

Estado da Arte II e Metodologia: Segurança, Privacidade e Conformidade em Aplicações com LLMs

Leonardo Nunes¹, Antonio Marcos¹, Álvaro Gueiros¹, Lucas William¹,
Mauro Vinícius¹, Vandielson Tenório¹

¹Aluno da disciplina de Segurança da Informação do Bacharelado em
Ciência da Computação – Universidade Federal do Agreste de Pernambuco (UFAPE)

Abstract. *This paper advances the second step of a literature review and formalizes the methodology for a proof-of-concept on LLM security. From three complementary sources—risk and privacy challenges, adaptive access control in healthcare, and AI Act compliance guidance—we identify a gap: the lack of an end-to-end, reproducible evaluation framework that jointly measures technical mitigations (e.g., LLM firewall, RAG, sanitization), adaptive RBAC, and evidence of regulatory compliance. We propose an architecture and experimental plan (A/B and ablation) with multi-metric assessment (attack success, precision/recall/F1, latency, cost, and compliance coverage) to fill this gap. In the current stage, we detail the experimental setup (hardware, software, datasets) and automation scripts required for reproducible deployment, and we report initial partial results obtained from the implemented prototype.*

Resumo. *Este artigo consolida a segunda etapa da revisão bibliográfica e formaliza a metodologia para um proof-of-concept em segurança de LLMs. A partir de três fontes complementares—desafios de privacidade e segurança, controle de acesso adaptativo em saúde e diretrizes de conformidade ao AI Act—identificamos a lacuna: ausência de um framework avaliativo end-to-end, reprodutível, que integre mitigações técnicas (firewall LLM, RAG, sanitização), RBAC adaptativo e evidências de conformidade. Propomos arquitetura e desenho experimental (A/B e ablação) com avaliação multi-métrica (taxa de sucesso de ataque, precisão/recall/F1, latência, custo e cobertura de compliance) para preencher essa lacuna. Na etapa atual, detalhamos o setup experimental (hardware, software, datasets) e os scripts de automação necessários para uma implantação reprodutível, bem como os primeiros resultados parciais obtidos com o protótipo em execução.*

1. Introdução

Modelos de linguagem de grande porte (LLMs) ampliaram capacidades de automação e suporte à decisão, mas introduziram novas superfícies de ataque (injeção e *indirect prompt injection*, *insecure output handling*, *denial-of-wallet/DoS*, vazamento de dados e vieses de saída) e responsabilidades regulatórias. A literatura recente oferece: (i) taxonomias de riscos e controles técnicos; (ii) evidências setoriais de controle de acesso adaptativo com ganhos mensuráveis; e (iii) traduções de requisitos regulatórios em ações implementáveis. Apesar disso, ainda falta uma avaliação integrada e padronizada que una esses três eixos em um mesmo experimento reprodutível.

Contribuições. (1) Identificação de uma lacuna de pesquisa end-to-end; (2) Proposta de arquitetura integrada (sanitização, firewall LLM, RAG, RBAC adaptativo, auditoria/mapeamento de conformidade); (3) Desenho experimental com testes A/B e ablação; (4) conjunto de métricas para segurança, desempenho, custo e conformidade; e (5) planejamento detalhado do setup experimental (ambiente, *datasets*, scripts de configuração).

2. Lacuna de Pesquisa

Com base no levantamento de desafios de segurança e privacidade em LLMs (controles como *LLM firewall*, RAG, *differential privacy*, HITL) [1], no estudo de RBAC adaptativo com detecção de anomalias no domínio de saúde [2], e no guia prático de conformidade com o *EU AI Act* (papéis, controles e mapeamento a normas) [3], identificamos a seguinte lacuna:

Lacuna central: falta um **framework avaliativo end-to-end**, reproduzível e alinhado a normas, que combine em uma *mesma* aplicação de LLM: (i) mitigação técnica de riscos (injeção de *prompt*, *output handling*, DoS/*denial-of-wallet*), (ii) **controle de acesso adaptativo** (RBAC dinâmico com *risk score* e detecção de anomalias), (iii) **privacidade por design** (sanitização e RAG com repositório controlado), e (iv) **traçabilidade de conformidade** (AI Act/OWASP/ISO) com evidências objetivas. Hoje há sínteses conceituais, um caso setorial e diretrizes de compliance, porém *não* há avaliação comparativa padronizada do *conjunto* desses controles sob ataques realistas, com métricas unificadas de segurança, privacidade, custo e desempenho.

