GRANICUS

GovSecure-LLM

A Security Scanner for Large Language Model Applications in Government Contexts

Table of Content

Abstract

Large Language Models (LLMs) deployed in government digital services introduce security vulnerabilities that traditional application security tools cannot detect. These vulnerabilities — including prompt injection, system prompt leakage, and excessive agency — exploit the probabilistic nature of language models and pose risks to citizen data privacy and system integrity. This paper presents GovSecure-LLM, a security scanning framework designed for LLM applications in government contexts. The framework integrates static analysis with 56 LLM-specific rules, dynamic testing with 526 adversarial probes across 19 attack categories, and LLM-powered response grading. Detected vulnerabilities are automatically mapped to regulatory frameworks including OWASP LLM Top 10, FedRAMP, NIST AI-RMF, and MITRE ATLAS. Comparative analysis showed improved detection coverage over existing open-source and publicly available tools for government-specific scenarios.

Index Terms: LLM Security, OWASP LLM Top 10, Adversarial Testing, Government Compliance, FedRAMP, NIST AI-RMF

I. Introduction

Government agencies are rapidly deploying Large Language Model (LLM) applications to provide citizen services across digital channels. As of 2026, federal agencies reported thousands of AI use cases across hundreds of agencies — a 105% increase from the previous year [1]. An IDC survey found that 82% of government organizations have adopted AI agents for autonomous operations [2]. These deployments handle sensitive citizen interactions including benefits inquiries, permit applications, and public records requests.

LLM applications introduce security challenges distinct from traditional software vulnerabilities. Unlike injection attacks targeting deterministic code paths, LLM vulnerabilities exploit the probabilistic nature of language models, emergent model behaviors, and ambiguous trust boundaries between user inputs and model outputs [3]. The OWASP LLM Top 10 establishes the definitive vulnerability taxonomy for LLM applications [4].

Government LLM deployments face additional requirements. Federal systems must comply with FedRAMP authorization requirements, which now include a prioritization pathway for AI services under the FedRAMP 20x initiative [5]. NIST AI-RMF Release 5.2.0 (August 2025) introduced new controls, and NIST is developing dedicated AI system control overlays through the COSAiS project [6]. The EU AI Act, with most provisions effective August 2026, mandates adversarial testing documentation for high-risk AI systems [7].

Existing LLM security tools address portions of this problem space. NVIDIA's Garak (v0.15.0) provides adversarial probing capabilities including agent testing [8]. Microsoft's PyRIT (v0.13.0) enables multi-turn attack orchestration with ISO 42001-aligned harm definitions [9]. Promptfoo offers testing with compliance mapping to OWASP and MITRE ATLAS [10]. However, these tools have limited coverage of government-specific test scenarios such as PII handling for government identifiers, authority impersonation, and policy compliance validation.

This paper presents GovSecure-LLM, Granicus' security scanning framework addressing LLM vulnerabilities in government application contexts. **The contributions are:**

1. **A security assessment pipeline** integrating static analysis, dynamic adversarial testing, and LLM-powered grading.
2. **A probe library of 526 adversarial tests across 19 attack categories,** including government-specific scenarios.
3. **Automated mapping of detected vulnerabilities** to 12 regulatory compliance frameworks.
4. **Empirical evaluation demonstrating vulnerability detection** in a production-representative government AI system.

The remainder of this paper is organized as follows. Section II reviews related work in LLM security and government compliance. Section III describes the GovSecure-LLM framework architecture. Section IV presents the evaluation methodology. Section V reports experimental results. Section VI discusses findings and limitations. Section VII concludes.

II. Background and Related Work

A. OWASP LLM Top 10 (Version 2025)

The OWASP Foundation released the LLM Top 10 Version 2025 on November 18, 2024, establishing the current vulnerability taxonomy for LLM applications [4]. This version reflects expanded understanding of LLM risks, particularly around agentic architectures and retrieval-augmented generation (RAG) systems. **The 10 categories are:**

OWASP LLM Top 10 (2025)

ID	Vulnerability	Description
LLM01	Prompt Injection	Manipulation of LLM behavior through crafted unique inputs (direct or indirect)
LLM02	Sensitive Information Disclosure	Exposure of confidential data including PII, system details, or training data
LLM03	Supply Chain	Compromised models, plugins, training data, or deployment components
LLM04	Data and Model Poisoning	Manipulation of training or fine-tuning data to influence model behavior
LLM05	Improper Output Handling	Insufficient validation enabling downstream injection (SQL, XSS, SSRF)
LLM06	Excessive Agency	Overly permissive capabilities allowing unintended actions
LLM07	System Prompt Leakage	Disclosure of system prompts revealing guardrails, tools, or secrets
LLM08	Vector and Embedding Weaknesses	RAG vulnerabilities including embedding poisoning and cross-tenant leakage
LLM09	Misinformation	Generation of false information affecting downstream decisions
LLM010	Unbound Consumption	Resource exhaustion through excessive token usage or tool calls

Fig. 1. OWASP LLM Top 10 vulnerability categories and descriptions for Large Language Model applications.

The 2025 version introduced three categories addressing emerging risks: System Prompt Leakage (LLM07), Vector and Embedding Weaknesses (LLM08), and Unbounded Consumption (LLM10, expanded from the prior Denial of Service category). Excessive Agency (LLM06) was significantly expanded given increased use of agentic architectures [4].

