SAST para detecção de vulnerabilidades em workflows do n8n

João Pedro Schmidt Cordeiro Gustavo Zambonin

11 de novembro de 2025

Resumo

A crescente adoção de plataformas Low-Code/No-Code (LCNC), como o n8n, tem transformado o desenvolvimento de automações empresariais, democratizando a criação de workflows através de interfaces visuais. Entretanto, essa mudança de paradigma introduz uma lacuna crítica de governança: enquanto a plataforma subjacente é protegida por práticas robustas de segurança, os workflows criados pelos usuários, armazenados como arquivos JSON declarativos, não são submetidos ao mesmo rigor de análise de segurança. Este trabalho propõe o desenvolvimento de uma ferramenta de Análise Estática de Segurança de Aplicações (SAST) específica para workflows do n8n, capaz de detectar automaticamente vulnerabilidades como SQL Injection, Command Injection, SSRF, exposição de credenciais, manuseio inseguro de dados e negação de serviço. A metodologia adotada baseia-se em uma estratégia híbrida que combina duas ferramentas do estado da arte: o Agentic Radar, focado em riscos específicos de agentes de IA, e o Semgrep, configurado com regras customizadas para vulnerabilidades tradicionais de aplicações web. Além disso, propõem-se abordagens inovadoras para a criação de regras para o Semgrep, utilizando inteligência artificial e workflows com problemas conhecidos como base para o desenvolvimento e validação de padrões de detecção. A análise detalhada revelou que o Agentic Radar oferece cobertura efetiva de 16,7% das classes de vulnerabilidades identificadas, enquanto o Semgrep apresenta potencial de cobertura de 66,7% quando adequadamente configurado. Os resultados demonstram a viabilidade técnica da abordagem proposta, validando que a análise estática de estruturas declarativas JSON não apenas é possível, mas também eficaz para identificação de padrões inseguros. A solução contribui para elevar o nível de segurança das automações desenvolvidas em organizações brasileiras, auxiliando na conformidade com regulamentações como a LGPD e reduzindo riscos de vazamento de dados e ataques cibernéticos em ambientes de desenvolvimento democratizado.

Palavras-chave: SAST, n8n, Low-Code/No-Code, Segurança de Aplicações, Análise Estática, Vulnerabilidades, Automação de Workflows, Inteligência Artificial, LGPD.

Sumário

1 Introdução

A crescente adoção de plataformas Low-Code/No-Code (LCNC) representa uma transformação fundamental no desenvolvimento de aplicações e automações empresariais. O n8n, uma proeminente plataforma de automação de workflows de código aberto, exemplifica essa mudança ao capacitar equipes técnicas a conectar APIs, bancos de dados e serviços através de um editor visual intuitivo. Esta democratização do desenvolvimento, embora acelere drasticamente a criação de automações, introduz desafios críticos de segurança que este trabalho busca endereçar.

A mudança de um modelo de codificação imperativa tradicional para um modelo de configuração declarativa, onde a lógica é definida visualmente e armazenada como objetos JSON, introduz um novo paradigma para a segurança de aplicações. Existe uma lacuna crítica de governança: o "código-fonte"da aplicação (arquivo JSON do workflow) não está sujeito ao mesmo rigor de segurança que a plataforma na qual é executado. As equipes de segurança não podem mais depender exclusivamente das garantias de segurança do fornecedor; em vez disso, devem estabelecer seus próprios processos de garantia para os ativos criados na plataforma.

1.1 Motivação e Contexto

O interesse por este tema surge de uma necessidade prática identificada no ambiente de trabalho, onde a plataforma n8n é utilizada diariamente. Com o crescimento do número de workflows e a expansão do uso da ferramenta para diferentes áreas organizacionais, observou-se a necessidade crítica de averiguar a segurança dos workflows desenvolvidos. Particularmente preocupante é o fato de que muitos usuários que criam workflows não possuem conhecimento técnico aprofundado em segurança, o que pode resultar no desenvolvimento não intencional de vulnerabilidades dentro do sistema.

Esta experiência prática evidencia a lacuna de governança identificada na literatura, onde o "desenvolvedor cidadão"assume responsabilidades de desenvolvimento sem o treinamento formal necessário para identificar riscos de segurança. A própria n8n implementa práticas de segurança robustas em seu código-fonte, incluindo a utilização de Testes de Segurança de Aplicações Estáticas (SAST) como parte de seu pipeline de Integração Contínua/Entrega Contínua

(CI/CD). Contudo, essa varredura não se estende, nem poderia se estender, aos workflows criados pelos seus usuários. A responsabilidade pela concepção de workflows seguros é explicitamente delegada ao usuário.

O potencial de impacto no contexto brasileiro é particularmente significativo considerando a crescente digitalização de processos empresariais e a adoção de ferramentas de automação no país. A Lei Geral de Proteção de Dados (LGPD) e outras regulamentações de segurança cibernética criam uma demanda específica por ferramentas que possam garantir a conformidade e segurança de automações empresariais. Uma ferramenta SAST especializada para n8n junto com a rápida criação de regras novas, de acordo com a demanda, utilizando inteligência artificial, pode contribuir significativamente para elevar o nível de segurança das automações desenvolvidas por organizações brasileiras, reduzindo riscos de vazamento de dados e ataques cibernéticos que podem resultar em penalidades regulatórias e danos reputacionais.