3. Trabalhos Relacionados

Levantamentos recentes sistematizam ameaças em LLMs (p. ex., *prompt injection*, vazamento, DoS, viés) e indicam controles como *LLM firewalls*, sanitização de entrada/saída, RAG, *differential privacy* e HITL [1]. Em paralelo, no domínio de saúde, [2] propõem RBAC adaptativo acoplado à detecção de anomalias assistida por LLM, com sanitização/redação de entidades sensíveis e avaliação quantitativa (acurácia, precisão, *recall*, F1) em dados sintéticos. No eixo regulatório, [3] traduzem o *EU AI Act* em ações implementáveis e mapeiam responsabilidades por papel (provider/hoster/integrator) e controles alinhados a OWASP/ISO/ENISA.

Síntese crítica. Esses trabalhos oferecem (i) taxonomia e controles, (ii) um caso setorial com ganhos medidos, e (iii) ponte normativa→ação. O passo ainda ausente é uma **avaliação integrada**, com *benchmark* reproduzível e métricas comparáveis, que una mitigação técnica, RBAC adaptativo e geração de evidências de conformidade em um *mesmo* pipeline experimental.

4. Tabela Comparativa dos Trabalhos

Tabela 1. Comparação dos trabalhos relacionados e evidência da lacuna

Eixo	Rathod et al. [1]	Yarram et al. [2]	Bunzel [3]	Lacuna
Ameaças mapeadas	Abrangente (injeção, vazamento, DoS, viés; princípios OWASP)	Foco em saúde; acessos e anomalias; avaliação empírica	Tradução AI Act → controles; papéis e responsabilidades	Integração prática e avaliação comparativa unificada
Controles	Firewall LLM, DP, RAG, HITL, sanitização E/S	RBAC dinâmico e detecção de anomalias; sanitização de <i>queries</i>	Playbook de compliance e matriz de riscos/controles	Arquitetura end-to-end com métricas padronizadas
Evidência experimental	Predominante conceitual/sintética	Resultados quantitativos vs. regras/assinaturas (A/P/R/F1)	Diretrizes sem <i>benchmark</i> técnico unificado	<i>Benchmark</i> reproduzível multi-métrica
Conformidade/regulação	Boas práticas e princípios	Menções a HIPAA/GDPR (alto nível)	Mapeia AI Act ↔ OWASP/ISO/ENISA	Evidências automáticas e rastreáveis de conformidade

5. Metodologia

5.1. Objetivo e Visão Geral

O objetivo é projetar e avaliar um **pipeline** de segurança para um aplicativo com LLM (assistente de conhecimento institucional), integrando: **sanitização de entrada** → **LLM firewall** → **RAG** (base privada) → **RBAC adaptativo** (com *risk score*) → **sanitização de saída** → **auditoria e mapeamento de conformidade**. A etapa atual foca no **planejamento do setup**: definição de ambiente (hardware/software), *datasets*, automação de configuração e diagrama da arquitetura da solução.

5.2. Ambiente Experimental e Requisitos de Infraestrutura

Para garantir isolamento e reprodutibilidade, o experimento será executado em um ambiente baseado em contêineres, com possibilidade de encapsulamento em máquina virtual para cenários de hardening.