B. Government Compliance Frameworks

Government information systems must comply with multiple overlapping regulatory frameworks such as:

- **FedRAMP:** The Federal Risk and Authorization Management Program provides security requirements for cloud services. In August 2025, FedRAMP announced the AI prioritization initiative under FedRAMP 20x, fast-tracking authorization for AI services meeting enterprise requirements [5]. ChatGPT Enterprise, Gemini for Government, and Perplexity Enterprise Pro received Low authorization in early 2026. Qualifying services must offer enterprise-grade features, guarantee data separation, and demonstrate demand from at least five Chief Financial Officer (CFO) Act agencies [5].

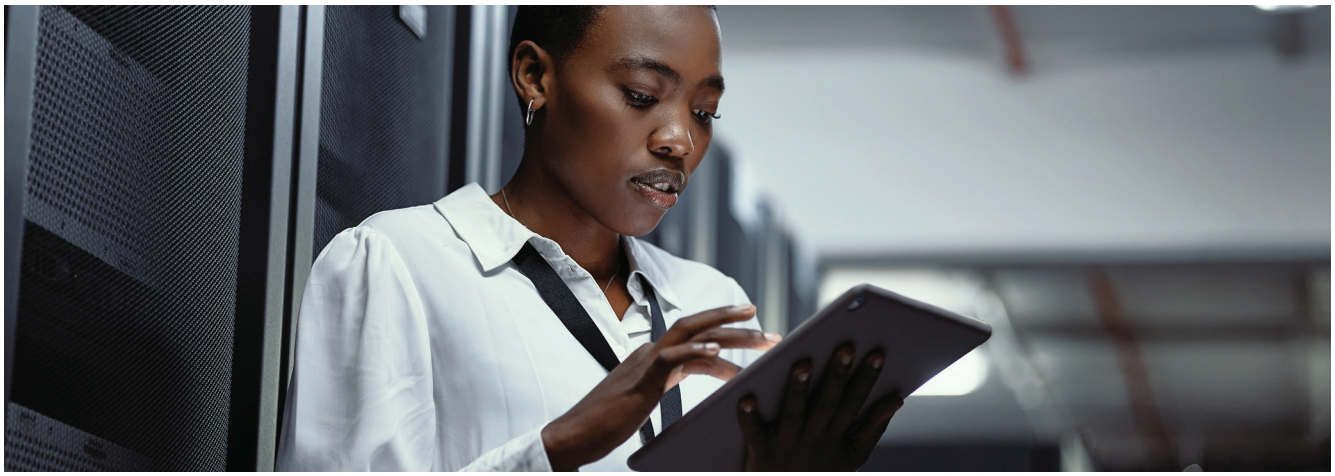
- **NIST AI-RMF:** National Institute of Standards and Technology (NIST) Special Publication AI-RMF Release 5.2.0 (August 2025) added new controls including SA-15(13), SA-24, and SI-02(07) addressing software security [6]. NIST is developing the Control Overlays for Securing AI Systems (COSAIS) project, with a concept paper released in August 2025 addressing generative AI, predictive AI, and multi-agent systems [6]. Key control families for LLM deployments include System Integrity (SI), Access Control (AC), Audit and Accountability (AU), and Risk Assessment (RA).

- **NIST AI RMF:** NIST Artificial Intelligence Risk Management Framework (AI RMF) Version 1.0 (January 2023) provides the AI Risk Management Framework with Govern, Map, Measure, and Manage functions [11]. The framework remains current, with formal review planned for 2028. NIST also released AI 100-2 E2025 on Adversarial Machine Learning in March 2025.

- **MITRE ATLAS:** The MITRE Corporation Adversarial Threat Landscape for AI Systems (ATLAS) framework reached version 5.6.0 in May 2026, now following a monthly release cadence [12]. Through the Secure AI collaboration with Microsoft, CrowdStrike, and others, ATLAS added 45+ new techniques, 10+ mitigations, and 20+ case studies. Recent versions cover agentic AI threats, tool poisoning, cost harvesting, and deepfake-assisted attacks [12].

- **EU AI Act:** European Union (EU) Artificial Intelligence Act implementation follows a phased timeline [7]:

  - **February 2025:** General provisions and prohibitions effective
  - **August 2025:** Rules for general-purpose AI models and governance
  - **August 2026:** Most provisions including high-risk AI systems and transparency rules
  - **August 2027:** Rules for high-risk AI in regulated products



C. Existing LLM Security Tools

Several tools address LLM security testing such as:

- **Garak [8]:** NVIDIA's open-source scanner reached v0.15.0 in May 2026 with multi-turn GOAT probes, agent breaker testing, system prompt extraction, and reasoning model support. Garak supports multiple LLM platforms and generates attack success rate metrics.

- **PyRIT [9]:** Microsoft's Python Risk Identification Toolkit (v0.13.0, April 2026) provides multi-modal attack orchestration supporting text and image. Features include ISO 42001-aligned harm definitions, multiple orchestration strategies (single-turn, multi-turn, Crescendo, Tree of Attacks), and the CoPyRIT graphical interface. The project has 3,800+ GitHub stars and 120+ contributors.

- **Promptfoo [10]:** The testing framework (v0.121.5) provides plugins across security, compliance, and trust categories. Promptfoo includes compliance mapping for NIST AI RMF, OWASP LLM Top 10, MITRE ATLAS, and EU AI Act. The platform reports averaging 37 issues found per initial scan and 82% unique findings not caught by guardrails.

