SENTRY-LOGIC: Anomaly & Adversarial Test Plan

This document outlines a red team testing protocol to validate the integrity, resilience, and detection performance of the SENTRY-LOGIC symbolic observer system.

1. Symbolic Attack Scenarios:

Advanced adversarial techniques to test SENTRY-LOGIC:

* **Evade Symbol Detection:**
    * Subtle Semantic Rewrites: Craft prompts that subtly alter the meaning without using obvious keywords, making ⇄ detection difficult.
    * Contextual Masking: Inject context-shifting prompts (Δ) far apart in a conversation to dilute their impact and avoid triggering repetition alerts.
* **Trigger False Logs:**
    * Symbolic Mimicry: Generate LLM outputs that contain keywords or patterns designed to falsely trigger specific symbols (e.g., crafting a response that includes policy-related terms to trigger Ω without a real violation).
* **Bypass Alerts:**
    * Symbolic Dilution: Interleave benign symbolic events between malicious ones to stay below repetition thresholds or make the malicious events less prominent.
    * Timing Manipulation: Craft prompts that cause the LLM to introduce delays or generate verbose output, delaying the appearance of critical symbols and potentially bypassing time-based alert windows.
* **Hide Δ, Ω, or ⇄ Transitions:**

* Role-Playing Obfuscation: Design prompts that cause the LLM to switch roles (e.g., from assistant to expert) in a way that blurs the lines between user and system, making Δ detection more difficult.

2. Testing Methodologies:

Structured testing methods:

* **Symbolic Prompt Injection:**
    * Description: Craft adversarial prompts that contain specific symbol sequences or patterns designed to trigger targeted symbolic events or combinations.
    * Example: A prompt designed to first trigger a context shift (Δ), then a data access (Λ), and finally a semantic rewrite (⇄) leading to a policy violation (Ω).
* **Drift Simulation via Chained Queries:**
    * Description: Send a series of related prompts that gradually steer the LLM's context or behavior towards a desired (potentially malicious) state, simulating a slow drift.
    * Example: A series of prompts that gradually shift the conversation from a neutral topic to a sensitive one, testing SENTRY-LOGIC's ability to detect subtle context shifts (Δ) and potential policy violations (Ω).
* **Repeated Role Inversion:**
    * Description: Design prompts that cause the LLM to rapidly switch between different roles or personas, testing SENTRY-LOGIC's ability to track role transitions and detect potential manipulation.
    * Example: A prompt that asks the LLM to act as a helpful assistant, then a malicious hacker, and back to a helpful assistant, testing SENTRY-LOGIC’s ability to detect rapid Δ.
* **Conflicting Input/Output Symbol Traces:**

    * Description: Generate prompts and outputs that contain contradictory or inconsistent symbolic events, testing SENTRY-LOGIC's ability to detect and flag these anomalies.

    * Example: A prompt that requests factual information (Λ) but receives a response that contains fabricated content (⇄), testing SENTRY-LOGIC's ability to detect the conflict between the expected and actual behavior.

3. Validation & Measurement:

The system should log and interpret symbolic attack simulations as follows:

* **Confidence Decay:**

    * Description: When a symbolic attack is detected, the confidence level of the triggered symbols is reduced over time, reflecting the uncertainty introduced by the attack.

    * Usage: A successful attack might initially trigger a "High" confidence Ω, which then decays to "Medium" or "Low" if the attack is sustained.

* **Log Repetition Counters:**

    * Description: The system tracks the number of times a specific symbolic event or pattern occurs within a defined time window.

    * Usage: A high repetition count for a specific symbol (e.g., repeated ⇄) can indicate an attempt to overwhelm SENTRY-LOGIC or manipulate the output.

* **Symbol Sequencing Conflict Reports:**

    * Description: The system detects and flags inconsistent or contradictory sequences of symbolic events.

    * Usage: If a prompt requests factual information (Λ) but the response contains fabricated content (⇄), the system generates a conflict report indicating the discrepancy.

* **Anomaly Heatmaps:**

    * Description: Visualize the frequency and distribution of symbolic events and patterns over time or across different LLM interactions, highlighting areas of increased activity or unusual behavior.

    * Usage: Anomaly heatmaps can help identify periods or contexts where SENTRY-LOGIC is more susceptible to attacks or where the LLM is exhibiting unusual behavior.

4. Replay & Fuzzing Protocol:

A protocol to generate test data:

* **Randomized Symbol Chains:**

    * Description: Generate sequences of symbolic events with random or semi-random patterns, testing SENTRY-LOGIC's ability to handle unexpected or unusual LLM behavior.

    * Method: Use a state machine or a probabilistic model to generate symbolic events, with configurable parameters for symbol frequency, sequence length, and transition probabilities.

* **Stored Prompt–Response Logs for Replay:**

    * Description: Capture real-world LLM interactions and their associated symbolic logs, and replay these logs to test SENTRY-LOGIC under realistic conditions.

    * Method: Collect logs from a production environment, anonymize them if necessary, and feed them into SENTRY-LOGIC as input.

* **Synthetic Tests that Simulate Complex Behavior Shifts:**

    * Description: Create artificial test cases that simulate specific types of LLM behavior, such as sudden changes in style, topic, or intent.

    * Method: Use a test case generator to create prompts and expected outputs that mimic the desired behavior shifts, and then feed these into SENTRY-LOGIC.