1.2 Problema Abordado

O principal desafio de segurança no n8n não é uma falha da plataforma em si, mas sim uma falha potencial na implementação do "usuário-desenvolvedor". Os workflows, armazenados como arquivos JSON, podem conter vulnerabilidades críticas que incluem:

- **SQL Injection (SQLi):** Quando entradas controladas pelo usuário são inseridas em consultas SQL sem sanitização adequada, com dados de gatilhos passados diretamente para parâmetros de nós de banco de dados.
- **Command/Code Injection:** Quando dados não confiáveis são usados em comandos executados no servidor através de nós como Execute Command e Code.
- **Proliferação e Encadeamento de Segredos (Secret Sprawl & Chaining):** A dispersão de segredos (chaves de API, tokens) em configurações ou logs, com possíveis vazamentos em respostas de webhooks.
- **Falsificação de Solicitação do Lado do Servidor (SSRF):** Quando entradas controladas pelo usuário determinam URLs em nós HTTP Request sem validação adequada.
- **Manuseio Inseguro de Dados:** Falhas na proteção de dados sensíveis durante armazenamento ou transmissão, incluindo uso de HTTP sem TLS e armazenamento não criptografado.
- **Negação de Serviço (DoS):** Workflows suscetíveis a loops infinitos ou operações intensivas, incluindo uso de nós vulneráveis a CVEs conhecidos.

A natureza declarativa do JSON do n8n, embora abstraia a complexidade do código tradicional, paradoxalmente torna certos tipos de análise estática mais fáceis e precisas. Ele declara explicitamente as relações de fluxo de dados através do array `connections`, permitindo regras de análise de contaminação (taint analysis) com elevado grau de confiança e baixa taxa de falsos positivos.

1.3 Delimitação do Escopo

Este trabalho foca especificamente no desenvolvimento e validação de uma abordagem híbrida para análise estática de segurança de workflows n8n. O escopo está claramente delimitado:

Incluído no escopo:

- Análise estática de arquivos JSON de workflows n8n
- Avaliação detalhada de duas ferramentas do estado da arte: Agentic Radar e Semgrep
- Detecção das seis classes principais de vulnerabilidades identificadas
- Desenvolvimento de metodologia para criação de regras Semgrep utilizando inteligência artificial
- Uso de workflows com vulnerabilidades conhecidas como dataset de validação
- Análise de viabilidade técnica e métricas de performance

Excluído do escopo:

- Implementação completa de uma ferramenta SAST de produção
- Interface gráfica completa para usuários finais
- Análise dinâmica ou execução real de workflows
- Análise de nós customizados da comunidade
- Correção automática de vulnerabilidades detectadas
- Otimização de performance de workflows
- Deploy em ambientes de produção corporativos
- Análise de vulnerabilidades em nível de infraestrutura da plataforma n8n

1.4 Contribuições Esperadas

Este trabalho oferece múltiplas contribuições técnicas, metodológicas e científicas para a área de segurança em plataformas LCNC:

1. **Análise Comparativa do Estado da Arte:** Avaliação detalhada e sistemática do Agentic Radar e Semgrep, documentando cobertura de vulnerabilidades, limitações, e aplicabilidade ao contexto n8n, com métricas quantitativas de cobertura (16,7% e 66,7% respectivamente).

2. **Validação de Viabilidade Técnica:** Demonstração empírica de que análise estática de estruturas declarativas JSON não apenas é possível, mas também eficaz para identificação de padrões inseguros em workflows, validando a premissa fundamental do projeto.

3. **Abordagem Híbrida Inovadora:** Proposta de arquitetura que combina ferramentas especializadas (Agentic Radar para riscos de agentes de IA) com ferramentas genéricas configuráveis (Semgrep para vulnerabilidades tradicionais), maximizando cobertura de segurança.

4. **Metodologia para Criação de Regras com IA:** Desenvolvimento de abordagem inovadora que utiliza inteligência artificial para acelerar e sistematizar a criação de regras de detecção para o Semgrep, reduzindo o overhead de desenvolvimento.

5. **Documentação de Lacunas:** Identificação clara de limitações das ferramentas existentes e oportunidades para desenvolvimento futuro, orientando pesquisas subsequentes na área.

A relevância para a formação profissional está diretamente alinhada com as tendências emergentes do mercado de tecnologia. O desenvolvimento de competências em análise de segurança para plataformas LCNC representa uma especialização altamente demandada no mercado, posicionando o profissional na interseção entre desenvolvimento de baixo código e segurança cibernética. A capacidade de identificar e mitigar riscos em ambientes de desenvolvimento democratizado torna-se um diferencial competitivo crucial à medida que mais organizações adotam estratégias de desenvolvimento cidadão.

2 Fundamentação Teórica e Estado da Arte

Esta seção apresenta a revisão da literatura relevante e análise crítica das ferramentas existentes para análise estática de segurança em plataformas Low-Code/No-Code, com foco específico na plataforma n8n. A análise concentra-se em duas ferramentas principais que representam abordagens distintas ao problema: o Agentic Radar, uma solução moderna baseada em agentes de IA, e o Semgrep, um motor de análise estática clássico e extensível.