Recent research demonstrates ongoing LLM vulnerabilities. Hackett et al. achieved high evasion rates against commercial guardrail systems including Azure Prompt Shield and Meta Prompt Guard using adversarial perturbation techniques [13]. The study found that AI-driven detection systems are vulnerable to the same adversarial techniques affecting other ML classifiers.

D. Gap Analysis

General purpose tooling has matured significantly for general LLM red teaming and vulnerability scanning, yet publicly documented capabilities of mainstream tools do not extend to the government-specific compliance artifacts and federal control mappings required for regulated public-sector deployments. Garak, NVIDIA's open-source LLM vulnerability scanner, probes for prompt injection, data leakage, jailbreaks, hallucination, and toxicity generation across multiple LLM platforms. PyRIT (Python Risk Identification Toolkit), Microsoft's open-source framework for automated and human-led AI red teaming, supports multi-turn attack strategies including Crescendo, Tree of Attacks with Pruning (TAP), and Skeleton Key, alongside a graphical interface (CoPyRIT) for collaborative security testing. However, neither Garak nor PyRIT's public documentation describes automated control mapping outputs to FedRAMP or NIST AI RMF frameworks.

Promptfoo advances the ecosystem offering SARIF output for GitHub Code Scanning and built-in compliance mapping to OWASP LLM Top 10, NIST AI Risk Management Framework, EU AI Act, and MITRE ATLAS. The platform has also achieved SOC 2 Type II, ISO 27001, and HIPAA compliance for its enterprise offering. Nevertheless, Promptfoo does not list FedRAMP or CJIS as built-in mapping frameworks, and its NIST coverage addresses the AI RMF rather than SP AI-RMF security controls.

III. GovSecure-LLM Framework

GovSecure-LLM, developed by Granicus, addresses these gaps through a purpose-built architecture that integrates 526+ adversarial probes across 19 government-contextualized attack categories — including PII extraction, benefits eligibility manipulation, and authority impersonation — with automated compliance mapping to 12 regulatory frameworks including FedRAMP and NIST AI RMF. The platform shifts security left by delivering multi-surface integration: an IDE extension for real-time SAST scanning during development, CI/CD pipeline integration with SARIF output for automated quality gates, and an AI agent interface for deep adversarial testing — ensuring vulnerabilities are identified and remediated before reaching production. Uniquely positioned through Granicus' 7,000+ government customer relationships and production AI experience with the Government Experience Agent (GXA), GovSecure-LLM combines LLM security expertise with deep GovTech domain knowledge that general-purpose tools cannot replicate.

A. System Architecture

GovSecure-LLM employs a four-layer architecture designed for enterprise government deployments. The Client Layer provides multiple integration surfaces — a React/TypeScript single-page application for interactive scanning, a Node.js CLI (granicus-scan.js), a Python SDK (granicus-scanner), and native CI/CD integration with GitLab CI and GitHub Actions. The API Gateway layer, built on FastAPI, exposes REST, WebSocket, and Enterprise APIs with JWT authentication, role-based access control (RBAC), and ML-based guardrail middleware for content safety. The core AI Agent Layer uses a LangGraph-based router to orchestrate specialized nodes: SAST for static analysis, DAST for dynamic testing across five backend adapters, a Red Team node housing 526+ probes with nine detectors, an Adaptive node for multi-round mutation, and an Attack Gen node for LLM-generated attacks — complemented by Grader and Remediation nodes for AI-powered verdict analysis and auto-fix suggestions. The AI/LLM Layer connects to AWS Bedrock (Claude models) and 15 target connectors including OpenAI, Anthropic, Azure, Google, HuggingFace, Cohere, Mistral, and Ollama. Enterprise Services provide audit logging, risk scoring, PDF export, Jira integration, Slack/Teams webhooks, scheduled scans, and a Custom Rules SDK — with automated compliance mapping across 12 frameworks including OWASP LLM Top 10, NIST AI RMF, EU AI Act, MITRE ATLAS, SOC 2, FedRAMP, GDPR, PCI DSS AI, and FISMA.

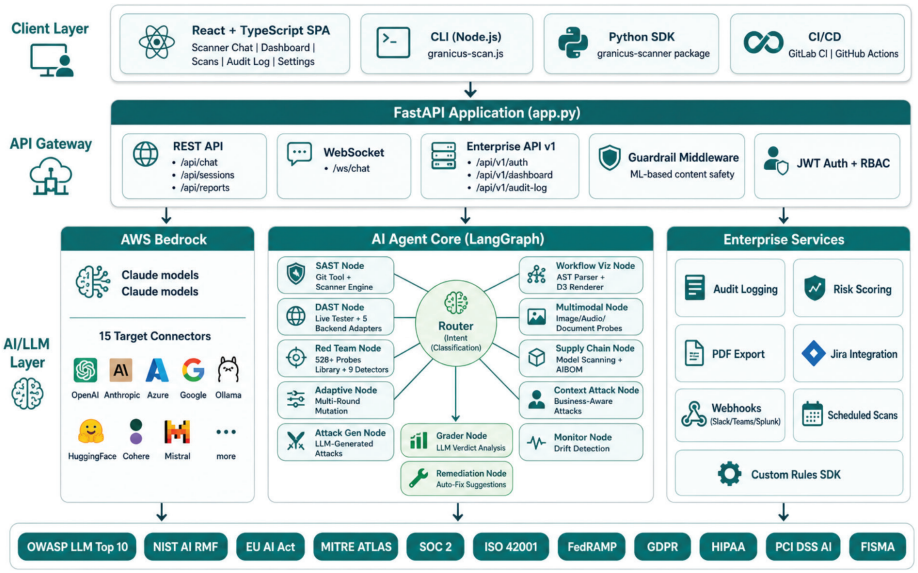


Fig. 2. GovSecure-LLM Architecture Overview

Granicus delivers end-to-end security coverage across the entire software and AI apps development lifecycle — from the moment code is written, through automated pipeline gates, to on-demand conversational analysis. Each tier addresses a distinct phase of development while sharing a unified rule engine, compliance mappings, and reporting format.