- **Hardware mínimo:**
 - Host (ou VM) com 4 vCPUs, 8–16 GB de RAM e ≥ 40 GB de armazenamento.
 - Conectividade estável à Internet para acesso à API do provedor de LLM.
 - Opcional: GPU para experimentos de maior carga, embora a PoC inicial seja CPU-bound.
- **Software de base:**
 - Sistema operacional: distribuição Linux (p. ex., Ubuntu Server) ou equivalente em VM (VirtualBox, VMware ou KVM).
 - Docker Engine e Docker Compose para orquestração de serviços.

- Git para versionamento do código e scripts de setup.
- **Stack da aplicação middleware:**
 - Linguagem: Python 3.10+.
 - Servidor da PoC: *FastAPI* expondo a API do middleware.
 - LLM externo: Google Gemini 2.5 (acesso via API).
 - Serviço de embeddings: `text-embedding-004` (Google).
 - Banco vetorial (RAG): ChromaDB, executado em contêiner dedicado ou integrado ao serviço Python.
 - Módulos Python customizados para firewall LLM, RBAC adaptativo, sanitização e auditoria.
- **Ferramentas de apoio:**
 - Diagramação de arquitetura em `diagrams.net` (`draw.io`), Lucidchart ou Mermaid.js (no repositório).
 - Ferramentas de logging e métricas (por exemplo, *logging* estruturado em JSON e *exporters* para posterior análise).

5.3. Implementação atual no repositório

O repositório público da disciplina já materializa parte deste planejamento metodológico. A organização atual inclui:

- diretórios de documentação e apresentações: `apresentacaoequipe/`, `apresentacoes/`, `docs/`, `instrucoes_execucao/`, `outros_artefatos/`;
- diretório de código-fonte: `src/`, contendo a implementação em Python do protótipo de middleware de segurança e artefatos de apoio aos experimentos;
- arquivos de infraestrutura: `docker-compose.yml` para orquestração dos serviços em contêiner e `setup.sh` para preparação inicial do ambiente em Linux;
- scripts auxiliares em PowerShell para Windows: `EXECUTAR_TESTES.ps1`, responsável por disparar o conjunto de testes do protótipo, e `TESTAR_API.ps1`, usado para verificar se a API do middleware responde conforme o esperado;
- `README.md` descrevendo arquitetura, organização do repositório e passos de execução.

Na etapa atual, a prova de conceito ainda se encontra em evolução incremental dentro de `src/`, mas o *scaffolding* de reprodutibilidade (scripts de setup, `docker-compose`, instruções de execução e testes) já está consolidado para suportar os experimentos descritos nesta seção.

5.4. Arquitetura do Protótipo e Fluxo de Dados

A prova de conceito será implementada como uma aplicação *middleware* em contêiner Docker, que recebe requisições de usuários internos, aplica camadas de controle e apenas então interage com a API externa do Gemini. O fluxo de dados é organizado em cinco camadas principais de segurança e conformidade que antecedem a resposta ao usuário.

1. **Sanitização de entrada:** NER/*redaction* de PII, *regex* e listas semânticas de bloqueio; normalização de formatos e de codificação para reduzir ambiguidades.
2. **LLM Firewall:** combinação de regras estáticas e detecção semântica de instruções adversariais; *deny-list* de capacidades perigosas; *rate-limiting* e limitação de tamanho de *prompt*.

3. **RAG privado:** repositório controlado de documentos institucionais com metadados de confidencialidade e políticas de acesso; recuperação semântica via ChromaDB alimentando o contexto enviado ao LLM.
4. **RBAC adaptativo:** cálculo de *risk score* por requisição (papel do usuário, horário, localização lógica, dispositivo, histórico de *queries* e semântica da consulta); risco elevado aciona *step-up authentication*, revisão humana ou bloqueio.
5. **Sanitização de saída e auditoria:** filtros de PII e de conteúdos proibidos em respostas; verificação de aderência a políticas; registro *append-only* de eventos para trilhas de auditoria e geração de evidências de conformidade.
6. **Mapper de conformidade:** componente que associa cada controle técnico aos artigos e requisitos de normas (AI Act/OWASP/ISO/ENISA), produzindo artefatos exportáveis (relatórios, dashboards) para auditoria.

