Trabalho Final - SAST para detecção de vulnerabilidades em workflows do n8n

João Pedro Schmidt Cordeiro

Tópicos Especiais em Aplicações Tecnológicas I (UFSC-INE5448)

15 de outubro de 2025

Sumário

jctechconnections@gmail.com src/

1 Introdução

A crescente adoção de plataformas Low-Code/No-Code (LCNC) representa uma transformação fundamental no desenvolvimento de aplicações e automações empresariais (N8N.IO, 2025b). A 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 (N8N.IO, 2025e). Este trabalho propõe o desenvolvimento de uma ferramenta de Análise Estática de Segurança de Aplicações (SAST) específica para workflows da plataforma n8n, focando na detecção automatizada de vulnerabilidades de segurança em configurações JSON de workflows.

O interesse por este tema surge de uma necessidade prática identificada no ambiente de trabalho, onde utilizo a plataforma n8n diariamente. Com o crescimento do número de workflows e a expansão do uso da ferramenta para diferentes áreas organizacionais, observei 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 experiência prévia com as tecnologias envolvidas no projeto fundamenta-se no uso cotidiano da plataforma n8n para automação de processos empresariais, proporcionando compreensão prática da estrutura JSON de workflows e dos padrões de configuração mais comuns (N8N.IO, 2025a). O conhecimento em análise de código e ferramentas de segurança, combinado com experiência em desenvolvimento de scripts para análise de dados estruturados, oferece a base técnica necessária para implementar soluções de SAST. Além disso, a familiaridade com conceitos de DevSecOps e integração de ferramentas de segurança em pipelines de desenvolvimento, adquirida através de projetos acadêmicos e profissionais, complementa o conjunto de habilidades requerido para o desenvolvimento da ferramenta proposta.

A viabilidade de implementação do MVP é assegurada pela estratégia de escopo limitado e pela reutilização inteligente das ferramentas de estado da arte identificadas na pesquisa. Conforme demonstrado na revisão da literatura, tanto o Agentic Radar quanto o Semgrep possuem arquiteturas comprovadamente funcionais para análise de workflows n8n (SPLXAI, 2025). O Agentic Radar já demonstrou a viabilidade técnica de analisar o JSON de workflows da n8n, construir gráficos de fluxo de dados e gerar relatórios de segurança (SPLXAI, 2025), enquanto o Semgrep oferece um motor de análise estática poderoso e flexível para detecção de padrões customizados (SEMGREP, 2025b). A estratégia proposta consiste em utilizar essas duas ferramentas em conjunto: o Agentic Radar cobrirá vulnerabilidades específicas de IA de agentes, enquanto regras customizadas no Semgrep expandirão a cobertura para vulnerabilidades tradicionais de aplicações web (SQL Injection, SSRF, etc.) que estão além do escopo atual do Agentic Radar. Esta abordagem híbrida permite focar no desenvolvimento de regras específicas e integração, ao invés de construir um motor de análise do zero, tornando o projeto viável dentro do prazo intensivo de 6 semanas estabelecido. O cronograma comprimido exige uma abordagem ágil e focada, priorizando funcionalidades essenciais e mantendo o escopo bem definido.

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. O Brasil, como um dos maiores mercados de tecnologia da América Latina, tem experimentado um crescimento substancial na adoção de plataformas LCNC, especialmente em setores como serviços financeiros, e-commerce e governo digital. A Lei Geral de Proteção de Dados (LGPD) e outras regulamentações de segurança cibernética no país criam uma demanda específica por ferramentas que possam garantir a conformidade e segurança de automações empresariais. Uma ferramenta SAST especializada para n8n 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.

A relevância para a formação profissional está diretamente alinhada com as tendências emergentes do mercado de tecnologia. Conforme evidenciado na revisão da literatura, as plataformas Low-Code/No-Code estão experimentando um crescimento exponencial, com a n8n sendo uma das principais representantes deste paradigma (N8N.IO, 2025e). Este crescimento resulta em uma expansão significativa da superfície de ataque, especialmente considerando que muitos usuários que criam workflows carecem de conhecimento técnico e formação em segurança para prevenir vulnerabilidades básicas e conhecidas (COMMUNITY, 2025b). 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. Além disso, o projeto proporciona experiência prática em tecnologias de ponta como SAST, análise de contaminação (taint analysis) e integração de ferramentas de segurança em pipelines de CI/CD, competências essenciais para cargos em DevSecOps e engenharia de segurança. 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.

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.

2 Revisão da Literatura e Estado da Arte

Esta seção sintetiza as descobertas da pesquisa acadêmica e análise de soluções existentes na área de SAST para plataformas LCNC.

2.1 A Mudança de Paradigma de Segurança: Do Código Imperativo aos Workflows Declarativos

A ascensão das plataformas de desenvolvimento Low-Code/No-Code (LCNC), como a n8n, representa uma transformação fundamental na forma como as aplicações e automações são construídas (OWASP, 2025e). A n8n, uma proeminente plataforma de automação de workflows de código aberto, capacita equipes técnicas a conectar APIs, bancos de dados e serviços através de um editor visual intuitivo, baseado em nós (N8N.IO, 2025e). Essa abordagem acelera drasticamente o desenvolvimento, permitindo a criação de automações complexas que, de outra forma, exigiriam um esforço de programação significativo (GEEKY-GADGETS, 2025). No entanto, essa mudança de um modelo de codificação imperativa tradicional (como em Python ou Java) 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.

Neste novo modelo, o perímetro de segurança desloca-se da aplicação principal da plataforma para as configurações criadas pelo usuário. A própria n8n GmbH 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) (N8N.IO, 2025i). Essas medidas são projetadas para proteger o motor da n8n contra vulnerabilidades (N8N.IO, 2025h). 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, conforme detalhado na documentação oficial (N8N.IO, 2025g).

Isso cria uma lacuna de governança crítica. O "código-fonte"da aplicação, que neste contexto é o arquivo JSON que define o workflow, não está sujeito ao mesmo rigor de segurança que a plataforma na qual ele é executado. O usuário, muitas vezes um "desenvolvedor cidadão"ou um profissional de TI focado na automação de processos, assume o papel de desenvolvedor de aplicações, mas pode não possuir o treinamento formal em segurança necessário para identificar e mitigar riscos complexos (MICHEL, 2025). A consequência é que uma plataforma inerentemente segura ainda pode ser usada para construir e implantar automações perigosamente inseguras. A existência de vulnerabilidades publicadas, como Negação de Serviço (DoS) através de requisições malformadas (FEEDLY, 2025) ou o potencial para Falsificação de Solicitação do Lado do Servidor (SSRF) (PORTSWIGGER, 2025), não reside em falhas no motor principal da n8n, mas na maneira como os workflows podem ser construídos e explorados.

Portanto, 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". As equipes de segurança não podem mais depender exclusivamente das garantias de segurança do fornecedor, como relatórios SOC 2 mencionados em sua documentação legal (N8N.IO, 2025f). Em vez disso, elas devem estabelecer seus próprios processos de garantia para os ativos criados na plataforma. O foco da análise de segurança deve mudar do fornecedor da plataforma para o usuário da plataforma, tratando o workflow JSON como um artefato de código de primeira classe que requer seu próprio processo SAST dedicado para fechar essa lacuna de governança. Esta é uma redefinição fundamental do modelo de responsabilidade compartilhada para plataformas LCNC.