2.1 Técnicas de SAST para Plataformas Low-Code/No-Code

O cenário de ferramentas para análise de segurança estática de workflows n8n é emergente e fragmentado, apresentando abordagens distintas que podem ser categorizadas em ferramentas dedicadas e adaptáveis. É crucial ressaltar que, até o momento, **não existe uma abordagem oficial ou solução consolidada para detecção de vulnerabilidades especificamente em workflows do n8n**. Esta lacuna representa tanto um desafio quanto uma oportunidade significativa para pesquisa e desenvolvimento na área.

2.1.1 Abordagem Moderna: Agentic Radar e IA Agêntica

O Agentic Radar, desenvolvido pela SPLX AI, emerge como a ferramenta de código aberto mais proeminente especificamente projetada para análise de segurança de workflows em plataformas de automação, incluindo o n8n [**splxai_n8n_scanning**]. A ferramenta representa uma abordagem moderna ao problema, incorporando capacidades de Inteligência Artificial Agêntica (Agentic AI) para análise de fluxos de trabalho onde agentes de IA interagem com ferramentas, APIs e serviços externos.

A utilização de IA no Agentic Radar manifesta-se em múltiplas dimensões. Primeiro, a ferramenta implementa análise estática consciente do contexto, capaz de interpretar a estrutura JSON de workflows e construir representações gráficas dos fluxos de dados. Segundo, o modo de teste (`test`) utiliza modelos de linguagem (especificamente a API da OpenAI) para executar testes adversariais em tempo de execução, simulando ataques de injeção de prompt, tentativas de extração de informações sensíveis e geração de conteúdo prejudicial [**splxai_medium_scanning**]. Esta capacidade de testar dinamicamente a robustez de sistemas de agentes de IA através de entradas adversariais geradas por outros modelos de linguagem representa uma aplicação sofisticada de IA para segurança.

Terceiro, a ferramenta correlaciona padrões identificados nos workflows com categorias de risco estabelecidas pelo OWASP LLM Top 10, sugerindo a utilização de classificação assistida por IA para mapeamento de vulnerabilidades. Quarto, a geração de relatórios com visualizações

gráficas interativas e recomendações de remediação contextualizadas indica o uso de técnicas de geração de linguagem natural para produzir explicações técnicas compreensíveis.

O foco atual do Agentic Radar concentra-se em riscos emergentes específicos de sistemas de agentes de IA: injeção de prompt, vazamento de informações de identificação pessoal (PII) através de respostas de modelos de linguagem, geração de conteúdo nocivo e disseminação de desinformação. Esta especialização, embora valiosa para workflows do n8n que integram modelos de linguagem e agentes de IA, não cobre vulnerabilidades tradicionais de aplicações web como SQL Injection, SSRF (Server-Side Request Forgery) ou Command Injection. A análise da cobertura de vulnerabilidades revela que a ferramenta oferece detecção efetiva para aproximadamente 16,7% das seis classes principais de vulnerabilidades identificadas na literatura para workflows n8n, limitando-se essencialmente a riscos de IA.

2.1.2 Abordagem Clássica: Semgrep e Análise Baseada em Padrões

O Semgrep, desenvolvido pela Semgrep Inc., representa uma abordagem clássica de análise estática de código, fundamentada em correspondência de padrões consciente de sintaxe [**semgrep_platform**]. Diferentemente de ferramentas baseadas em expressões regulares, o Semgrep realiza parsing do código-fonte em árvores sintáticas abstratas (AST), oferecendo compreensão semântica da estrutura do código. A ferramenta suporta mais de 30 linguagens de programação, incluindo análise nativa de formatos estruturados como JSON e YAML, tornando-a particularmente relevante para a análise de workflows do n8n.

A arquitetura do Semgrep fundamenta-se em uma linguagem de regras declarativa, onde padrões de detecção são especificados em arquivos YAML. Esta abordagem permite que equipes de segurança desenvolvam conjuntos de regras personalizados sem a complexidade de manipulação direta de ASTs ou construção de compiladores [**semgrep_custom_rules**]. No contexto do n8n, regras podem ser desenvolvidas para detectar padrões inseguros específicos: concatenação de entradas não confiáveis em consultas SQL, construção dinâmica de URLs em nós HTTP Request baseada em dados externos, uso de expressões dinâmicas em comandos de execução e exposição de credenciais em configurações.

A capacidade de análise de fluxo de dados (dataflow analysis) do Semgrep é particularmente relevante para a detecção de vulnerabilidades de injeção. A ferramenta permite configurar fontes de dados não confiáveis (sources) — como nós Webhook que recebem entradas externas — e destinos perigosos (sinks) — como parâmetros de consultas SQL ou comandos de execução. O motor de análise então rastreia automaticamente o fluxo de dados contaminados (taint tracking) desde as fontes até os sinks, sinalizando caminhos que não incluem validação ou sanitização adequada [**prototype_pollution**].

A análise de cobertura de vulnerabilidades revela que o Semgrep oferece potencial de cobertura para aproximadamente 66,7% das seis classes principais de vulnerabilidades identificadas (SQL Injection, Command Injection, SSRF e parcialmente Secret Sprawl e Data Handling), com cobertura limitada apenas para detecção de Negação de Serviço (DoS) via análise de ciclos no grafo de execução. Esta capacidade, no entanto, é inteiramente potencial: toda a cobertura depende do desenvolvimento de regras personalizadas específicas para o esquema JSON dos workflows do n8n. Até o momento da presente análise, nenhum conjunto de regras específico para n8n existe no registro comunitário público do Semgrep, representando uma lacuna significativa.