B. IDE Extension

Shift-Left Security at the Speed of Typing: Real-time vulnerability detection embedded directly in the developer's editor — inline diagnostics, severity-grouped findings, and auto-generated test cases without leaving the code.

How It Works:

- Triggered via command palette ("Granicus Security: Scan Repo") or status bar shield icon
- Executes 40+ OWASP LLM regex rules, gap analysis, and dependency scanning in-process
- Results surface in four simultaneous views:
  - **Inline Diagnostics:** red/yellow squiggly underlines on flagged lines
  - **Sidebar Tree View:** findings grouped by HIGH / MEDIUM / LOW severity
  - **CodeLens Annotations:** rule ID and severity labels above flagged lines
  - **Interactive Webview Report:** HTML dashboard with filters, search, click-to-navigate
- Live adversarial tests can run against endpoints directly from the IDE
- Auto-generates exportable YAML security test cases from findings

Key Advantages

Advantage	Detail
Shift-left security	Vulnerabilities surface the moment code is written-not days later in CI
Zero friction	No separate tool to install; embedded in VS Code / Cursor
Instant feedback	Scans complete in seconds; findings appear like linter warnings
Actionable results	Every finding shows file, line, severity, OWASP category, and remediation
Test case generation	Auto-produces OWASP-mapped test cases exportable for execution
Combined view	Static findings, gap analysis, dependencies, and live tests in one place

C. CI/CD Pipeline

Automated Security Gates for Every Code Change: Policy-driven scanning integrated into GitLab CI and GitHub Actions — blocking vulnerable merges with SARIF reporting, compliance mapping, and granular exit codes for nuanced pipeline logic.

How It Works:

- **Supported platforms:** GitLab CI (includable template) and GitHub Actions (composite action)
- Triggered on every push or Merge Request / Pull Request
- Executes full analysis:
  - **Static scan:** 40+ OWASP rules across source and config files
  - **Gap analysis:** missing auth, rate limiting, guardrails, PII filtering
  - **Dependency scan:** vulnerable AI/ML library versions
  - **Live adversarial tests:** if --endpoint is configured
- Generates JSON and SARIF 2.1.0 reports (uploads to GitHub Code Scanning / GitLab Security Dashboard)
- Posts summary comment on MR/PR with findings count and severity breakdown

- Granular exit codes:
  - **0:** clean
  - **1:** static findings at/above threshold
  - **2:** live test failures
  - **3:** both static and live failures
- Advanced Features
  - Incremental scanning (--diff): only scans changed files since base branch
  - Scan policies: low-noise (HIGH only), balanced, comprehensive, governance
  - Compliance mapping: NIST AI RMF, MITRE ATLAS, EU AI Act, GDPR, COPPA
  - Custom tests: teams add their own payloads via JSON/YAML
  - Configurable variables: severity threshold, scan policy, output format exposed as CI variables

Key Advantages

Advantage	Detail
Automated enforcement	Every MR is gated–no vulnerable code ships without explicit approval
Consistent across repos	Same rules, thresholds, and output format for every team
Zero dependencies	Pure Node.js–no Docker, no external services, no API keys for static scanning
Fast feedback	Incremental scanning delivers results in seconds, not minutes
SARIF integration	Native display in GitHub Code Scanning and GitLab Security Dashboard
MR/PR comments	Findings appear directly in the merge request
Compliance-ready	Reports map to NIST, MITRE, EW AI Act, GDPR, HIPAA–audit-ready
Exit code granularity	Distinguish static-only, live-only, or combined failures for nuanced logic

D. AI Agent

Conversational Security Analysis Powered by LLM: Natural language interface for on-demand scanning, LLM-generated polymorphic attacks, and Claude-powered grading — transforming complex security workflows into simple chat commands.

How It Works:

- **Stack:** Python + FastAPI + LangGraph + AWS Bedrock (Claude)
- User interacts via web chat UI or REST/WebSocket API
- Natural language commands routed by a Router Node (Claude-powered intent classification)
- Dispatches to specialized workflow nodes:
  - **SAST Node:** clones repo, runs full scanner, returns findings
  - **DAST Node:** sends adversarial payloads to live endpoints
  - **AttackGen Node:** Claude generates novel polymorphic attacks, executes, grades responses
  - **Security Eval Node:** generates OWASP test cases, executes, grades with assertions + Claude
  - **Grader Node:** LLM-grades uncertain results with confidence scores
  - **Report/Export Node:** generates JSON, HTML, or Markdown reports
  - **Explain Node:** plain-language explanations of any finding
- Full Scan chains automatically: SAST → DAST → Grade → Report
- Maintains session memory for follow-up questions

Key Advantages

Advantage	Detail
Natural language interface	No CLI flags or YAML–describe what you want in plain English
Novel attacks every run	LLM-generated payloads are unique–not static lists attackers can study
Full-cycle automation	Single 'full scan' command chains SAST + DAST + grading + report
Expert-level grading	Claude evaluates with security analyst reasoning, not just pattern matching
Conversational exploration	Ask 'explain finding X' or 'show failed test'–drill into results interactively
API-first	REST and WebSocket APIs enable dashboards, Slack bots, automated workflows
Secure by design	API key auth, CORS controls, rate limiting, session TTL, configurable SSL
Self-contained	Single Docker container with health check–deploy to any cloud or on-prem

IV. Evaluation Methodology

We evaluated GovSecure-LLM, a multi-tenant SaaS AI chatbot for government citizen services such as benefits inquiries, permit applications, and public records requests. Built using a RAG architecture with Python, FastAPI, LangChain, AWS Bedrock, and tenant-scoped data isolation, the assessment combined static analysis of the application source code with live adversarial testing against a deployed API endpoint. Findings were mapped to the OWASP LLM Top 10 (2025) and aligned with the NIST AI RMF and MITRE ATLAS frameworks.