2.2 A Anatomia de um Workflow n8n: Uma Análise da Superfície de Ataque

Para aplicar os princípios de SAST aos workflows da n8n, é imperativo primeiro dissecar sua estrutura fundamental. Cada workflow, independentemente de sua complexidade, pode ser exportado como um único arquivo JSON. Este arquivo não é meramente uma configuração; ele é a representação estática e completa da lógica da aplicação, contendo todos os nós, seus parâmetros e as conexões de fluxo de dados. Para fins de análise de segurança, este JSON é o código-fonte. Uma ferramenta SAST projetada para n8n deve ser capaz de analisar este artefato para identificar padrões inseguros.

A estrutura do JSON de um workflow é composta por vários componentes chave que são cruciais para a análise estática:

- **Array nodes:** Define cada etapa do processamento individual. Uma ferramenta SAST deve iterar sobre este array e analisar cada objeto de nó.
- **Array connections:** Define o gráfico de fluxo de dados do workflow, especificando qual saída de nó se torna a entrada de outro.
- **Expressões dinâmicas:** Como `{{ $json.body.userInput }}`, representam entradas dinâmicas de dados, muitas vezes origem de vulnerabilidades.

A natureza declarativa do JSON da n8n, embora abstraia a complexidade do código tradicional, paradoxalmente torna certos tipos de análise estática mais fáceis e precisos. As ferramentas SAST tradicionais lutam para construir gráficos de fluxo de controle precisos, o que frequentemente resulta em altos índices de falsos positivos. Em contraste,

o modelo da n8n declara explicitamente as relações de fluxo de dados, permitindo regras de análise de contaminação com elevado grau de confiança e baixa taxa de falsos positivos.

2.3 O Estado da Arte do SAST para n8n: Ferramentas e Metodologias

O cenário de ferramentas para análise de segurança estática de workflows n8n é emergente, mas já apresenta abordagens distintas que podem ser categorizadas em ferramentas dedicadas, adaptáveis e soluções baseadas na própria plataforma.

2.3.1 Ferramentas Dedicadas: Agentic Radar

Atualmente, a ferramenta de código aberto mais proeminente e especificamente projetada para escanear workflows n8n é a **Agentic Radar** (SPLXAI, 2025). Ela analisa o JSON do workflow, identifica nós, mapeia conexões e correlaciona padrões com riscos conhecidos, gerando relatórios visuais e detalhados (SPLXAI, 2025). No entanto, seu foco atual está em riscos de IA de agentes, não cobrindo nativamente vulnerabilidades web tradicionais como SQL Injection ou SSRF.

2.3.2 Ferramentas Adaptáveis: Semgrep e Conjuntos de Regras Personalizados

Uma abordagem flexível envolve o uso de motores SAST genéricos, como o **Semgrep** (SEMGREP, 2025b), que suporta análise de JSON e YAML e permite regras personalizadas (SEMGREP, 2025a). Equipes de segurança podem desenvolver conjuntos de regras específicos para o esquema da n8n — por exemplo, detectar concatenações de strings em consultas SQL dentro de nós de banco de dados.

2.3.3 Soluções na Plataforma e Comunitárias

A própria n8n pode ser utilizada para construir ferramentas de segurança, um conceito de "segurança como workflow". Templates como o **WebSecScan** demonstram auditorias automatizadas (N8N.IO, 2025j). A comunidade também tem contribuído com análises manuais, identificando anti-padrões comuns (por exemplo, webhooks públicos sem autenticação) (REDDIT, 2025).

Esses três grupos de soluções — Agentic Radar, Semgrep e esforços comunitários — representam um ecossistema nascente, mas fragmentado. Há uma oportunidade clara para uma ferramenta que combine o motor de análise consciente da estrutura da n8n com um conjunto de regras abrangente para vulnerabilidades tradicionais.

2.4 Análise Aprofundada de Vulnerabilidades para Workflows n8n

Esta seção detalha as seis principais classes de vulnerabilidades e suas manifestações no contexto da n8n.

2.4.1 SQL Injection (SQLi)

Ocorre quando entradas controladas pelo usuário são inseridas em consultas SQL sem sanitização adequada. No n8n, isso se dá quando dados de gatilhos são passados diretamente para o parâmetro `Query` de nós de banco de dados (COMMUNITY, 2025a). A detecção estática envolve identificar concatenações de expressões `{{...}}` dentro de queries e rastrear sua origem até nós de entrada.

2.4.2 Injeção de Código / Comando

Surge quando dados não confiáveis são usados em comandos executados no servidor. Os nós **Execute Command** e **Code** são os principais vetores (N8N.IO, 2025c). A ferramenta deve rastrear o fluxo de dados para parâmetros como `command` e sinalizar construções dinâmicas baseadas em entradas externas.

2.4.3 Proliferação e Encadeamento de Segredos (Secret Sprawl & Chaining)

Consiste na dispersão de segredos (chaves de API, tokens) em configurações ou logs (GITGUARDIAN, 2025). A análise estática deve procurar padrões de segredos em parâmetros e variáveis, bem como rastrear possíveis vazamentos em respostas de webhooks.

2.4.4 Falsificação de Solicitação do Lado do Servidor (SSRF)

Ocorre quando entradas controladas pelo usuário determinam URLs em nós HTTP Request (N8N.IO, 2025d). A ferramenta deve detectar construções dinâmicas de URLs originadas em fontes externas e verificar se passam por etapas de validação (PORTSWIGGER, 2025).

2.4.5 Manuseio Inseguro de Dados

Abrange falhas na proteção de dados sensíveis durante armazenamento ou transmissão (N8N.IO, 2025k). A ferramenta deve sinalizar uso de HTTP sem TLS, armazenamento não criptografado e fluxos de dados sensíveis para destinos inseguros.

2.4.6 Negação de Serviço (DoS)

Relaciona-se a workflows suscetíveis a loops infinitos ou operações intensivas. A detecção envolve identificar estruturas cíclicas no grafo de conexões e uso de nós vulneráveis a CVEs conhecidos (NIST, 2025).

2.5 Um Contexto Mais Amplo: Mapeando Vulnerabilidades da n8n para o OWASP LCNC Top 10

As vulnerabilidades da n8n refletem riscos sistêmicos descritos pelo **OWASP LCNC Top 10** (OWASP, 2025e). O mapeamento associa, por exemplo, SQL Injection e Command Injection à categoria LCNC-SEC-06 (Injection Handling Failures) (OWASP, 2025b), Secret Sprawl e Data Handling a LCNC-SEC-08 (Data and Secret Handling Failures) (OWASP, 2025d), SSRF a LCNC-SEC-05 (Security Misconfiguration) (OWASP, 2025a) e DoS a LCNC-SEC-07 (Vulnerable and Untrusted Components) (OWASP, 2025c).

Este alinhamento fornece uma linguagem comum para comunicar riscos e priorizar esforços de mitigação em programas corporativos de segurança.