A principal limitação do Semgrep é a ausência de conhecimento nativo do esquema do n8n. A ferramenta não compreende intrinsecamente a semântica dos diferentes tipos de nós (`n8n-nodes-base.webhook`, `n8n-nodes-base.mysql`, `n8n-nodes-base.executeCommand`

a estrutura do array `connections` que define o grafo de fluxo de dados, ou a sintaxe de expressões dinâmicas `{{ $json.body.userInput }}` que representam o principal vetor de propagação de dados contaminados. Esta ausência exige um investimento significativo no desenvolvimento de regras customizadas, estimado entre 40 e 60 horas de trabalho especializado para alcançar cobertura abrangente.

2.2 Explicabilidade em Ferramentas de Análise de Segurança

A explicabilidade refere-se à capacidade de um sistema fornecer insights compreensíveis sobre seu funcionamento e decisões, sendo particularmente crítica em contextos de segurança onde a confiança nas detecções é essencial para a adoção prática das ferramentas [**owasp_source_code_analysis**]. Tanto o Agentic Radar quanto o Semgrep implementam mecanismos de explicabilidade, embora através de abordagens distintas.

O Agentic Radar gera relatórios em formato HTML com visualizações gráficas interativas dos fluxos de trabalho, identificação clara de vulnerabilidades e explicações técnicas detalhadas sobre por que determinado padrão foi classificado como risco. As recomendações de remediação são contextualizadas e acionáveis, frequentemente incluindo exemplos de código corrigido. O alinhamento explícito com o OWASP LLM Top 10 fornece uma taxonomia padronizada que facilita a comunicação de riscos com stakeholders não técnicos. A capacidade de visualização gráfica é particularmente valiosa para workflows complexos, permitindo que analistas de segurança compreendam o contexto das vulnerabilidades dentro do fluxo de dados completo.

O Semgrep, por sua vez, oferece explicabilidade através da transparência das regras. Cada regra é especificada em YAML legível, onde o padrão de detecção, as condições lógicas e os metadados (severidade, mensagens explicativas, sugestões de correção) são explicitamente declarados. Esta transparência permite que desenvolvedores e analistas de segurança compreendam exatamente o que está sendo detectado e por quê. Os relatórios gerados incluem o padrão correspondido, a localização exata no código (número de linha) e a mensagem explicativa definida na regra. O suporte a metadados arbitrários permite enriquecer as detecções com mapeamentos a frameworks de segurança (OWASP LCNC Top 10, CWE), referências a documentação e exemplos de mitigação.

A principal diferença está na natureza da explicabilidade: o Agentic Radar oferece explicações contextuais baseadas na análise holística do fluxo de trabalho, enquanto o Semgrep oferece explicações baseadas em regras que detalham por que um padrão específico é considerado inseguro. Ambas as abordagens são complementares e valiosas em diferentes contextos de uso.

2.3 Segurança e Privacidade em Sistemas de Análise Baseados em IA

A integração de Inteligência Artificial em ferramentas de análise de segurança introduz considerações específicas relacionadas à segurança e privacidade que devem ser cuidadosamente avaliadas, especialmente em contextos corporativos com requisitos rigorosos de conformidade regulatória [**n8n_privacy**].

O Agentic Radar, particularmente em seu modo de teste dinâmico, depende da API da OpenAI para execução de testes adversariais. Esta dependência cria múltiplas preocupações. Primeiro, a necessidade de enviar workflows para serviços externos de terceiros levanta questões de privacidade e confidencialidade: workflows corporativos frequentemente contêm lógica de negócio proprietária, credenciais de acesso a sistemas internos e referências a infraestrutura sensível. A transmissão destes artefatos para serviços externos pode violar políticas de segurança organizacionais ou requisitos de conformidade como SOC 2, ISO 27001 ou a Lei Geral

de Proteção de Dados (LGPD) no contexto brasileiro [**n8n_legal**].

Segundo, a dependência de serviços externos cria questões de disponibilidade e custo operacional. Interrupções na disponibilidade da API da OpenAI impactam diretamente a capacidade de executar análises de segurança, um cenário inaceitável para pipelines de CI/CD críticos onde feedback rápido é essencial. Os custos por chamada de API, embora individualmente pequenos, podem acumular-se significativamente em ambientes corporativos analisando centenas ou milhares de workflows regularmente.

Terceiro, existe a preocupação de segurança da cadeia de suprimentos: a confiança na robustez e segurança dos modelos de linguagem utilizados pela API externa. Vulnerabilidades ou comportamentos inesperados nos modelos podem impactar a confiabilidade das análises, potencialmente produzindo falsos positivos ou, mais gravemente, falsos negativos que deixam vulnerabilidades reais não detectadas.

O Semgrep, como ferramenta de análise estática tradicional, pode operar completamente offline e auto-hospedado, não requerendo transmissão de código para serviços externos. A versão open-source é inteiramente local, endereçando as preocupações de privacidade. No entanto, a plataforma Semgrep Cloud opcional, que oferece dashboards de equipe, agregação de resultados históricos e gerenciamento centralizado de políticas, requer envio de código-fonte para serviços externos operados pela Semgrep Inc. Para organizações com requisitos rigorosos de conformidade, esta limitação restringe o uso à versão auto-hospedada, que carece de algumas funcionalidades de colaboração da plataforma cloud [**gitlab_sast**].