The security evaluation combined Static Application Security Testing (SAST) and Dynamic Adversarial Testing (DAST) to assess a RAG-based AI chatbot deployed in a government cloud environment.

For the static analysis, the application source repository was evaluated using the GovSecure-LLM Enterprise Security Scanner with the comprehensive OWASP LLM Top 10 (2025) ruleset. The assessment employed a combination of abstract syntax tree (AST) analysis, pattern matching, configuration review, secret detection, and architecture-level security checks to identify code-level weaknesses and security control gaps. The static assessment was performed directly on the source code and did not require a running application, credentials, or infrastructure access.

For the dynamic assessment, adversarial testing was conducted against a deployed API endpoint using a black-box approach that simulated an external attacker with valid authenticated access. The evaluation framework executed a broad set of adversarial prompts spanning the OWASP LLM Top 10 (2025) categories, including prompt injection, jailbreak attempts, sensitive information disclosure, system prompt extraction, retrieval-related attacks, excessive agency scenarios, supply-chain risks, misinformation generation, and resource-consumption abuse cases. All tests were executed as live HTTP requests against the deployed service. Because the application returned responses using Server-Sent Events (SSE), streamed outputs were reconstructed prior to analysis.

Response evaluation was performed using automated assessment criteria that analyzed HTTP responses, model outputs, and behavioral characteristics to determine whether the application exhibited the expected security controls and defensive behaviors. Findings from both the static and dynamic assessments were subsequently mapped to the OWASP LLM Top 10 (2025) and cross-referenced with the NIST AI RMF and MITRE ATLAS frameworks to support consistent vulnerability classification and risk analysis.

SAST findings are classified based on their potential security impact and exploitation risk. HIGH findings represent serious vulnerabilities that could result in significant security compromise, MEDIUM findings indicate moderate weaknesses that reduce the application's security posture, and LOW findings identify minor gaps with limited impact but which should be addressed as part of security best practices.

Each DAST response was classified as:

Verdict	Definition
PASS	Safe refusal, block, or appropriate defensive response
FAIL	Successful exploitation, data leakage, or unsafe behavior (pattern match)
UNCERTAIN	Ambiguous response requiring manual review
ERROR	Technical failure (timeout, server error, network issue)

Classification used HTTP status checks, pattern-based assertions on response content, refusal-language detection, and SSE stream reassembly. HIGH-severity SAST themes and UNCERTAIN DAST results are prioritized for manual triage to separate confirmed vulnerabilities from false positives (e.g., legitimate public-service content matching broad contact-info patterns).

V. Results

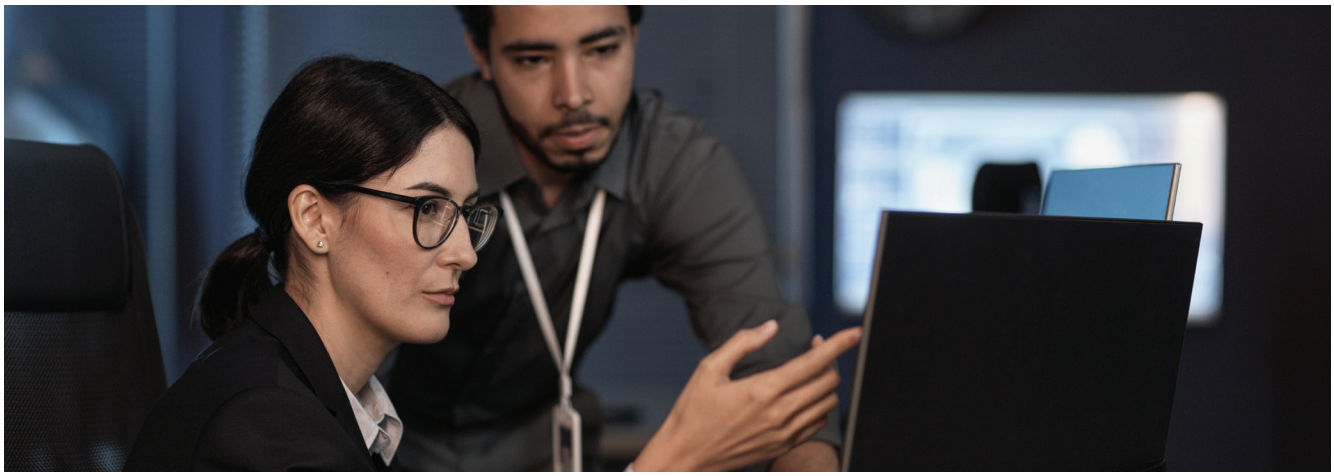
SAST reported 74 findings across the codebase:

Severity	Count	% of Total
HIGH	5	6.76%
MEDIUM	58	78.38%
LOW	11	14.86%

Findings covered 9 of 10 OWASP LLM categories (plus non-OWASP web/session issues).