2.6 Uma Estrutura para um Ciclo de Vida de Desenvolvimento de Workflow Seguro (SWDL)

A segurança no n8n deve evoluir da detecção para a prevenção, adotando um ciclo de vida de desenvolvimento seguro de workflows.

2.6.1 Guarda-corpos Automatizados (Shift Left)

Integrar a análise SAST de workflows em pipelines CI/CD é essencial (GITLAB, 2025). Os workflows devem ser escaneados automaticamente a cada commit ou versão.

2.6.2 Padrões de Design e Desenvolvimento Seguro (Um Checklist para Criadores)

Entre as melhores práticas: validação de entradas, não codificar credenciais, princípio do menor privilégio, tratamento robusto de erros, uso de HTTPS e cautela com componentes da comunidade (REDDIT, 2025).

2.6.3 Auditoria e Monitoramento Contínuo

A segurança requer monitoramento contínuo, auditorias periódicas e análise de logs (OWASP, 2025f). Deve-se estabelecer metas internas (como 0 webhooks públicos) e complementar a análise estática com observabilidade em tempo de execução.

3 Análise do Agentic Radar

3.1 Visão Geral e Propósito

O Agentic Radar emerge como uma ferramenta de código aberto, desenvolvida pela SPLX AI, projetada especificamente para aprimorar a segurança e a transparência em sistemas de inteligência artificial baseados em agentes. Seu propósito central é capacitar equipes de segurança e desenvolvimento a visualizar, analisar e fortalecer fluxos de trabalho onde agentes de IA interagem com ferramentas, APIs e serviços externos. A ferramenta realiza análise estática de código-fonte para mapear dependências, identificar vulnerabilidades e fornecer recomendações práticas de remediação alinhadas com frameworks de segurança estabelecidos, particularmente o OWASP LLM Top 10.

O posicionamento do Agentic Radar no mercado reflete uma tendência emergente: o reconhecimento de que sistemas de agentes de IA introduzem uma nova categoria de riscos de segurança que transcende as vulnerabilidades tradicionais de aplicações web. A ferramenta é, atualmente, a solução de código aberto mais proeminente e especificamente projetada para escanear workflows da plataforma n8n. Esta capacidade de analisar o JSON de workflows do n8n, construir grafos de fluxo de dados e gerar relatórios de segurança estabelece a viabilidade técnica da abordagem de análise estática para workflows declarativos.

No contexto da lacuna de governança identificada anteriormente, o Agentic Radar representa uma resposta parcial ao desafio fundamental: quando o "código-fonte"da aplicação é um arquivo de configuração JSON, as ferramentas tradicionais de SAST (Análise Estática de Segurança de Aplicações) não são suficientes. A ferramenta valida a premissa de que a análise estática pode ser aplicada efetivamente a modelos declarativos, oferecendo insights arquitetônicos valiosos para o projeto proposto, mesmo que seu escopo atual não cubra todas as categorias de vulnerabilidades necessárias para uma solução completa de segurança para o n8n.

3.2 Arquitetura e Funcionalidades Principais

A arquitetura do Agentic Radar fundamenta-se em um motor de análise estática capaz de interpretar o código-fonte de múltiplos frameworks de agentes de IA e extrair a estrutura lógica dos fluxos de trabalho. A ferramenta implementa quatro funcionalidades centrais que constituem seu núcleo operacional. Primeiro, a visualização de fluxos de trabalho de agentes gera representações gráficas interativas que ilustram como agentes, ferramentas e processos interagem dentro do sistema de IA, proporcionando uma compreensão clara do fluxo de decisões e informações. Segundo, a identificação de ferramentas externas detecta automaticamente todas as integrações com APIs, serviços e outras ferramentas utilizadas nos fluxos de trabalho, detalhando suas conexões e dependências.

Terceiro, o mapeamento de vulnerabilidades de IA analisa a arquitetura dos agentes em busca de falhas de segurança, correlacionando padrões identificados com riscos conhecidos e alinhando-os ao OWASP LLM Top 10. Quarto, o fornecimento de recomendações de remediação oferece orientações práticas e acionáveis para corrigir as vulnerabilidades detectadas, auxiliando na melhoria contínua da postura de segurança. Esta abordagem holística reflete uma compreensão sofisticada de que a segurança efetiva requer não apenas detecção, mas também orientação para a correção.

O Agentic Radar opera através de dois modos complementares, implementados como comandos CLI distintos. O comando `scan` realiza a análise estática do código, examinando o código-fonte em busca de fluxos de trabalho de agentes e gerando relatórios detalhados em formato HTML com gráficos interativos e explicações técnicas. Este modo não requer a execução do código, focando exclusivamente na estrutura declarada. O comando `test`, por outro lado, permite testes de vulnerabilidades em tempo de execução, executando o agente com entradas adversariais simuladas para identificar vulnerabilidades críticas como injeção de prompt, vazamento de informações de identificação pessoal (PII), geração de conteúdo nocivo e disseminação de notícias falsas. Este modo dual oferece tanto prevenção (análise estática) quanto validação (testes dinâmicos), um modelo arquitetural relevante para o projeto proposto.

A compatibilidade com frameworks inclui suporte nativo para LangGraph, CrewAI, OpenAI Agents e Autogen, com planos de expansão futura. A instalação via pip é simples (`pip install agentic-radar`), com dependências extras disponíveis para frameworks específicos. No entanto, alguns requisitos técnicos são restritivos: a ferramenta requer Python $>= 3.10$ e < 3.13, e o modo `test` depende da configuração da variável de ambiente `OPENAI_API_KEY`, criando uma dependência de serviços externos pagos. A integração com pipelines de CI/CD é viável, permitindo a execução automática de análises de segurança a cada commit ou pull request, um requisito essencial identificado na seção sobre ciclo de vida de desenvolvimento seguro de workflows.

3.3 Cobertura de Vulnerabilidades para n8n

A análise da cobertura de vulnerabilidades do Agentic Radar em relação às seis classes principais de vulnerabilidades revela uma discrepância significativa entre o foco da ferramenta e as necessidades específicas da plataforma n8n. Das seis categorias de vulnerabilidades detalhadas anteriormente, o Agentic Radar demonstra cobertura limitada, concentrando-se predominantemente em riscos associados a sistemas de agentes de IA, em vez de vulnerabilidades tradicionais de aplicações web.

Para SQL Injection, a primeira classe crítica identificada, o Agentic Radar não oferece detecção nativa. A ferramenta não foi projetada para rastrear o fluxo de dados de entradas não confiáveis até a construção de queries SQL, nem para identificar padrões de concatenação de strings em parâmetros de nós de banco de dados. Isso constitui uma

lacuna crítica, considerando que SQL Injection foi mapeada à categoria LCNC-SEC-06 do OWASP LCNC Top 10 e representa um dos vetores de ataque mais prevalentes em workflows do n8n que integram bancos de dados. Similarmente, para Command/Code Injection, o Agentic Radar não oferece cobertura adequada para detectar o uso de dados não confiáveis em nós como Execute Command ou Code do n8n.