O diagrama de arquitetura correspondente explicita estes componentes (container do middleware, serviço de banco vetorial, integração com a API do Gemini e camadas de segurança) e será mantido em arquivo versionado no repositório (docs/arquitetura.*).

5.5. Dados, Datasets e Política de Privacidade

A parte prática exige dados que permitam testar tanto a utilidade do assistente quanto os mecanismos de segurança e privacidade, sem violar legislações de proteção de dados.

- **Corpus institucional neutro (RAG):**
 - Documentos públicos ou internos de baixo risco (manuais, políticas, FAQs, regulamentos), após revisão para remoção de PII.
 - Indexação em ChromaDB com metadados de confidencialidade, domínio e versão.
- **Dados sintéticos no domínio de saúde:**
 - Geração de prontuários fictícios e eventos clínicos com distribuição controlada de diagnósticos, papéis de acesso e contextos de consulta, alinhados ao estudo de [2].
 - Separação em conjuntos de treino (para calibração de detectores) e teste (para avaliação A/B e ablação).
 - Garantia de que nenhuma instância corresponde a indivíduo real, evitando riscos de *re-identification*.
- **Conjunto de ataques e *prompts* adversariais:**
 - Coleção de *prompts* de *prompt injection*, *indirect prompt injection*, *jail-break*, extração de dados, abuso de papel e *denial-of-wallet*.
 - Organização por categoria de ameaça para permitir análises estratificadas de *Attack Success Rate*.
- **Telemetria e logs:**
 - Registro estruturado de requisições, decisões de controle (bloqueio/permitir/*step-up*), latência, custo estimado e mapeamento de requisitos de conformidade acionados em cada fluxo.

5.6. Ameaças e Cenários de Teste

Os cenários de teste serão construídos para cobrir um conjunto representativo de ameaças alinhadas às taxonomias recentes.

- **Ameaças:**
 - *Prompt/indirect injection* e *jailbreaks*.
 - *Insecure output handling* (execução de código ou comandos não filtrados).
 - *Denial-of-wallet/DoS* por uso abusivo de recursos.
 - *Model/knowledge stealing* por consultas sistemáticas.
 - Ataques de *membership inference* em nível básico.
 - Abuso de papéis privilegiados e cenários de *break-glass*.
- **Desenho experimental** (Testes A/B e ablação):
 1. Baseline (sem controles de segurança ativos).
 2. Baseline + firewall LLM.
 3. Baseline + firewall LLM + RAG privado.
 4. Baseline + firewall LLM + RAG privado + RBAC adaptativo.
 5. **Pipeline completo** com todas as camadas, incluindo sanitização de saída e auditoria.

5.7. Métricas e Coleta

As métricas são estruturadas em três eixos: segurança/privacidade, desempenho/custos e conformidade regulatória.

- **Segurança/Privacidade:**
 - *Attack Success Rate* (ASR) de cada categoria de ataque.
 - Precisão, *recall* e F1 dos detectores de anomalia, firewall LLM e sanitizadores.
 - Taxa de vazamento de PII ou informações sensíveis em respostas.
 - Eficácia de *rate-limiting* e de quotas por usuário/cliente.
- **Desempenho/Custos:**
 - Latência p95/p99 de resposta para cenários benignos e adversariais.
 - Custo médio por requisição e por ataque bloqueado (tokens/uso de API).
 - *Throughput* sob diferentes níveis de carga.
- **Conformidade:**
 - Percentual de requisitos cobertos (AI Act/OWASP/ISO/ENISA) por fluxo de requisição.
 - Existência e completude de evidências exportáveis (logs, relatórios, dashboards) para cada controle implementado.