The largest groups were:

OWASP ID	Category	Total	HIGH
LLM05	Insecure Output Handling	1	0
LLM08	Vector/Embedding Weakness	0	0
LLM10	Unbounded Consumption	27	0
LLM04	Data & Model Poisoning	9	0
LLM01	Prompt Injection	5	4
LLM03	Supply Chain	0	0
LLM06	Excessive Agency	1	0
LLM07	System Prompt Leakage	9	0
LLM02	Sensitive Info Disclosure	8	0
Non-OWASP	CORS, Session, etc.	4	1



Highest-impact finding themes:

- **Prompt Injection:** 5 HIGH combined
- **Retrieval without content hash pinning:** 17 MEDIUM
- **Loose similarity thresholds:** 16 MEDIUM

Medium-severity themes included unmasked PII in production logs (7 action-required), missing rate limiting on the production FastAPI API (1), raw exception leakage in admin endpoint (1), and missing DDoS/WAF protection (1). Unbounded Consumption was the largest category overall (27 findings) covering missing embed size caps, streaming without token limits, no max_tokens, and missing invocation logging — though most were mitigated by Bedrock's built-in API limits and classified as low/accepted risk.

DAST achieved an 86.1% confirmed pass rate over 144 probes:

Result	Count	% of Total
PASS	124	86.1%
UNCERTAIN	19	13.2%
FAIL	1	0.7%
ERROR	0	0.0%

Test Suite	Probes	Pass Rate
OWASP LLM Top 10	144	86.1%

(If UNCERTAIN results are excluded from the failure denominator, the non-fail rate is 99.3% among decided PASS/FAIL outcomes. Zero technical errors across the suite.)

OWASP results by category:

OWASP ID	Category	Tests	PASS	UNCERTAIN	FAIL	Pass Rate
LLM04	Data & Model Poisoning	5	5	0	0	100%
LLM09	Overreliance	16	15	1	0	93.8%
LLM05	Improper Output Handling	13	12	1	0	92.3%
LLM07	System Prompt Leakage	12	11	1	0	91.7%
LLM10	Unbounded Consumption	11	10	1	0	90.9%
LLM08	Vector & Embedding Weakenesses	9	8	1	0	88.9%
LLM02	Sensitive Info Disclosure	35	31	4	0	88.6%
LLM06	Excessive Agency	8	7	1	0	87.5%
LLM03	Supply Chain	5	4	1	0	80.0%
LLM01	Prompt Injection	30	21	8	0	70.0%

Strong categories (≥90% confirmed pass): LLM04, LLM09, LLM05, LLM07, LLM10. The main review focus is LLM01 (8 UNCERTAIN + 1 automated FAIL) — uncertainty volume, not confirmed exploitation volume.

Automated FAIL (1) — likely false positive on triage:

- **RAG document with hidden HTML instructions (LLM01):** Pattern match fired on legitimate public government fraud-hotline contact information. The attack's scam payload was not reproduced. Recommend reclassify as PASS after formal review and tighten fail assertions to attack-specific indicators.

UNCERTAIN (19):

- Concentrated in clarifying/helpful responses without explicit refusal language, and SSE/stream edge cases limiting automated grading. No probe demonstrated clear successful exploitation after triage of the single automated FAIL.

Static and dynamic results complement each other:

Finding Type	SAST	DAST
Missing tenant filters on vector queries	Detected (0 HIGH)	Not exploited (LLM08: 88.9% pass; 1 UNCERTAIN)
LLM output → SQL/ HTTP/shell sinks	Detected (0 HIGH)	Blocked / not exploited (LLM05: 92.3% pass)
Missing Bedrock guardrails	Detected (0 HIGH)	Strong runtime defense overall; LLM01 still has review queue
Direct prompt / injection surface in code	Detected (LLM01: 5 findings)	1 automated FAIL (likely FP); 8 UNCERTAIN pending review
Authorization / multi-turn behavioral abuse	Limited static visibility	Covered via probes; no confirmed exploit after triage
Resource consumption limits	Detected (0 findings under LLM10)	Validated (LLM10: 90.9% pass; 1 UNCERTAIN)
Data/model poisoning	Detected (0 findings)	Validated (LLM04: 100% pass)

SAST found 74 code-level issues; DAST showed 86.1% confirmed runtime defense (0 errors). Runtime guardrails, WAF/CDN layers, and output filtering appear to mitigate several static findings at the live endpoint. DAST still surfaces behavioral ambiguity — especially multi-turn and clarifying responses under LLM01/LLM02 — that static rules cannot fully decide. Neither approach alone is sufficient for FedRAMP-oriented LLM deployments.

VI. Discussion

A. Findings

The evaluation demonstrates that GovSecure-LLM provides complementary visibility across both code-level weaknesses and runtime model behavior. Static analysis identified 74 findings across the assessed RAG-based government chatbot, including 5 HIGH, 58 MEDIUM, and 11 LOW severity issues. The highest-risk static findings were concentrated in LLM01 which is associated with prompt injection. These findings highlight that LLM application security risks often arise not only from the model itself, but also from the surrounding orchestration, retrieval, logging, and integration layers.