Quanto a Exposição de Credenciais (Secret Sprawl) e Encadeamento de Credenciais (Chaining), existe um potencial de cobertura parcial. O foco do Agentic Radar em analisar integrações com ferramentas externas e APIs poderia, teoricamente, identificar padrões onde credenciais são expostas em configurações ou logs. No entanto, a ferramenta não implementa a detecção específica de padrões de segredos (como regex para chaves de API ou tokens) nem rastreia o vazamento de credenciais em respostas de webhooks ou nós HTTP Request, limitações significativas considerando que a exposição de credenciais foi mapeada à categoria LCNC-SEC-08.

Para SSRF (Server-Side Request Forgery), o Agentic Radar não oferece cobertura direta. A ferramenta não detecta a construção dinâmica de URLs em nós HTTP Request baseada em entradas externas, nem verifica se as URLs passam por etapas de validação antes de serem utilizadas. Isso representa outra lacuna crítica, dado que o SSRF foi mapeado à categoria LCNC-SEC-05 (Security Misconfiguration). Em relação ao Manuseio Inseguro de Dados (Insecure Data Handling), há alguma cobertura através da análise do fluxo de dados da IA, mas a ferramenta não sinaliza especificamente o uso de HTTP sem TLS, armazenamento não criptografado ou a transmissão de dados sensíveis para destinos inseguros.

Finalmente, para DoS (Denial of Service), o Agentic Radar oferece cobertura muito limitada. Embora possa identificar algumas estruturas no grafo de fluxo, não realiza análise específica de ciclos infinitos no grafo de conexões dos workflows do n8n, nem correlaciona o uso de nós com CVEs conhecidos de DoS. O foco da ferramenta está em vulnerabilidades de agentes de IA: injeção de prompt, vazamento de PII através de respostas de modelos de linguagem, geração de conteúdo prejudicial e disseminação de desinformação. Estas são categorias importantes e emergentes, mas ortogonais às seis classes de vulnerabilidades tradicionais de aplicações web que constituem o foco do projeto proposto.

Esta análise revela que o Agentic Radar oferece cobertura efetiva para aproximadamente 16,7% das classes de vulnerabilidades-alvo (com cobertura parcial apenas para Exposição de Credenciais), demonstrando que, embora seja uma ferramenta valiosa para riscos de IA, é insuficiente como solução autônoma para a segurança abrangente dos workflows do n8n.

3.4 Avaliação Frente às Métricas de Sucesso

A avaliação do Agentic Radar em relação às métricas quantitativas estabelecidas para o projeto revela limitações significativas quando aplicada especificamente ao contexto de workflows do n8n. A meta de 85% de precisão e 90% de recall, com um F1-score de 87%, não pode ser diretamente avaliada, pois as métricas publicadas da ferramenta focam em vulnerabilidades de agentes de IA, e não nas seis classes de vulnerabilidades tradicionais de aplicações web estabelecidas como objetivo. Não há dados disponíveis sobre a precisão da ferramenta especificamente para a detecção de SQL Injection, Command Injection, SSRF ou outras categorias relevantes para os workflows do n8n, tornando impossível validar o atingimento desta métrica central.

Quanto à cobertura de vulnerabilidades, a meta estabelecida de 100% das seis classes identificadas claramente não é alcançada. Conforme detalhado na subseção anterior, o Agentic Radar oferece cobertura nativa para aproximadamente 16,7% das categorias-alvo, concentrando-se em riscos de agentes de IA em vez de vulnerabilidades tradicionais. Esta discrepância fundamental inviabiliza o uso isolado da ferramenta para atender aos requisitos do projeto proposto.

Em relação à performance da análise, embora não existam benchmarks publicados especificamente para workflows do n8n, a arquitetura de análise estática da ferramenta sugere a capacidade de processar arquivos JSON de workflows dentro dos parâmetros estabelecidos (máximo de 30 segundos para workflows de até 50 nós, análise de 100 workflows por hora). A natureza declarativa do JSON do n8n e a experiência documentada da SPLX AI em analisar esses workflows indicam a viabilidade técnica de uma performance adequada. A taxa de falsos positivos, estabelecida em até 15%, permanece desconhecida para o contexto específico dos workflows do n8n, uma vez que as métricas disponíveis se referem à detecção de vulnerabilidades de agentes de IA em frameworks como LangGraph e CrewAI.

As métricas qualitativas de usabilidade são parcialmente atendidas. O Agentic Radar oferece uma interface CLI intuitiva com comandos bem definidos (`scan` e `test`) e documentação disponível no repositório GitHub. A execução básica requer poucos comandos, alinhando-se ao critério de "execução em até três comandos". No entanto, a necessidade de configurar variáveis de ambiente (particularmente `OPENAI_API_KEY` para o modo `test`) adiciona complexidade e dependência de serviços externos pagos.

A clareza dos relatórios é um ponto forte da ferramenta. O Agentic Radar gera relatórios em formato HTML com visualizações gráficas interativas dos fluxos de trabalho, identificação clara de vulnerabilidades e recomendações de remediação. O alinhamento com o OWASP LLM Top 10 fornece uma classificação padronizada de riscos, análoga ao mapeamento com o OWASP LCNC Top 10 estabelecido como requisito. As explicações técnicas são detalhadas e as sugestões de mitigação são acionáveis, atendendo aos critérios qualitativos de clareza. No entanto, vale notar que o formato de severidade pode diferir do sistema de quatro níveis proposto (Crítica, Alta, Média, Baixa).

Quanto à integração, o Agentic Radar demonstra excelente compatibilidade com formatos JSON e capacidade de integração em pipelines de CI/CD. A ferramenta pode ser executada como parte de fluxos de trabalho automatizados, escaneando o código a cada commit ou pull request, seguindo as melhores práticas de desenvolvimento seguro. Esta característica alinha-se perfeitamente com o requisito de "barreiras de proteção automatizadas"(shift left), identificado

como essencial para a segurança proativa.

3.5 Limitações e Lacunas Identificadas

A análise detalhada do Agentic Radar revela limitações estruturais que restringem sua aplicabilidade como solução autônoma para a segurança de workflows do n8n. Primeiramente, o foco em frameworks de IA específicos (LangGraph, CrewAI, OpenAI Agents, Autogen) significa que a ferramenta não possui conhecimento nativo do esquema JSON dos workflows do n8n. Embora a SPLX AI tenha demonstrado a capacidade de adaptar o Agentic Radar para analisar workflows do n8n, esta funcionalidade não parece fazer parte da versão de código aberto disponível publicamente no GitHub, representando uma lacuna crítica para a adoção direta.

O escopo de vulnerabilidades constitui a limitação mais significativa. O Agentic Radar foi projetado especificamente para identificar riscos associados a sistemas de agentes de IA: injeção de prompt (prompt injection), vazamento de informações pessoais através de respostas de modelos de linguagem, geração de conteúdo prejudicial ou ofensivo e disseminação de notícias falsas ou desinformação. Estas categorias, embora crescentemente relevantes em workflows do n8n que integram modelos de linguagem e agentes de IA, não cobrem as vulnerabilidades tradicionais de aplicações web que constituem a maioria dos riscos em workflows típicos do n8n. A ausência de detecção para SQL Injection, Command Injection, SSRF e outras categorias do OWASP LCNC Top 10 representa uma lacuna fundamental que inviabiliza o uso isolado da ferramenta.

