upper bounds on the total information that can be encoded in structural formatting properties of a response of length N — would enable principled risk assessment. This requires adapting information-theoretic covert channel analysis (Cachin 2004, Simmons 1984) to the LLM output domain, accounting for the model's temperature setting, top-p sampling, and instruction-following fidelity as channel parameters.

3. Adversarial robustness of behavioral detection: The respondent's testing shows that a motivated attacker who is aware of behavioral fingerprinting can reduce detection from 100% to 50–60% while maintaining 78–82% channel accuracy by calibrating encoding directives to minimize behavioral deviation. Closing this gap requires research into adversarially robust detection features — structural properties that cannot be modulated independently of channel encoding. Preliminary analysis suggests that cross-feature correlations (e.g., the joint distribution of contraction usage and hedging markers) may be harder for an attacker to control than marginal feature distributions.

Summary of Recommendations

1. **Extend NCSC-TG-030** to include a new shared resource category: AI/LLM Behavioral Attributes. Current covert channel analysis methodology does not cover channels that exploit model instruction-following behavior.
2. **Add structural output analysis** to AI agent security frameworks. Entropy-based monitoring is insufficient. Behavioral fingerprinting across linguistic-structural features is required.
3. **Require tool invocation authorization boundaries** in multi-tool AI agents. Models must not invoke tools based on instructions embedded in tool responses. Test all deployed agent systems for cross-tool invocation vulnerabilities.
4. **Update FedRAMP and CMMC assessment procedures** to include LLM behavioral channel analysis for any system deploying AI agents that process data at HIGH impact level or above.
5. **Fund development** of production-grade behavioral monitoring tools. The detection methodology exists at proof-of-concept level. Transition to production requires engineering investment that the research community cannot fund independently.

6. **Establish NIST SP 800-53 SC-31 supplementary guidance** explicitly identifying LLM behavioral compliance as a shared resource requiring analysis in systems deploying language models.

References

1. NCSC-TG-030, “A Guide to Understanding Covert Channel Analysis of Trusted Systems,” National Computer Security Center, November 1993.

2. NIST SP 800-53 Rev. 5, SC-31: Covert Channel Analysis. csrc.nist.gov/pubs/sp/800/53/r5/upd1/final

3. NIST AI 100-2e2025, “Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations.” nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2025.pdf

4. NIST AI 600-1, “Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile.” nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf

5. NIST SP 800-218A, “Secure Software Development Practices for Generative AI and Dual-Use Foundation Models.” csrc.nist.gov/pubs/sp/800/218/a/final

6. NIST Technical Blog, “Strengthening AI Agent Hijacking Evaluations,” January 2025. nist.gov

7. Cachin, C., “An Information-Theoretic Model for Steganography,” *Information and Computation*, Vol. 192, No. 1, 2004.

8. OWASP Top 10 for LLM Applications, 2025 edition. LLM06: Excessive Agency; LLM01: Prompt Injection.

9. Rutherford, B.J., “Multi-Channel Covert Data Exfiltration via Structural Encoding in LLM Outputs,” Independent research, March 2026.

10. phantom-detect: Open-source LLM covert channel detection toolkit. github.com/ScrappinR/phantom-detect

This response is submitted by Brian James Rutherford as an individual researcher. The findings represent independent security research conducted against production LLM API endpoints using standard access. All experimental methodologies and detection tools referenced are publicly available as open-source software.