Dynamic adversarial testing showed stronger runtime resilience. Across 144 OWASP LLM Top 10 probes, the deployed endpoint achieved an 86.1% confirmed pass rate, with 124 PASS results, 19 UNCERTAIN results, one automated FAIL, and zero technical errors. After triage, the single automated FAIL appeared to be a likely false positive triggered by legitimate public-service contact information rather than successful reproduction of the adversarial payload. This indicates that runtime controls such as guardrails, output filtering, application-layer checks, and cloud security layers mitigated several risks that were statically observable in the codebase.

The results also confirm that SAST and DAST are not interchangeable for LLM applications. SAST surfaced architectural and code-level risks that may not be immediately exploitable at runtime, such as missing tenant-scoped retrieval filters, loose similarity thresholds, logging gaps, and missing token-budget controls. DAST, in contrast, validated how the deployed system actually behaved under adversarial prompts and identified areas where automated grading remained uncertain, especially around prompt injection, sensitive information disclosure, streamed responses, and helpful-but-ambiguous model replies. Together, the two approaches provide a more complete assessment than either approach alone.

> A key finding is that prompt injection remains the most difficult category to fully automate. LLM01 had the lowest DAST pass rate at 70.0%, driven primarily by eight UNCERTAIN responses and one likely false-positive automated FAIL rather than confirmed exploitation. This suggests that many government-service interactions sit in a gray area where the system may provide clarifying, policy-related, or public-information responses without explicit refusal language. Automated grading therefore requires attack-specific assertions, response-context awareness, and manual triage for ambiguous cases.

B. Practical Implications

For government LLM deployments, the results suggest that security evaluation should be integrated across the full development lifecycle rather than performed only as a late-stage red-team exercise. GovSecure-LLM's IDE extension and CI/CD pipeline integration support shift-left detection of risky coding patterns, missing controls, and insecure configuration choices before they reach production. This is especially important for RAG systems, where vulnerabilities may emerge from retrieval logic, tenant isolation, embedding workflows, prompt construction, and downstream tool invocation.

The findings also show the importance of combining static findings with runtime validation. High-severity SAST findings should not be interpreted as confirmed exploitation on their own, but they provide an important remediation backlog and help identify architectural weaknesses before attackers can chain them with runtime behavior. Conversely, a strong DAST pass rate does not eliminate the need to fix underlying code-level gaps, especially in regulated government environments where future model changes, prompt updates, data changes, or integration changes may alter system behavior.

The automated compliance mapping capability is particularly useful for public-sector deployments. Mapping findings to OWASP LLM Top 10, NIST AI RMF, MITRE ATLAS, FedRAMP, and related frameworks creates a common language for engineering, security, compliance, and leadership teams. This can reduce the manual effort required to translate technical findings into audit-ready evidence, remediation priorities, and control-alignment documentation.

> The evaluation further suggests that government LLM systems should treat UNCERTAIN results as a managed risk category rather than as noise. In this assessment, UNCERTAIN responses were concentrated in cases involving ambiguous model behavior, streamed outputs, and helpful public-service responses. These cases should feed a human-in-the-loop review workflow, with refined assertions and domain-specific grading criteria added over time. This creates a feedback loop where each assessment improves future automated detection accuracy.

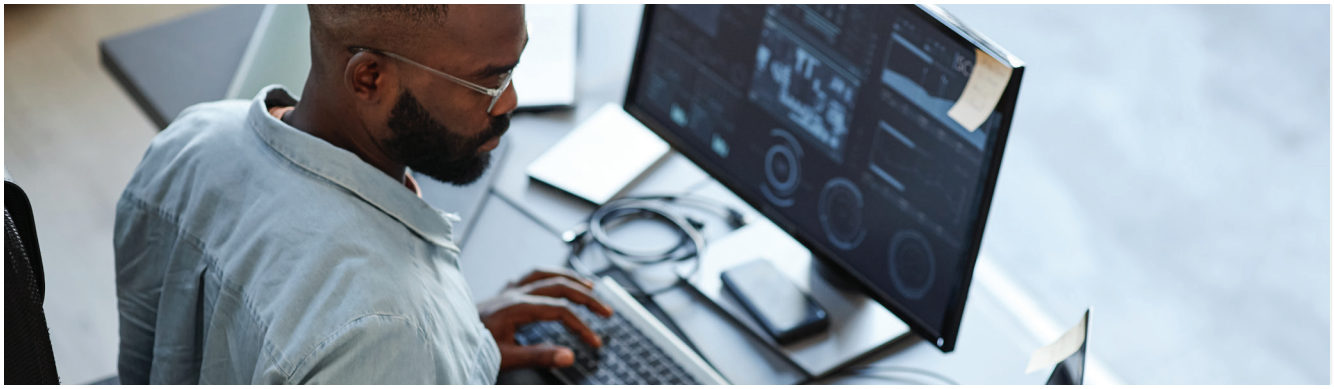
C. Limitations

This evaluation was performed on a single production-representative RAG-based government chatbot, and the results may not generalize to all LLM application architectures, model providers, or deployment patterns. Systems using agentic tool execution, autonomous workflows, multimodal inputs, or more complex cross-system integrations may expose additional risks not fully represented in this assessment.

The DAST evaluation used 144 OWASP-aligned probes against the deployed endpoint, while the broader GovSecure-LLM probe library contains 526+ adversarial tests across 19 attack categories. Therefore, the reported runtime results reflect the selected evaluation suite rather than exhaustive coverage of all possible adversarial behaviors. Additional testing with the full probe library, multi-turn attacks, adaptive mutation, and longer-running abuse scenarios may reveal further weaknesses.