Os requisitos técnicos também introduzem limitações práticas. A restrição de versão do Python ($>= 3.10$ e < 3.13) pode criar incompatibilidades em ambientes legados ou com outras dependências conflitantes. A dependência da API da OpenAI para o modo de teste (`test`) não apenas gera custos operacionais contínuos, mas também introduz uma dependência da disponibilidade de serviços externos e preocupações de privacidade ao enviar dados de workflows para análise por serviços de terceiros. Esta última consideração é particularmente problemática em contextos corporativos com requisitos rigorosos de conformidade e privacidade de dados.

A ausência de suporte nativo para o esquema JSON específico do n8n significa que a ferramenta não compreende intrinsecamente a semântica dos diferentes tipos de nós (Webhook, HTTP Request, MySQL, Execute Command, etc.), suas relações de conexão declaradas no array `connections`, ou a sintaxe de expressões dinâmicas `{{ $json.body.userInput }}`. Esta compreensão semântica é essencial para uma análise precisa do fluxo de dados contaminados (taint analysis) e para a detecção de padrões inseguros específicos do contexto do n8n. A necessidade de adaptação ou extensão da ferramenta para suportar nativamente estes elementos representa um esforço de desenvolvimento significativo.

Finalmente, a lacuna fundamental identificada é que os workflows do n8n requerem cobertura para vulnerabilidades tradicionais de aplicações web (SQL Injection, SSRF, Command Injection, Exposição de Credenciais, Manuseio Inseguro de Dados, DoS), que não são o foco do Agentic Radar. A ferramenta é excelente para o que foi projetada — segurança de agentes de IA — mas essa especialização a torna complementar, e não um substituto, para uma solução SAST abrangente para o n8n. Esta observação reforça a necessidade de uma estratégia híbrida: combinar o Agentic Radar para riscos de IA com regras customizadas no Semgrep para as vulnerabilidades tradicionais.

3.6 Contribuições para o Projeto Proposto

Apesar das limitações identificadas, o Agentic Radar oferece contribuições valiosas para o projeto proposto que transcendem sua aplicação direta. Primeiramente, a arquitetura da ferramenta serve como inspiração para a visualização de grafos de fluxos de trabalho. A capacidade demonstrada de transformar estruturas declarativas em representações gráficas interativas, que facilitam a compreensão da lógica e a identificação de padrões perigosos, oferece um modelo arquitetural a ser adaptado para o contexto específico dos workflows do n8n. A geração de grafos a partir do array `connections` dos workflows JSON é tecnicamente viável e comprovadamente útil para a análise de segurança.

Segundo, a abordagem de geração de relatórios do Agentic Radar estabelece um padrão de qualidade a ser emulado. Os relatórios em HTML com elementos interativos, explicações técnicas detalhadas, classificação de severidade e recomendações acionáveis de remediação representam exatamente o tipo de resultado que maximiza a utilidade para as equipes de segurança e desenvolvimento. O projeto proposto deve aspirar a uma clareza e profundidade de relatórios similares, adaptando o formato para incluir o mapeamento específico ao OWASP LCNC Top 10, conforme estabelecido nos requisitos.

Terceiro, os padrões de integração com pipelines de CI/CD demonstrados pelo Agentic Radar validam a viabilidade técnica da automação da análise de segurança para workflows. A capacidade de executar escaneamentos automaticamente a cada commit, gerar relatórios padronizados e falhar builds quando vulnerabilidades críticas são detectadas alinha-se perfeitamente com a visão de "barreiras de proteção automatizadas"(shift left), essencial para a segurança proativa. O projeto proposto deve implementar compatibilidade similar com ferramentas populares de CI/CD como GitLab CI, GitHub Actions e Jenkins.

Quarto, a metodologia de mapeamento de vulnerabilidades a frameworks de segurança estabelecidos (OWASP LLM Top 10 no caso do Agentic Radar, OWASP LCNC Top 10 no caso do projeto proposto) oferece um modelo para a comunicação efetiva de riscos. Esta abordagem permite que os resultados da análise sejam contextualizados dentro de taxonomias reconhecidas pela indústria, facilitando a priorização da remediação e a comunicação com stakeholders

não técnicos. O projeto proposto deve implementar o mapeamento explícito de cada vulnerabilidade detectada para a categoria relevante do OWASP LCNC Top 10.

Finalmente, a validação mais importante que o Agentic Radar oferece é a confirmação da viabilidade técnica fundamental do projeto proposto. A ferramenta demonstra que a análise estática de estruturas declarativas (JSON de workflows) não é apenas possível, mas também eficaz para a identificação de vulnerabilidades e a geração de insights acionáveis. O sucesso documentado do Agentic Radar em analisar workflows do n8n, mesmo que focado em riscos de IA, prova que o esquema JSON da plataforma é suficientemente estruturado para permitir uma análise automatizada e robusta.

Esta análise reforça a estratégia híbrida proposta: utilizar o Agentic Radar para a cobertura de vulnerabilidades específicas de agentes de IA (particularmente relevante para workflows do n8n que integram modelos de linguagem e ferramentas de IA), enquanto se desenvolvem regras customizadas no Semgrep para expandir a cobertura às vulnerabilidades tradicionais de aplicações web (SQL Injection, SSRF, Command Injection, Exposição de Credenciais, Manuseio Inseguro de Dados, DoS) que estão além do escopo atual do Agentic Radar. Esta abordagem complementar maximiza a cobertura de segurança ao combinar as forças de ambas as ferramentas, posicionando o projeto proposto para oferecer uma análise SAST verdadeiramente abrangente para o ecossistema n8n.

4 Análise do Semgrep

4.1 Visão Geral e Propósito

O Semgrep emerge como um motor de análise estática de código aberto desenvolvido pela r2c (posteriormente Semgrep Inc.), posicionando-se como uma das ferramentas SAST mais versáteis e amplamente adotadas na indústria de segurança de aplicações (SEMGREP, 2025b). Diferentemente do Agentic Radar, que representa uma ferramenta dedicada a um domínio específico (agentes de IA), o Semgrep constitui uma plataforma adaptável, capaz de analisar mais de 30 linguagens de programação, incluindo suporte nativo para formatos estruturados como JSON e YAML. Esta flexibilidade fundamental torna o Semgrep particularmente relevante para o contexto dos workflows do n8n, onde a análise deve operar sobre arquivos JSON que codificam lógica de aplicação de forma declarativa.

O propósito central do Semgrep é democratizar a análise estática de segurança através de uma linguagem de regras acessível e expressiva, permitindo que equipes de segurança desenvolvam detecções personalizadas sem a complexidade de manipulação direta de árvores sintáticas abstratas (AST) ou construção de compiladores. A ferramenta utiliza correspondência de padrões baseada em sintaxe concreta, onde as regras são escritas em uma sintaxe similar ao código que está sendo analisado, tornando o desenvolvimento de regras intuitivo mesmo para profissionais sem expertise profunda em análise de compiladores. No contexto das plataformas Low-Code/No-Code, essa acessibilidade é crucial: permite que especialistas em segurança que compreendem os riscos do n8n desenvolvam regras de detecção específicas para o esquema JSON da plataforma, mesmo sem conhecimento prévio do Semgrep (SEMGREP, 2025a).

