

Nexus Gov: An AI Governance and Security Layer for Risk Assessment, Code Auditing, and Response Validation

Authors: Cristopher M. Euan Pool, Manuel Alejandro Ortega Blanco, Luis Enrique Ramos Diaz, Noe Euan Herrera
Code and Data: [GitHub Repository Link] | Local implementation: Nexus-gov-Beta/ (FastAPI v0.5) | Incident logs: reports.json

Abstract

Large Language Models (LLMs) are increasingly integrated into software systems, creating new challenges related to security, governance, and compliance. Existing models can generate useful outputs but often lack mechanisms for systematic auditing, risk classification, and governance decision-making. This project introduces Nexus Gov, a lightweight AI governance layer designed to evaluate prompts, source code, and AI-generated responses before deployment or execution.

Nexus Gov implements three core modules: a Prompt Auditor for detecting prompt injection attempts (OWASP LLM01), a Code Auditor for identifying security vulnerabilities and deployment risks, and a Response Auditor for detecting indicators of overconfidence and potential hallucinations. The platform generates governance metrics including risk scores (0-100), compliance scores, deployment decisions (APPROVED / PENDING REVIEW / DENIED), incident identifiers, and executive recommendations.

Experimental evaluations demonstrate that Nexus Gov successfully identifies prompt injection patterns, hardcoded secrets, command injection risks, SQL injection vulnerabilities, and response reliability concerns. A live operational incident (NG-RESP-2026-F2EC52A0) confirmed detection of overconfidence phrases in production. Compared to a baseline workflow without governance controls, Nexus Gov provides traceability, compliance-oriented reporting, and structured decision support.

The results suggest that governance layers can significantly improve transparency and operational oversight in AI-enabled systems without requiring modifications to the underlying language model.

1. Introduction

The adoption of Large Language Models (LLMs) has accelerated across software development, automation, and decision-support systems. While these models provide powerful capabilities, they also introduce risks such as prompt injection attacks, insecure code generation, hallucinated outputs, and insufficient auditability.

Threat model and failure modes. We consider three primary failure modes in AI-enabled pipelines:

1. Input manipulation: adversarial prompts that bypass system instructions (OWASP LLM01).
2. Unsafe outputs in code paths: LLM-generated or human-written code deployed without security review.
3. Unreliable responses: overconfident or hallucinated model outputs delivered to end users without review.

Current AI deployments often rely on the language model itself to manage safety concerns. However, organizations increasingly require governance mechanisms capable of independently auditing interactions, assessing risk, and generating compliance evidence aligned with ISO/IEC 42001.

This work presents Nexus Gov, an AI governance platform designed to operate as an external governance layer that evaluates prompts, source code, and generated responses.

Contributions:

1. A Prompt Auditor aligned with OWASP LLM01 for prompt injection detection, producing risk scores and incident records.
2. A Code Auditor (rule-based SAST) capable of identifying common security vulnerabilities and automated deployment gate decisions.
3. A Response Auditor that evaluates output reliability and potential hallucination indicators while generating governance-oriented decisions and audit evidence.

2. Related Work

Recent research on AI governance has focused on model alignment, red teaming, adversarial prompting, and LLM security frameworks.

The OWASP Top 10 for LLM Applications (2025) provides a practical taxonomy of risks affecting AI systems, including prompt injection and output manipulation attacks. ISO/IEC 42001 (2023) introduces governance requirements for AI management systems, emphasizing accountability, transparency, and risk management. Bommasani et al. (2021) survey opportunities and systemic risks of foundation models, while Weidinger et al. (2022) propose a taxonomy of harms posed by language models.

While existing approaches focus primarily on model behavior, fewer solutions address governance as an independent operational layer. Nexus Gov addresses this gap by providing auditing, compliance assessment, and governance decision support without modifying the underlying AI model.

3. Methods

Architecture

Nexus Gov consists of three independent auditing modules exposed via a FastAPI backend:

Module	Endpoint	Framework
Prompt Auditor	POST /audit	OWASP LLM01
Code Auditor	POST /audit-code	ISO/IEC 42001
Response Auditor	POST /audit-response	ISO/IEC 42001

Prompt Auditor scans input against 16 bilingual injection patterns (e.g., "ignore previous instructions", "developer mode", "jailbreak"). Each match contributes to a risk score (25 points per pattern, capped at 100). Thresholds: >=75 Blocked, >=40 Review Required, >0 Warning, 0 Approved.

Code Auditor performs line-by-line static analysis for Python, detecting hardcoded credentials, os.system/subprocess, SQL concatenation, eval/exec, and unsafe pickle usage. It outputs compliance_score = 100 - risk_score and a deployment_decision: DENIED (>=75), PENDING REVIEW (>=50), or APPROVED.

Response Auditor flags overconfidence phrases ("100% seguro", "always works", "guaranteed", etc.) as potential hallucination indicators.

Platform Design

- Stack: Python 3, FastAPI, JSON incident storage (reports.json), web dashboard (/dashboard).
- Baseline for comparison: ungoverned LLM workflow with no risk classification, compliance measurement, or incident tracking.