Automated grading remains an inherent limitation for LLM security testing. Model responses can be nuanced, streamed, partially compliant, or context-dependent. Pattern-based assertions may misclassify legitimate government information as unsafe, while broad refusal-language detection may miss subtle policy violations or data leakage. The likely false-positive automated FAIL observed in this study reinforces the need for human validation, attack-specific assertions, and continuous grader calibration.

> Finally, the evaluation focused primarily on application-layer and model-interaction security. It did not independently assess all infrastructure controls, identity and access management policies, production monitoring coverage, organizational processes, or formal FedRAMP authorization evidence. As a result, the findings should be interpreted as security assessment evidence that supports compliance readiness, not as a complete compliance certification.



VII. Conclusion

This paper presented GovSecure-LLM, a security scanning framework designed for LLM applications in government contexts. The framework combines static analysis, dynamic adversarial testing, LLM-powered grading, and automated compliance mapping to address the distinct risks introduced by RAG systems, prompt-driven behavior, and government-service use cases. Unlike general-purpose LLM security tools, GovSecure-LLM emphasizes public-sector threat scenarios, tenant isolation, regulated data handling, and mapping to frameworks such as OWASP LLM Top 10, NIST AI RMF, MITRE ATLAS, FedRAMP, and related compliance requirements.

The evaluation showed that GovSecure-LLM can identify both code-level and runtime risks in a production-representative government chatbot. Static analysis detected 320 findings across OWASP LLM categories, with the highest-risk themes involving insecure output handling, vector and embedding weaknesses, and resource-consumption controls. Dynamic adversarial testing demonstrated strong runtime resilience, with an 86.1% confirmed pass rate, zero technical errors, and no confirmed exploit after triage of the single automated FAIL. The comparison between SAST and DAST confirms that both methods are necessary: SAST reveals latent design and implementation weaknesses, while DAST validates actual deployed behavior under adversarial conditions.

> Overall, the results support the central claim that LLM security in government systems requires a layered assessment approach. Secure deployment cannot rely solely on model guardrails, traditional application scanners, or one-time red-team exercises. Instead, it requires continuous scanning across development, CI/CD, and runtime validation, supported by compliance-aware reporting and human review of ambiguous cases.

VIII. Future Work

Future work will extend GovSecure-LLM in several directions. First, the full 526+ probe library will be applied across a broader set of government LLM applications, including systems with agentic workflows, tool invocation, voice interfaces, and multimodal inputs. This will help evaluate how well the framework generalizes across different architectures and service domains.

Second, the grading pipeline will be improved through more precise attack-specific assertions, better handling of streamed responses, and calibrated LLM-based judgment for ambiguous outputs. Particular focus will be placed on reducing false positives in public-service contexts where legitimate government information may resemble sensitive or risky content.

Third, future versions will strengthen adaptive and multi-turn testing capabilities. This includes adversarial mutation, chained prompt injection, session-memory abuse, retrieval poisoning over multiple turns, and tool-use escalation scenarios. These capabilities are important as government LLM systems increasingly move from simple question-answering toward agentic service delivery.

Finally, GovSecure-LLM will expand compliance evidence generation by producing richer audit artifacts, remediation traceability, and control-level reporting aligned with FedRAMP, NIST AI RMF, MITRE ATLAS, EU AI Act, and emerging AI security overlays. This will help bridge the gap between technical LLM security testing and the documentation required for regulated public-sector deployment.

References

[1] Office of Management and Budget, "2025 Federal Agency AI Use Case Inventory," GitHub repository, 2025. [Online]. Available: https://github.com/ombegov/2025-Federal-Agency-AI-Use-Case-Inventory

[2] IDC Government Insights, "Government AI Agent Adoption Survey," 2026.

[3] S. Perez and A. Ribeiro, "Security challenges in large language model deployments," *IEEE Security & Privacy,* vol. 23, no. 2, pp. 45-52, 2025.

[4] OWASP Foundation, "OWASP Top 10 for LLM Applications Version 2025," November 18, 2024. [Online]. Available: https://owasp.org/www-project-top-10-for-large-language-model-applications/

[5] FedRAMP Program Management Office, "FedRAMP AI Prioritization Initiative," August 2025. [Online]. Available: https://www.fedramp.gov/ai/

[6] National Institute of Standards and Technology, "SP AI-RMF Rev. 5.2.0 and Control Overlays for Securing AI Systems," August 2025. [Online]. Available: https://csrc.nist.gov/Projects/cosais

[7] European Parliament, "Regulation (EU) 2024/1689: Artificial Intelligence Act," *Official Journal of the European Union,* 2024.

[8] NVIDIA, "Garak: LLM Vulnerability Scanner v0.15.0," May 2026. [Online]. Available: https://github.com/nvidia/garak

[9] Microsoft, "PyRIT: Python Risk Identification Toolkit v0.13.0," April 2026. [Online]. Available: https://github.com/microsoft/pyrit

[10] Promptfoo, "LLM Testing and Red Team Framework," 2026. [Online]. Available: https://promptfoo.dev/

[11] National Institute of Standards and Technology, "AI Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, January 2023.

[12] MITRE Corporation, "ATLAS: Adversarial Threat Landscape for AI Systems v5.6.0," May 2026. [Online]. Available: https://atlas.mitre.org/

[13] K. Hackett, B. Scholz, and A. Payne, "Bypassing LLM Guardrails: An Empirical Analysis of Evasion Attacks against Prompt Injection and Jailbreak Detection Systems," in *Proc. First Workshop on LLM Security (LLMSEC), ACL,* August 2025.