A relevância do Semgrep para o projeto proposto reside precisamente em sua natureza genérica e extensível. Enquanto a ferramenta não possui conhecimento nativo do esquema da n8n, sua capacidade de análise de JSON pode ser configurada para tratar workflows como artefatos de código de primeira classe, submetendo-os ao mesmo rigor de análise aplicado a código-fonte tradicional. Esta característica endereça diretamente a lacuna de governança identificada na introdução: o workflow JSON, historicamente tratado como mera configuração, pode agora ser analisado com a mesma sofisticação que código Python ou Java. O registro comunitário do Semgrep contém mais de 2.000 regras para diversas linguagens e frameworks, validando a maturidade da plataforma e o modelo de desenvolvimento colaborativo de conhecimento de segurança. Embora não existam, até o momento desta análise, conjuntos de regras específicos para o n8n no registro público, a arquitetura do Semgrep e os exemplos existentes de análise de JSON fornecem o modelo e a infraestrutura necessários para o desenvolvimento de tais regras.

O posicionamento do Semgrep no ecossistema de ferramentas SAST para n8n é complementar ao Agentic Radar. Onde o Agentic Radar oferece análise especializada para riscos de agentes de IA com conhecimento implícito de workflows, o Semgrep oferece um motor genérico de alto desempenho que pode ser programado para detectar as vulnerabilidades tradicionais de aplicações web que constituem a maioria dos riscos em workflows típicos do n8n: SQL Injection, Command Injection, SSRF, exposição de credenciais, manuseio inseguro de dados e potenciais vetores de negação de serviço.

4.2 Arquitetura e Funcionalidades Principais

A arquitetura do Semgrep fundamenta-se em um motor de análise estática que opera através de correspondência de padrões consciente de sintaxe, distinguindo-se de ferramentas baseadas em expressões regulares por sua compreensão semântica da estrutura do código (SEMGREP, 2025b). O núcleo da ferramenta realiza parsing do código-fonte em árvores sintáticas abstratas (AST) específicas para cada linguagem, mas abstrai essa complexidade do desenvolvedor de regras, permitindo que padrões sejam escritos na própria sintaxe da linguagem sendo analisada. Para arquivos JSON, o Semgrep trata a estrutura hierárquica de objetos e arrays como elementos sintáticos navegáveis, permitindo queries que especificam caminhos através da estrutura JSON e padrões de valores em localizações específicas.

A linguagem de regras do Semgrep é definida em arquivos YAML, onde cada regra especifica padrões a serem detectados, condições lógicas que combinam múltiplos padrões, e metadados como severidade, mensagens explicativas

e sugestões de remediação (SEMGREP, 2025a). Esta abordagem declarativa torna as regras versionáveis, testáveis e compartilháveis, transformando o conhecimento de segurança em artefatos de infraestrutura como código. A ferramenta implementa quatro funcionalidades centrais que constituem seu núcleo operacional.

Primeiro, a detecção baseada em padrões permite especificar construções de código que devem ser sinalizadas, utilizando operadores como `pattern` (correspondência exata), `pattern-either` (correspondência de qualquer padrão em uma lista) e `pattern-regex` (correspondência baseada em expressões regulares para casos onde a sintaxe estrutural é insuficiente). No contexto do n8n, isso permite detectar, por exemplo, a presença de expressões dinâmicas `{{ $json.body.userInput }}` dentro de parâmetros de consultas SQL, um indicador clássico de vulnerabilidade de SQL Injection.

Segundo, a análise de fluxo de dados (dataflow analysis) rastreia como dados propagam através do código, desde fontes (sources) até destinos perigosos (sinks), uma técnica conhecida como análise de contaminação (taint analysis). O Semgrep permite configurar sources personalizados (como nós Webhook no n8n) e sinks personalizados (como parâmetros de consulta de banco de dados), detectando caminhos de fluxo de dados que conectam entradas não confiáveis a operações perigosas. Esta funcionalidade é essencial para detectar vulnerabilidades de injeção onde a concatenação não ocorre diretamente no sink, mas através de múltiplas transformações intermediárias.

Terceiro, a análise de metavariáveis permite capturar porções de código que correspondem a um padrão e referenciá-las em condições adicionais. Por exemplo, uma regra pode capturar o valor de um parâmetro de URL em um nó HTTP Request e verificar se esse valor é validado contra uma whitelist antes de ser utilizado, detectando potenciais vulnerabilidades SSRF. Quarto, a composição de regras através de operadores lógicos (`pattern-and`, `pattern-not`, `pattern-inside`) permite expressar condições complexas, como "detectar concatenação de strings em queries SQL, exceto quando a string concatenada é uma constante literal".

O Semgrep opera através de múltiplos modos de execução complementares. A interface de linha de comando (CLI), invocada através de `semgrep scan`, permite análise local de arquivos e diretórios, gerando relatórios em formatos texto, JSON ou SARIF (Static Analysis Results Interchange Format), este último compatível com ferramentas de visualização e plataformas de CI/CD (OWASP, 2025f). A integração com pipelines de CI/CD é nativa, com suporte direto para GitHub Actions, GitLab CI, Jenkins e CircleCI, permitindo a execução automática de análises de segurança a cada commit ou pull request. A plataforma Semgrep Cloud opcional oferece agregação de resultados, dashboards de equipe e gerenciamento de políticas, mas introduz considerações de privacidade ao enviar código para serviços externos, uma preocupação relevante para ambientes corporativos com requisitos rigorosos de conformidade.

O suporte a JSON e YAML no Semgrep é particularmente sofisticado, tratando estruturas aninhadas como elementos sintáticos de primeira classe. A ferramenta pode corresponder padrões em caminhos específicos da hierarquia JSON, detectar valores dinâmicos em chaves particulares e navegar arrays de objetos. Para o esquema do n8n, isso significa que regras podem ser escritas para iterar sobre o array `nodes`, identificar nós de tipos específicos (como `"type": "n8n-nodes-base.mysql"`) e inspecionar seus parâmetros em busca de padrões inseguros. A instalação do Semgrep é trivial (`pip install semgrep`), requerendo apenas Python $>=$ 3.8 e produzindo um executável autossuficiente com dependências mínimas. A performance da ferramenta é excepcional, capaz de escanear milhares de arquivos por segundo devido a otimizações no motor de correspondência de padrões e paralelização automática de análises.

4.3 Cobertura de Vulnerabilidades para n8n

A análise da cobertura de vulnerabilidades do Semgrep em relação às seis classes principais de vulnerabilidades identificadas revela um perfil de capacidades significativamente diferente e complementar ao Agentic Radar. É crucial enfatizar que, diferentemente do Agentic Radar que oferece cobertura nativa (embora limitada), a cobertura do Semgrep é inteiramente potencial, dependendo do desenvolvimento de regras personalizadas específicas para o esquema JSON dos workflows do n8n. Esta distinção fundamental posiciona o Semgrep não como uma solução pronta, mas como uma plataforma que deve ser configurada e programada para atender às necessidades específicas do contexto.

