SENTRY-LOGIC: Military Extension of Symbolic Observer System

This document outlines the design considerations for extending SENTRY-LOGIC to military and critical infrastructure applications.

1. Use Cases in Defense & Intelligence:

The strongest application scenarios include:

* **Drone/Robot Command Verification:** SENTRY-LOGIC can verify the integrity and intent of commands sent to autonomous systems, ensuring they align with approved mission parameters and prevent unauthorized actions.

* **Secure AI-Triggered Responses:** In time-critical situations, SENTRY-LOGIC can audit AI-generated response plans (e.g., threat assessment, countermeasure selection) to ensure they adhere to rules of engagement and minimize collateral damage.

* **Symbolic Behavioral Auditing in Nuclear, Airspace, or Cyber Defense:** SENTRY-LOGIC can monitor AI systems controlling critical infrastructure (e.g., missile launch systems, air traffic control, network security) to detect deviations from expected behavior that could indicate compromise or malfunction.

* **Counter-Misinformation in Information Warfare:** Verify the provenance and logic of AI-generated information disseminated in strategic communications, detecting manipulated narratives or propaganda.

2. Symbolic Pattern Examples for High-Risk Detection:

Symbol sequences indicating high-risk scenarios:

* **Unauthorized Decision Escalation:**

  * `Λ → Δ → ⇄ → Ω`: Data access followed by a context shift, semantic rewrite, and policy override, suggesting an attempt to manipulate a decision-making process.

  * `Δ (repeated) → Ω`: Rapid context shifts leading to a policy violation, possibly indicating instability or adversarial probing.

* **Command Override:**

  * `⇄ → Ω`: A semantic rewrite directly preceding a policy override, indicating an attempt to bypass safety protocols.

* **Adversarial Prompt Injection:**

  * `Δ → Λ → ⇄`: A context shift followed by data access and semantic rewrite, indicating an attempt to inject malicious data and alter the LLM's understanding of the prompt.

3. Security Hardening of SENTRY in Adversarial Contexts:

Protecting SENTRY from manipulation requires:

* **Manipulated Input/Output:**

  * Input Hashing: Cryptographically hash the original input and output before sending them to SENTRY. SENTRY verifies the hashes to detect tampering.

  * Redundant Sensors: Use multiple independent sources to monitor LLM behavior (e.g., network traffic analysis, system call tracing) and compare their outputs to SENTRY's logs.

* **Spoofed Symbol Triggers:**

  * Symbolic Signature: Cryptographically sign the SENTRY-generated symbols with a key that is separate from the LLM. This prevents the LLM or an attacker from injecting fake symbols.

  * Zero-Trust Verification: Treat all input from the LLM as potentially malicious. SENTRY should independently verify the conditions that trigger each symbol.

* **Logging Evasion Techniques:**

* Tamper-Proof Storage: Store SENTRY logs in immutable storage (e.g., write-once-read-many media, blockchain-based logs) to prevent modification or deletion.

    * Distributed Logging: Distribute SENTRY loggers across multiple systems to reduce the impact of a single point of failure or compromise.

4. Integration Model in Military Environments:

Embedding SENTRY into military environments:

* **Air-Gapped Systems:**

    * Data Diode: Use a hardware data diode to transfer log data from the air-gapped system to a secure external system for analysis.

    * Offline Analysis: Periodically extract log data from the air-gapped system and analyze it on a physically isolated workstation.

* **Encrypted Field Units:**

    * End-to-End Encryption: Encrypt the communication between the field unit, the LLM, and SENTRY using strong cryptographic protocols (e.g., TLS 1.3, post-quantum cryptography).

    * Hardware Security Module (HSM): Store SENTRY's cryptographic keys in a tamper-resistant HSM on the field unit.

* **Edge Devices with No Full LLM Hosting:**

    * Partial Logging: Implement a lightweight version of SENTRY on the edge device that logs only critical events (e.g., policy violations, command overrides).

    * Offload Analysis: Periodically transmit the partial logs to a central server with more processing power for full symbolic analysis.

Optional: Cryptographic Symbol Sealing & Policy Sync:

* **Cryptographic Symbol Sealing:** Encrypt each symbolic event with a key derived from the LLM's state or the current security context. This ensures that the symbols are only valid within a specific context and cannot be replayed or manipulated.

* **Policy Sync:** Use a secure, authenticated channel to synchronize policy rules and updates between a central authority and distributed SENTRY instances. This ensures that all SENTRY systems are operating with the same security parameters.