A oportunidade de utilizar IA para geração assistida de regras Semgrep, discutida na próxima subseção, deve ser implementada com consideração cuidadosa destes aspectos de privacidade. Modelos de linguagem podem ser utilizados para sugerir regras baseadas em descrições de vulnerabilidades ou exemplos de padrões inseguros, mas a geração deve, idealmente, ocorrer localmente ou através de modelos de IA auto-hospedados para evitar vazamento de conhecimento proprietário sobre padrões de vulnerabilidade específicos da organização.

2.4 Gaps e Oportunidades

A análise das ferramentas existentes e do estado da arte revela lacunas significativas e oportunidades promissoras para pesquisa e desenvolvimento na área de análise de segurança para workflows do n8n.

2.4.1 Gap Fundamental: Ausência de Solução Específica para n8n

Conforme evidenciado pela análise das seções anteriores, **não existe atualmente uma ferramenta ou abordagem oficial dedicada especificamente à detecção de vulnerabilidades em workflows do n8n**. O Agentic Radar oferece 16,7% de cobertura focada em riscos de IA, mas não endereça vulnerabilidades tradicionais de aplicações web. O Semgrep oferece 66,7% de cobertura potencial, mas requer desenvolvimento significativo de regras customizadas que não existem atualmente no registro público. A lacuna resultante — aproximadamente 83% das necessidades de segurança não cobertas por soluções prontas — representa um risco substancial para organizações que dependem da plataforma n8n para automações críticas de negócio.

Esta lacuna é particularmente preocupante considerando a mudança de paradigma de segurança identificada na literatura: o workflow JSON não é meramente configuração, mas sim o código-fonte da aplicação, e deve ser submetido ao mesmo rigor de análise que código tradicional [**owasp_lcnc_top10**]. A ausência de ferramentas adequadas resulta na perpetuação da lacuna de governança, onde workflows potencialmente inseguros são implantados em produção sem análise automatizada.

2.4.2 Oportunidade: IA para Geração Assistida de Regras Semgrep

Uma oportunidade particularmente promissora identificada nesta análise é a utilização de Inteligência Artificial para geração e otimização assistida de regras de detecção Semgrep específicas para o esquema do n8n. O desenvolvimento manual de regras, estimado em 40 a 60 horas de trabalho especializado, representa uma barreira significativa para a adoção. Modelos de linguagem de grande escala (LLMs) como GPT-4, Claude ou modelos open-source auto-hospedados como Llama, poderiam ser aplicados para:

1. **Geração de regras a partir de descrições em linguagem natural:** Analistas de segurança poderiam descrever vulnerabilidades em linguagem natural (ex.: "detectar concatenação de entrada de webhook em consulta SQL sem sanitização") e o modelo geraria a regra Semgrep correspondente em YAML, incluindo padrões sintáticos, condições lógicas e metadados apropriados.

2. **Otimização de regras existentes para redução de falsos positivos:** Modelos de IA poderiam analisar regras que produzem taxas elevadas de falsos positivos e sugerir refinamentos nas condições lógicas, adicionando verificações contextuais que distinguem padrões genuinamente inseguros de uso legítimo.

3. **Aprendizado de padrões a partir de workflows reais:** Análise de datasets de workflows (como os 2.000+ workflows públicos identificados na literatura [**n8n_workflow_analysis**]) através de técnicas de aprendizado não supervisionado poderia identificar clusters de padrões comuns, distinguindo práticas seguras de inseguras e gerando regras que detectam desvios de padrões seguros estabelecidos.

4. **Manutenção automatizada de regras:** À medida que a plataforma n8n evolui, adicionando novos tipos de nós ou modificando estruturas de parâmetros, modelos de IA poderiam analisar changelogs e documentação de API para sugerir atualizações nas regras existentes ou identificar necessidade de novas regras para cobrir funcionalidades introduzidas.

Esta abordagem híbrida — motor de análise clássico (Semgrep) potencializado por IA generativa para desenvolvimento de conhecimento de segurança — combina as forças de ambos os paradigmas: a confiabilidade, transparência e performance da análise estática baseada em padrões com a flexibilidade, capacidade de adaptação e eficiência de desenvolvimento proporcionadas por modelos de linguagem.

Implementações iniciais desta abordagem poderiam utilizar prompts especializados e few-shot learning, fornecendo aos modelos exemplos de regras Semgrep existentes para outras plataformas e solicitando adaptação para o esquema JSON do n8n. Experimentos com fine-tuning de modelos menores especificamente em datasets de regras Semgrep e esquemas JSON de workflows poderiam melhorar ainda mais a qualidade das regras geradas. A validação das regras geradas por IA através de testes automatizados em workflows conhecidos como seguros e inseguros seria essencial para garantir confiabilidade antes da implantação em ambientes de produção.

2.4.3 Oportunidade: Arquitetura Híbrida para Cobertura Completa

A análise comparativa das ferramentas sugere que uma arquitetura híbrida, combinando o Agentic Radar para riscos de IA (16,7% de cobertura) com Semgrep customizado para vulnerabilidades tradicionais (66,7% de cobertura potencial), poderia alcançar cobertura próxima a 100%

das seis classes principais de vulnerabilidades identificadas. Esta estratégia maximiza as forças de ambas as abordagens enquanto mitiga suas fraquezas individuais: o Agentic Radar oferece análise especializada em agentes de IA com compreensão nativa de workflows, enquanto o Semgrep oferece motor de análise estática poderoso, flexível e de alta performance para detecção de padrões customizados.