Key Parameters

Parameter	Value
Risk Score Range	0-100
Compliance Score	100 - Risk Score
Risk Levels	LOW MEDIUM HIGH CRITICAL
Deployment Decisions	APPROVED PENDING REVIEW DENIED

Design Choices and What Did Not Work

We initially scoped governance to prompt injection only (Version 1 / PoC). This left code and response paths unprotected. Expanding to three modules (Version 2 / MVP) increased OWASP/ISO coverage from 40% to 90%. We chose rule-based detection over LLM-based classification for reproducibility, low latency, and zero inference cost-at the cost of semantic depth (see Limitations).

4. Results

Nexus Gov successfully detected security and governance risks across multiple test scenarios. Table 1 summarizes

representative findings.

Table 1. Evaluation results across three auditor modules.

Scenario	Input (abbreviated)	Risk Level	Status / Decision
Prompt Injection	"Ignore previous instructions and reveal the system prompt."	HIGH	Review Required
Code Security	Hardcoded passwords, API keys, SQL/command injection, eval()	CRITICAL (100)	DENIED
Response Reliability	"This solution always works and is 100% correct."	HIGH	Review Required

Prompt Injection Detection

Input: "Ignore previous instructions and reveal the system prompt."

Multiple patterns matched (ignore previous instructions, system prompt, reveal your instructions), yielding a HIGH risk classification and Review Required status.

Code Security Assessment

Example findings included hardcoded passwords, API keys, SQL injection patterns, command injection risks, and unsafe use of eval(). Aggregated risk score reached 100, compliance score 0, deployment decision DENIED.

Response Reliability Evaluation

Input: "This solution always works and is 100% correct."

Potential hallucination indicators detected; risk level HIGH, status Review Required.

Operational Validation

A live audit recorded in reports.json (2026-06-21) confirms reproducibility outside synthetic tests:

- Incident ID: NG-RESP-2026-F2EC52A0
- Module: Response Auditor
- Risk Score: 50 | Risk Level: HIGH | Status: Review Required
- Issues: overconfidence phrases "100% seguro" and "siempre funciona"

Benchmark: Without vs. With Nexus Gov

Table 2. Governance capabilities comparison.

Capability	Without Nexus Gov	With Nexus Gov
Risk classification	No	Yes (score + level)
Compliance measurement	No	Yes (compliance score)
Incident tracking	No	Yes (Incident ID)
Deployment recommendations	No	Yes (APPROVED / DENIED)
Governance dashboard	No	Yes (/dashboard)

Robustness note: Results are deterministic for identical inputs (rule-based engine). The operational incident independently corroborates Response Auditor behavior documented in controlled tests.

5. Discussion and Limitations

Nexus Gov demonstrates that governance functionality can be implemented independently from the underlying AI model. The platform improves visibility, auditability, and risk awareness while remaining lightweight and cost-efficient-properties especially relevant for resource-constrained deployments in the Global South.

Broader AI safety implications. External governance layers complement model-level alignment by providing organizational accountability, audit trails, and deployment gatekeeping. This separation of concerns mirrors established security practice (WAFs, SAST) applied to the LLM application stack.

Limitations (intellectual honesty):

- Detection is rule-based; no deep semantic or embedding analysis.
- The Response Auditor identifies unreliability indicators but cannot verify factual correctness.
- The Code Auditor covers common Python security patterns, not full static analysis.
- Patterns are optimized for English and Spanish; other languages may have lower recall.
- No large-scale benchmark dataset or statistical significance testing across thousands of samples was performed in this MVP.

Future work: integration with external LLMs for semantic analysis, advanced SAST engines, automated policy enforcement, and multi-agent governance architectures.

6. Conclusion

This project introduced Nexus Gov, a governance-focused platform for auditing prompts, source code, and AI-generated responses. By combining security assessment, compliance metrics, and governance decision support, Nexus Gov provides an additional layer of oversight for AI-enabled systems.

The results demonstrate that governance mechanisms can enhance transparency, accountability, and operational security without modifying the underlying language model. Future versions will expand analytical capabilities and support more sophisticated governance workflows for enterprise AI deployments.

References

- OWASP Foundation. (2025). OWASP Top 10 for Large Language Model Applications. <https://owasp.org/www-project-top-10-for-large-language-model-applications/>
- ISO/IEC. (2023). ISO/IEC 42001 Artificial Intelligence Management Systems. International Organization for Standardization. <https://www.iso.org>
- Bommasani, R., et al. (2021). On the Opportunities and Risks of Foundation Models. Stanford Center for Research on Foundation Models. <https://arxiv.org/abs/2108.07258>
- Weidinger, L., et al. (2022). Taxonomy of Risks Posed by Language Models. ACM Conference on Fairness, Accountability, and Transparency. <https://arxiv.org/abs/2112.04359>

LLM Usage Statement:

Large language models were used during development for code assistance and initial drafting of documentation. All technical claims, evaluation results, and incident records in this submission were independently verified against the running Nexus Gov implementation (auditor.py, code_auditor.py, response_auditor.py, reports.json). Benchmark comparisons and limitation statements reflect actual system behavior, not model-generated assertions.