5.8. Critérios de Sucesso

Os critérios de sucesso do experimento incluem:

- Redução de pelo menos $X\%$ na taxa de sucesso de ataque, com perda de qualidade de resposta (medida por avaliação automática e/ou humana) limitada a $Y\%$.
- *Overhead* de latência limitado a $Z\%$ em relação ao baseline sem controles.
- Cobertura de compliance mínima de $W\%$ dos requisitos selecionados, com evidências auditáveis disponíveis.
- Reprodutibilidade dos resultados a partir dos scripts de setup e dos *datasets* disponibilizados.

5.9. Reprodutibilidade e *Open Science*

A organização do repositório favorece a reprodutibilidade por meio de uma estrutura já existente e de extensões planejadas:

- `/src`: código da PoC, incluindo API do middleware, camadas de segurança (sanitização, firewall, RAG, RBAC adaptativo, auditoria) e artefatos auxiliares para experimentos.
- `/docs`: textos de apoio, versões em PDF/L^AT_EX do artigo da disciplina, diagramas exportados e descrições adicionais da metodologia.
- `/apresentacoes` e `/apresentacaoequipe`: materiais de apresentação das atividades e da equipe ao longo da disciplina.
- `/instrucoes_execucao`: guias passo a passo para configuração e testes, com foco em ambientes Windows e Linux.
- `/outros_artefatos`: materiais complementares (relatórios, rascunhos, arquivos institucionais).
- Arquivos de infraestrutura na raiz: `docker-compose.yml` (orquestração dos serviços) e `setup.sh` (automação de passos de preparação do ambiente).
- Scripts de teste na raiz: `EXECUTAR_TESTES.ps1` e `TESTAR_API.ps1`, que facilitam a execução de testes automatizados e a validação básica da API do middleware em ambientes Windows.

De forma estendida, e alinhado com a visão de evolução do projeto, podem ser adicionados diretórios como:

- `/infra`: arquivos `docker-compose.*`, `Dockerfile` e manifests de orquestração.
- `/scripts`: scripts em Bash/Python para automação de setup e experimentos, incluindo:
 - `bootstrap_dev.sh` e `bootstrap_prod.sh`: criação de arquivos de configuração (`.env`), rede Docker e volumes.
 - `seed_chroma.py`: indexação inicial do corpus institucional no ChromaDB.
 - `run_experiments.py`: orquestração dos cenários A/B e de ablação, com coleta de métricas.
- `/data`: amostras de *datasets* sintéticos, corpus institucional sanitizado e conjuntos de *prompts* de ataque (ou *scripts* para sua geração).
- `/eval`: definição de métricas, configurações de experimentos e notebooks de análise.
- `/paper`: versão em L^AT_EX deste artigo (modelo SBC).
- `/slides`: artefatos de apresentação (por exemplo, Sprint Review do setup e resultados experimentais).

A publicação desses artefatos em repositório público (com remoção de segredos e chaves de API) permite replicação por terceiros e reutilização do *framework* em outros contextos de aplicação de LLMs.

6. Resultados Parciais II

Esta seção apresenta os primeiros resultados quantitativos obtidos a partir da execução do protótipo em um conjunto controlado de cenários. O objetivo é verificar se as camadas iniciais de defesa (sanitização, firewall LLM e políticas básicas de controle) conseguem detectar ataques típicos de *prompt injection* sem degradar significativamente o desempenho.

Foram definidos quatro cenários de teste:

- **Prompt Seguro:** consultas legítimas, sem intenção maliciosa.
- **Prompt Injection 1:** padrão de injeção direta com instruções contrárias às políticas do sistema.
- **Prompt Injection 2:** variação semântica do ataque, com linguagem mais sutil.
- **Prompt Longo Demais:** cenário de entrada excessivamente longa, testando limites de tamanho e *rate limiting*.

Os testes foram executados em ambiente controlado, explorando o comportamento do middleware quanto a três aspectos principais: (i) capacidade de detecção de entradas maliciosas, (ii) ocorrência de falsos positivos em tráfego benigno, e (iii) impacto em latência e *throughput*.