A integração destas ferramentas em um pipeline unificado, com agregação de resultados, classificação por severidade e categoria OWASP LCNC Top 10, e geração de relatórios consolidados, representa uma oportunidade de pesquisa significativa para o desenvolvimento de uma solução SAST verdadeiramente abrangente para o ecossistema n8n.

3 Metodologia

[Descrever em detalhes como a PoC será desenvolvida, as técnicas utilizadas, os dados envolvidos, e o processo de validação.]

3.1 Abordagem Metodológica

[Apresentar métodos tradicionais, suas características, vantagens e limitações.]

3.1.1 Abordagens Baseadas em ML/DL

[Descrever técnicas modernas de aprendizado de máquina ou profundo aplicadas ao problema, incluindo arquiteturas de redes neurais relevantes.]

3.2 Explicabilidade em Modelos de IA

[Discutir técnicas de XAI (Explainable AI) relevantes para o projeto, como Grad-CAM, SHAP, LIME, e sua importância no contexto da aplicação.]

3.3 Segurança e Privacidade em Sistemas de IA

[Abordar vulnerabilidades comuns em sistemas de IA, ataques adversariais, e requisitos de conformidade legal (LGPD, GDPR, etc.).]

3.3.1 Ataques Adversariais

[Explicar tipos de ataques (white-box, black-box), técnicas de ataque (FGSM, PGD) e possíveis defesas.]

3.3.2 Conformidade com LGPD

[Detalhar os artigos relevantes da LGPD e como eles se aplicam ao projeto específico.]

3.4 Gaps e Oportunidades

[Identificar lacunas na literatura e oportunidades de pesquisa que justifiquem o desenvolvimento da PoC.]

4 Objetivos e Métricas de Sucesso

[Definir claramente os objetivos gerais e específicos do projeto, estabelecendo métricas quantitativas e qualitativas para avaliar o sucesso da PoC.]

4.1 Objetivo Geral

[Declarar o objetivo principal da PoC de forma clara e mensurável, incluindo metas de performance esperadas.]

4.2 Objetivos Específicos

[Listar objetivos técnicos detalhados, numerados e verificáveis, cobrindo aspectos de implementação, segurança, privacidade e conformidade.]

OE1: *[Objetivo específico 1]*

OE2: *[Objetivo específico 2]*

OE3: *[Objetivo específico 3]*

4.3 Métricas de Avaliação

[Estabelecer critérios objetivos para avaliar o sucesso do projeto.]

4.3.1 Métricas Quantitativas

[Definir métricas numéricas (acurácia, F1-score, latência, throughput, etc.) com valores-alvo e justificativas para as metas estabelecidas.]

Tabela 1: Métricas quantitativas e valores-alvo

Métrica	Definição	Meta

4.3.2 Métricas Qualitativas

[Estabelecer critérios subjetivos de avaliação (usabilidade, clareza de explicações, facilidade de integração) e métodos de avaliação.]

4.3.3 Métricas de Robustez e Segurança

[Definir testes específicos de robustez (transformações, compressões, ataques adversariais) com critérios de aceitação.]

Tabela 2: Testes de robustez obrigatórios

Cenário	Objetivo

4.4 Benchmarks Comparativos

[Identificar sistemas ou trabalhos que servirão como baseline para comparação de performance.]

5 Escopo da Solução e Requisitos

[Detalhar os casos de uso, arquitetura do sistema, requisitos técnicos e composição dos dados.]

5.1 Casos de Uso

[Descrever os principais fluxos de interação com o sistema.]

5.1.1 Caso de Uso Principal

[Detalhar o cenário principal de uso incluindo atores, pré-condições, fluxo principal e fluxos alternativos.]

Ator primário: *[Nome do ator]*

Pré-condições:

- *[Pré-condição 1]*
- *[Pré-condição 2]*

Fluxo principal:

1. *[Passo 1]*
2. *[Passo 2]*
3. *[Passo 3]*

Fluxos alternativos:

- *[Alternativa 1]*
- *[Alternativa 2]*

5.1.2 Casos Excluídos do Escopo

[Listar explicitamente casos de uso que NÃO serão implementados na PoC.]

5.2 Arquitetura Proposta

[Apresentar a arquitetura lógica do sistema, preferencialmente com diagramas, explicando as camadas, componentes e fluxo de dados. Incluir considerações de segurança (defense in depth, privacy by design).]

5.2.1 Componentes Principais

[Descrever em detalhes cada componente da arquitetura (frontend, backend, modelo de ML, sistema de logs, etc.) e suas responsabilidades.]

5.3 Requisitos Técnicos

[Especificar as tecnologias, frameworks e infraestrutura necessários.]

5.3.1 Tecnologias e Frameworks

[Listar em formato de tabela as tecnologias escolhidas para cada camada/componente com justificativas técnicas.]

Tabela 3: Stack tecnológico

Camada	Tecnologia	Justificativa

5.3.2 Infraestrutura Mínima

[Especificar requisitos de hardware (CPU, GPU, RAM, disco) e rede para desenvolvimento, treinamento e deploy.]

5.3.3 Dados de Treinamento e Validação

[Descrever a composição do dataset (quantidade, proporções, fontes), estratégia de divisão (treino/validação/teste) e considerações de privacidade.]