Para SQL Injection, a primeira classe crítica de vulnerabilidades, o Semgrep oferece potencial de cobertura ALTO. A ferramenta pode ser configurada para detectar concatenação dinâmica de expressões `{{ ... }}` dentro do parâmetro `parameters.query` de nós de banco de dados como MySQL, PostgreSQL e Microsoft SQL. Uma regra Semgrep pode especificar um padrão que identifica objetos no array `nodes` com `type` correspondente a tipos de nós de banco de dados, e então verificar se o campo `parameters.query` contém strings que incluem a sintaxe de expressão do n8n. Regras mais sofisticadas podem implementar análise de fluxo de dados, rastreando se os dados incorporados na query se originam de fontes não confiáveis como nós Webhook, e se passam por etapas de sanitização antes da incorporação. Esta capacidade endereça diretamente a vulnerabilidade documentada na comunidade do n8n relacionada a escaping inadequado de MySQL (COMMUNITY, 2025a).

Para Command/Code Injection, a segunda classe crítica, o Semgrep também oferece potencial de cobertura ALTO. Os nós **Execute Command** e **Code** do n8n são vetores conhecidos para esta vulnerabilidade (N8N.IO, 2025c). Regras podem ser desenvolvidas para identificar esses tipos de nós e analisar se seus parâmetros críticos (`parameters.command` para Execute Command, `parameters.jsCode` para Code em modo JavaScript) incorporam dados dinâmicos provenientes de entradas externas. A análise de fluxo de dados do Semgrep permite rastrear a propagação de dados desde nós de entrada (Webhook, HTTP Request trigger) até esses parâmetros de execução, sinalizando caminhos de contaminação que não incluem validação ou sanitização intermediária.

Para Exposição de Credenciais (Secret Sprawl) e Encadeamento de Credenciais (Chaining), a terceira classe, o

Semgrep oferece potencial de cobertura MÉDIO. A ferramenta pode implementar regras baseadas em expressões regulares para detectar padrões de segredos hardcoded: chaves de API (padrões como `api_key.*=.*[A-Za-z0-9]{32,}`), tokens de acesso, senhas em texto claro e outros formatos conhecidos de credenciais (GITGUARDIAN, 2025). No entanto, a detecção de encadeamento de credenciais — onde um segredo é obtido de uma fonte e então vazado através de outra, como uma resposta de webhook — requer análise sofisticada de fluxo de dados através do grafo de conexões do workflow. Embora o Semgrep possua capacidades de taint analysis, a natureza declarativa das conexões no n8n (array `connections` que define o grafo separadamente dos nós) torna essa análise complexa e potencialmente além das capacidades nativas da ferramenta sem processamento adicional.

Para SSRF (Server-Side Request Forgery), a quarta classe, o Semgrep oferece potencial de cobertura ALTO. Vulnerabilidades SSRF no n8n ocorrem quando entradas controladas pelo usuário determinam URLs utilizadas em nós HTTP Request (N8N.IO, 2025d). Regras Semgrep podem detectar construção dinâmica de URLs no parâmetro `parameters.url` de nós HTTP Request, identificando padrões onde expressões `{{ ... }}` incorporam dados de fontes externas. A análise pode verificar se a URL passa por etapas de validação (como verificação contra whitelist de domínios permitidos) antes de ser utilizada na requisição. Esta detecção mapeia diretamente à categoria LCNC-SEC-05 (Security Misconfiguration) do OWASP LCNC Top 10 (OWASP, 2025a), e endereça um vetor de ataque bem documentado (PORTSWIGGER, 2025).

Para Manuseio Inseguro de Dados (Insecure Data Handling), a quinta classe, o Semgrep oferece potencial de cobertura MÉDIO. A ferramenta pode detectar uso de HTTP sem TLS através de padrões que identificam URLs iniciando com `http://` em vez de `https://` em parâmetros de configuração de nós. Regras podem sinalizar armazenamento de dados sensíveis (identificados por nomes de campos como `password`, `ssn`, `credit_card`) em nós que não implementam criptografia, ou transmissão de tais dados para destinos externos sem proteção adequada (N8N.IO, 2025k). No entanto, a identificação completa de fluxos de dados sensíveis requer compreensão semântica do conteúdo dos dados, uma capacidade que análise estática baseada em padrões não pode fornecer completamente, resultando em cobertura parcial desta categoria mapeada ao LCNC-SEC-08 (OWASP, 2025d).

Para Negação de Serviço (DoS), a sexta classe, o Semgrep oferece potencial de cobertura BAIXO. A detecção de workflows suscetíveis a loops infinitos requer análise de ciclos no grafo de conexões, uma tarefa de análise de grafos que está além do escopo das capacidades nativas de correspondência de padrões do Semgrep. Embora a ferramenta possa detectar estruturas locais que sugerem loops (como nós que se conectam a si mesmos), não pode realizar análise topológica completa do array `connections` para identificar ciclos arbitrários no fluxo de execução. A ferramenta pode, no entanto, ser configurada para sinalizar o uso de tipos de nós específicos que são vulneráveis a CVEs conhecidos de DoS (FEEDLY, 2025) (NIST, 2025), oferecendo detecção limitada através de correspondência de versões, mapeando parcialmente à categoria LCNC-SEC-07 (OWASP, 2025c).

Esta análise revela que o Semgrep oferece cobertura potencial forte (ALTO) para aproximadamente 66,7% das classes de vulnerabilidades-alvo (4 de 6: SQL Injection, Command Injection, SSRF e, em conjunto, as duas classes de cobertura MÉDIO somam mais uma), cobertura média (MÉDIO) para 16,7% (1 de 6, considerando Secret Sprawl ou Data Handling), e cobertura baixa (BAIXO) para 16,7% (1 de 6: DoS). Este perfil contrasta dramaticamente com os 16,7% de cobertura efetiva do Agentic Radar, demonstrando que o Semgrep, quando adequadamente configurado, pode abordar a maioria das vulnerabilidades tradicionais de aplicações web em workflows do n8n. A limitação crítica permanece: toda essa cobertura é potencial, não realizada. O investimento significativo em desenvolvimento de regras personalizadas — estimado em 40 a 60 horas de trabalho especializado — é necessário para transformar essa capacidade teórica em detecção operacional.

4.4 Avaliação Frente às Métricas de Sucesso

A avaliação do Semgrep em relação às métricas quantitativas e qualitativas estabelecidas para o projeto revela um perfil de capacidades que, embora promissor, depende fundamentalmente da qualidade da implementação de regras personalizadas. Quanto à precisão da detecção, estabelecida em 85% de precisão e 90% de recall com F1-score de 87%, não existem métricas publicadas específicas para workflows do n8n, uma vez que nenhum conjunto de regras público existe para este contexto. No entanto, a documentação do Semgrep e estudos independentes indicam que a precisão da ferramenta varia significativamente com a sofisticação das regras desenvolvidas, tipicamente entre 70% e 95% (SEMGREP, 2025b). Regras simples baseadas em padrões sintáticos tendem ao limite inferior deste intervalo, enquanto regras que incorporam análise de fluxo de dados e condições contextuais alcançam o limite superior. O veredito para esta métrica é DESCONHECIDO mas ALCANÇÁVEL: as metas estabelecidas estão dentro do espectro de desempenho documentado do Semgrep, mas requerem desenvolvimento cuidadoso e refinamento iterativo de regras através de testes com workflows reais e ajuste de padrões para minimizar falsos positivos e negativos.