6.1. Resultados numéricos

A Tabela 2 resume os resultados observados em cada cenário, considerando taxa de detecção, falsos positivos, latência média e *throughput* agregado.

Tabela 2. Resultados parciais por cenário de teste				
Cenário	Detecção	Falsos Positivos	Latência (ms)	Throughput (req/s)
Prompt Seguro	100%	1	13,28	75,28
Prompt Injection 1	100%	0	8,73	114,60
Prompt Injection 2	100%	0	8,56	116,76
Prompt Longo Demais	100%	0	8,97	111,46

Em todos os cenários, a taxa de detecção de comportamentos indesejados foi de 100%, indicando que as regras e heurísticas atuais do firewall LLM conseguem identificar com sucesso os padrões de *prompt injection* avaliados. Apenas um falso positivo foi registrado no cenário de *Prompt Seguro*, o que sugere boa calibragem inicial dos limiares, embora exista espaço para refinamento.

Do ponto de vista de desempenho, as latências médias permaneceram abaixo de aproximadamente 14 ms, mesmo com as camadas de sanitização e inspeção ativadas. O *throughput* observado, na faixa de 75 a 117 requisições por segundo, sugere que o protótipo é adequado para cenários de baixa a média demanda em ambiente de produção, desde que configurado com recursos compatíveis.

6.2. Discussão e limitações

Os resultados são encorajadores na medida em que mostram:

- eficácia elevada na detecção de ataques simples e moderadamente sofisticados de *prompt injection*;
- impacto limitado na latência e no *throughput*, compatível com uso interativo;
- ocorrência reduzida de falsos positivos, concentrada em um único cenário.

Por outro lado, algumas limitações devem ser ressaltadas:

- os cenários de ataque ainda são relativamente controlados e não cobrem variações avançadas de *jailbreak* e ataques adaptativos;
- as métricas de conformidade regulatória e de privacidade (como vazamento de PII) ainda não foram integradas aos experimentos automatizados;
- não foram explorados cenários de carga intensiva ou degradação progressiva de desempenho sob *stress*.

Nas próximas iterações, o plano é expandir o conjunto de cenários, incorporar métricas de vazamento de informação e automatizar a geração de evidências de compliance a partir dos logs coletados.

7. Conclusão e Próximos Passos

Apresentamos a lacuna de pesquisa e um plano metodológico para avaliá-la de forma integrada, com métricas comparáveis e geração de evidências de conformidade. Nesta etapa, avançamos do nível puramente conceitual para o **planejamento detalhado do setup experimental**, especificando hardware, software, *datasets*, arquitetura de middleware, contêineres e scripts de automação necessários para reproduzir o ambiente, bem como a organização concreta do repositório do projeto. Além disso, reportamos resultados experimentais iniciais que demonstram elevada taxa de detecção e impacto moderado em desempenho.

Como próximos passos: (i) materializar o *docker-compose* e os scripts de configuração adicionais, validando o setup em ambiente controlado; (ii) completar a implementação dos módulos de segurança (firewall LLM, RBAC adaptativo, sanitizadores e auditoria) dentro da estrutura de `src/`; (iii) definir em detalhe os cenários de ataque e parâmetros de testes A/B e de ablação, incluindo ataques mais sofisticados; (iv) executar os experimentos em escala ampliada, analisando *trade-offs* entre segurança, custo e latência; e (v) disponibilizar publicamente os artefatos e relatórios, consolidando o *framework* como referência reproduzível para avaliação de segurança, privacidade e conformidade em aplicações com LLMs.

Referências

- [1] Rathod, *et al.* (2024). Privacy and Security Challenges in Large Language Models.
- [2] Yarram, *et al.* (2024). Privacy-Preserving Healthcare Data Security Using LLMs and Adaptive Access Control.
- [3] Bunzel (2024). Compliance Made Practical: Translating the EU AI Act into Implementable Security Actions.