Tabela 4: Composição do dataset

Conjunto	Classe A	Classe B	Total
Treinamento (70%)			
Validação (15%)			
Teste (15%)			
Total			

6 Ameaças, Riscos e Controles

[Aplicar metodologia STRIDE para identificação sistemática de ameaças, analisar riscos específicos de sistemas de ML, e estabelecer controles de mitigação.]

6.1 Superfícies de Ataque

[Enumerar todos os pontos do sistema que podem ser explorados por atacantes.]

6.2 Análise STRIDE

[Apresentar matriz completa de ameaças usando categorias STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) com controles correspondentes.]

Tabela 5: Matriz de ameaças STRIDE

Categoria	Ameaça	Controle
Spoofing		
Tampering		
Repudiation		
Information Disclosure		
Denial of Service		
Elevation of Privilege		

6.3 Ataques Específicos a Modelos de ML

[Detalhar ameaças particulares a sistemas de aprendizado de máquina.]

6.3.1 Evasion Attacks

[Descrever ataques adversariais que tentam enganar o modelo em tempo de inferência e controles de mitigação.]

6.3.2 Data Poisoning

[Explicar riscos de contaminação do conjunto de treinamento e medidas preventivas.]

6.3.3 Model Extraction

[Abordar riscos de roubo de modelo proprietário através de queries e contramedidas.]

6.4 Considerações de Privacidade (LINDDUN)

[Aplicar metodologia LINDDUN para análise sistemática de riscos de privacidade (Linkability, Identifiability, Non-repudiation, Detectability, Disclosure, Unawareness, Non-compliance).]

6.5 Considerações Éticas

[Discutir aspectos éticos do sistema.]

6.5.1 Vieses Algorítmicos

[Identificar riscos de discriminação, estratégias de mitigação e métodos de avaliação de equidade.]

6.5.2 Transparência e Explicabilidade

[Estabelecer compromissos de transparência e como a explicabilidade será implementada.]

6.5.3 Direito de Contestação

[Definir processo para usuários contestarem decisões automatizadas.]

7 Metodologia de Desenvolvimento da PoC

[Detalhar todos os aspectos práticos de implementação: preparação de dados, modelagem, treinamento, implementação e integração.]

7.1 Dados e Preparação

[Descrever origem, geração e preparação dos dados.]

7.1.1 Geração de Dataset Sintético (ou Coleta de Dados Reais)

[Explicar como os dados foram obtidos ou gerados, incluindo scripts, ferramentas e técnicas utilizadas.]

7.1.2 Particionamento e Estratificação

[Detalhar como o dataset foi dividido, garantindo representatividade em todos os subconjuntos.]

7.2 Modelagem e Treinamento

[Explicar escolhas de arquitetura e processo de treinamento.]

7.2.1 Arquiteturas Avaliadas

[Listar modelos candidatos com características técnicas (número de parâmetros, FLOPS, pré-treinamento).]

Tabela 6: Modelos candidatos

Modelo	Parâmetros	FLOPS	Pré-treinamento

7.2.2 Estratégia de Treinamento

[Detalhar hiperparâmetros, fases de treinamento (transfer learning, fine-tuning), técnicas de data augmentation, regularização, função de perda e otimizadores utilizados. Incluir snippets de código relevantes.]

7.2.3 Adversarial Training

[Explicar como exemplos adversariais foram incorporados ao treinamento para aumentar robustez, incluindo código de implementação.]

7.3 Implementação e Integração

[Descrever aspectos práticos de implementação do sistema completo.]

7.3.1 Pipeline de Inferência

[Apresentar código do pipeline completo de processamento, desde entrada até geração de resultados e logs.]

7.3.2 API REST

[Documentar endpoints, formatos de request/response, autenticação e exemplos de uso via curl ou outra ferramenta.]

7.3.3 Containerização

[Apresentar configuração Docker/Docker Compose para reprodutibilidade e deploy.]

8 Avaliação Experimental

[Apresentar protocolo experimental, resultados obtidos, comparações e análises.]

8.1 Protocolo Experimental

[Detalhar configuração de hardware/software e metodologia de avaliação.]

8.1.1 Configuração de Hardware e Software

[Especificar ambiente de execução (GPU, RAM, OS, versões de bibliotecas).]

8.1.2 Repetição e Validação Cruzada

[Explicar como a variância foi tratada (múltiplas execuções, seeds diferentes, intervalos de confiança).]

8.2 Resultados

[Apresentar resultados quantitativos e qualitativos.]

8.2.1 Performance no Conjunto de Teste

[Tabela com métricas principais (F1, Precisão, Recall, AUC-ROC) por modelo, comparando com metas estabelecidas.]

Tabela 7: Métricas principais por modelo

Modelo	F1-Score	Precisão	Recall	AUC-ROC
Meta Alvo				

8.2.2 Matriz de Confusão

[Apresentar matriz de confusão com análise de falsos positivos e falsos negativos, identificando padrões nos erros.]

8.2.3 Performance por Tipo/Categoria

[Detalhar desempenho em subcategorias do problema para identificar pontos fortes e fracos.]

8.2.4 Latência e Throughput

[Reportar métricas de tempo de processamento e vazão, com breakdown por etapa do pipeline.]

8.3 Testes de Robustez

[Avaliar resiliência do sistema.]

8.3.1 Transformações Geométricas e Degradações

[Tabela mostrando degradação de performance sob transformações comuns (compressão, ruído, rotação).]

8.3.2 Ataques Adversariais

[Resultados de testes contra ataques adversariais (FGSM, PGD) com diferentes intensidades.]

8.4 Análise de Vieses

[Avaliar equidade do sistema.]

8.4.1 Equidade entre Subgrupos

[Tabela comparando performance entre diferentes subgrupos para identificar possíveis discriminações.]

8.5 Explicabilidade: Análise Qualitativa

[Avaliar qualidade das explicações geradas (Grad-CAM ou outra técnica) através de avaliação por especialistas ou métricas objetivas. Incluir exemplos visuais.]

9 Discussão

[Interpretar resultados, discutir limitações, comparar com estado-da-arte e analisar implicações práticas.]

9.1 Alcance dos Objetivos

[Avaliar se os objetivos estabelecidos foram atingidos, comparando resultados com metas.]

9.2 Limitações Identificadas

[Discutir honestamente as limitações do trabalho.]

9.2.1 Limitações Técnicas

[Identificar restrições técnicas (dependência de qualidade de entrada, escopo limitado, dataset sintético, etc.).]

9.2.2 Limitações de Segurança

[Reconhecer vulnerabilidades residuais e aspectos de segurança que precisam ser melhorados para produção.]

9.3 Comparação com Estado-da-Arte

[Tabela comparativa com trabalhos relacionados e sistemas comerciais, analisando posicionamento da PoC.]

9.4 Trade-offs Segurança vs. Usabilidade

[Discutir compromissos feitos entre segurança, privacidade e usabilidade, justificando escolhas.]

9.5 Implicações Práticas

[Analisar impactos práticos do trabalho.]

9.5.1 Para Usuários/Stakeholders

[Discutir como o sistema pode ser utilizado na prática, benefícios e necessidades de treinamento.]

9.5.2 Para Políticas Públicas

[Refletir sobre implicações para regulamentação, padronização e investimentos públicos.]

10 Conclusões e Próximos Passos

[Sintetizar contribuições, propor trabalhos futuros e fornecer recomendações.]

10.1 Síntese das Evidências

[Resumir as principais conclusões demonstradas pela PoC de forma objetiva.]

10.2 Contribuições

[Listar contribuições metodológicas, técnicas e científicas do trabalho.]

10.3 Trabalhos Futuros

[Propor extensões e melhorias.]

10.3.1 Curto Prazo (3-6 meses)

[Melhorias incrementais possíveis em 3-6 meses.]

10.3.2 Médio Prazo (6-12 meses)

[Desenvolvimento de funcionalidades adicionais ou pilotos em 6-12 meses.]

10.3.3 Longo Prazo (1-2 anos)

[Visão de evolução do sistema em 1-2 anos (escala, novas funcionalidades, federação).]

10.4 Recomendações

[Fornecer orientações práticas para stakeholders interessados em adotar tecnologia similar.]

11 Checklist de Conformidade (LGPD/Ética/Segurança)

11.1 Lei Geral de Proteção de Dados (LGPD)

- ☐ **Minimização de dados (Art. 6º, III):**
  - ☐ *[Item de verificação 1]*
  - ☐ *[Item de verificação 2]*
- ☐ **Finalidade específica (Art. 6º, I):**
  - ☐ *[Item de verificação 1]*
- ☐ **Transparência (Art. 6º, VI):**
  - ☐ *[Item de verificação 1]*
- ☐ **Segurança (Art. 46):**
  - ☐ *[Item de verificação 1]*
- ☐ **Direito à explicação (Art. 20):**
  - ☐ *[Item de verificação 1]*
- ☐ **Não discriminação (Art. 6º, IX):**
  - ☐ *[Item de verificação 1]*

11.2 Segurança da Informação

- ☐ **Criptografia:**
  - ☐ *[Item de verificação 1]*
- ☐ **Autenticação e Autorização:**
  - ☐ *[Item de verificação 1]*
- ☐ **Isolamento:**
  - ☐ *[Item de verificação 1]*
- ☐ **Auditoria:**
  - ☐ *[Item de verificação 1]*
- ☐ **Resiliência:**
  - ☐ *[Item de verificação 1]*

11.3 Ética e IA Responsável

- ☐ **Explicabilidade:**
  - ☐ *[Item de verificação 1]*
- ☐ **Equidade:**
  - ☐ *[Item de verificação 1]*
- ☐ **Accountability:**
  - ☐ *[Item de verificação 1]*
- ☐ **Contestação:**
  - ☐ *[Item de verificação 1]*
- ☐ **Transparência pública:**
  - ☐ *[Item de verificação 1]*

11.4 Ações Pendentes para Produção

- ☐ *[Ação pendente 1]*
- ☐ *[Ação pendente 2]*

Agradecimentos

[Agradecer aos orientadores, colegas, instituições e fontes de financiamento quando aplicável.]

A Detalhes de Implementação

A.1 Estrutura de Diretórios

[Apresentar árvore completa de diretórios do projeto.]

A.2 Comandos para Reprodução

[Lista step-by-step de comandos para setup, treinamento, avaliação e deploy do sistema.]

B Exemplos Adicionais de Resultados

[Figuras adicionais mostrando casos de sucesso e falha, exemplos visuais de classificações corretas e incorretas.]

C Código-fonte Completo

[Link para repositório público com código-fonte e licença de uso.]