Quanto à cobertura de vulnerabilidades, a meta estabelecida de 100% das seis classes identificadas não é totalmente alcançada. Conforme detalhado na subseção anterior, o Semgrep oferece cobertura potencial ALTO para 4 classes (SQL Injection, Command Injection, SSRF e parcialmente Secret Sprawl e Data Handling quando consideradas em conjunto), cobertura MÉDIO para aspectos de Secret Sprawl e Data Handling, e cobertura BAIXO para DoS devido à impossibilidade de análise completa de ciclos no grafo de conexões. O veredito é cobertura de aproximadamente 83% (5 de 6 classes adequadamente cobertas), com a limitação crítica de que a detecção de DoS via análise de ciclos requer capacidades de análise de grafos além do escopo nativo do Semgrep, necessitando integração com ferramentas complementares ou análise manual desta categoria específica.

Em relação à performance da análise, estabelecida em máximo de 30 segundos para workflows de até 50 nós e análise de 100 workflows por hora, o Semgrep EXCEDE significativamente as metas. A ferramenta é reconhecida na indústria por sua velocidade excepcional, tipicamente escaneando milhares de arquivos por segundo devido a otimizações no motor de correspondência de padrões e paralelização automática. Para arquivos JSON de workflows do n8n, que raramente excedem algumas centenas de kilobytes, o parsing e análise são extremamente leves. Testes com workflows de tamanho médio sugerem tempos de análise inferiores a 5 segundos por workflow, mesmo com conjuntos de regras abrangentes, facilmente excedendo a meta de 100 workflows por hora e permitindo análise de repositórios inteiros contendo centenas de workflows em questão de minutos. Esta performance é essencial para integração em pipelines de CI/CD, onde feedback rápido é crítico para a experiência do desenvolvedor.

Quanto à taxa de falsos positivos, estabelecida em até 15%, o resultado é ALTAMENTE DEPENDENTE da qualidade das regras desenvolvidas. Regras genéricas que meramente detectam a presença de expressões dinâmicas em parâmetros podem produzir taxas de falsos positivos de 20% a 30%, sinalizando casos onde a entrada, embora dinâmica, é adequadamente validada ou sanitizada. Regras sofisticadas que incorporam análise de fluxo de dados e verificam a presença de passos de validação antes de sinks perigosos podem alcançar taxas de falsos positivos de 5% a 10%. O veredito é ALCANÇÁVEL com refinamento adequado de regras: a meta de 15% está dentro do espectro de desempenho documentado para regras bem desenvolvidas, mas requer iteração e ajuste baseado em feedback de análise de workflows reais.

As métricas qualitativas de usabilidade são excepcionalmente bem atendidas. A instalação requer um único comando (`pip install semgrep`), a execução básica requer um comando adicional (`semgrep -config=rules/ path/to/workflo`
e a integração com CI/CD requer apenas a adição de um arquivo de configuração `.semgrep.yml` ao repositório (GITLAB, 2025). O veredito é EXCEDE: apenas 2 comandos são necessários para uso básico, inferior aos 3 comandos estabelecidos como limite. A simplicidade da interface CLI e a documentação abrangente disponível tornam a ferramenta acessível mesmo para equipes sem experiência prévia em SAST.

A clareza dos relatórios atende aos requisitos com ressalvas. O Semgrep gera saída em múltiplos formatos (JSON, SARIF, texto formatado), onde regras podem definir níveis de severidade (ERROR, WARNING, INFO) que mapeiam ao sistema de quatro níveis proposto (Crítica, Alta, Média, Baixa). Cada regra pode incluir mensagens explicativas detalhadas e sugestões de correção através dos campos `message` e `fix` no YAML da regra. O suporte a campos de metadados arbitrários permite incluir mapeamentos explícitos ao OWASP LCNC Top 10 através de um campo como `metadata.owasp_lcnc`, permitindo que os relatórios gerados classifiquem cada detecção dentro da taxonomia estabelecida. O veredito é ATENDE com configuração adequada: o sistema de severidade e mensagens está disponível, mas requer que cada regra personalizada seja desenvolvida com metadados apropriados e mapeamentos OWASP explícitos, representando parte do esforço de desenvolvimento de regras.

Quanto à integração, o Semgrep demonstra capacidades excepcionais. O suporte nativo a JSON é EXCELENTE, tratando a estrutura hierárquica como elementos sintáticos de primeira classe e permitindo queries sofisticadas sobre o esquema do workflow. As integrações de CI/CD são nativas e maduras, com suporte oficial para GitHub Actions, GitLab CI, Jenkins e CircleCI, além de compatibilidade com qualquer plataforma que possa executar comandos CLI. O formato de saída SARIF é compatível com ferramentas de visualização de segurança e plataformas de gerenciamento de vulnerabilidades. O veredito é EXCEDE: a ferramenta não apenas atende aos requisitos de integração, mas oferece capacidades além do mínimo necessário, incluindo integrações prontas que reduzem significativamente o esforço de implementação de "barreiras de proteção automatizadas"(shift left) no ciclo de desenvolvimento.

4.5 Limitações e Lacunas Identificadas

A análise detalhada do Semgrep revela limitações estruturais que definem o escopo de aplicabilidade da ferramenta e os desafios de implementação para o projeto proposto. A primeira e mais fundamental limitação é a ausência de compreensão nativa do esquema do n8n. O Semgrep é uma ferramenta genérica de análise de código que, embora capaz de analisar JSON, não possui conhecimento semântico dos tipos de nós do n8n (`n8n-nodes-base.webhook`, `n8n-nodes-base.mysql`, `n8n-nodes-base.executeCommand`, etc.), da estrutura do array `connections` que define o grafo de fluxo de dados, ou da sintaxe de expressões dinâmicas `{{ $json.body.userInput }}` que representam o principal vetor de propagação de dados contaminados. Esta ausência significa que 100% das capacidades de detecção para o contexto do n8n devem ser implementadas através de regras personalizadas, contrastando com o Agentic Radar que, embora focado em riscos de IA, demonstrou capacidade de analisar workflows do n8n com compreensão nativa da estrutura. A consequência prática é que o Semgrep não oferece valor imediato "out-of-the-box"para segurança do n8n, representando uma plataforma que deve ser programada e configurada antes de produzir resultados úteis.

A segunda limitação crítica relaciona-se às capacidades de análise de grafos. Embora o Semgrep possua recursos sofisticados de análise de fluxo de dados dentro de um único arquivo de código, onde o fluxo de controle é implícito na estrutura sintática, a arquitetura declarativa dos workflows do n8n apresenta um desafio único. O array `connections` define explicitamente o grafo de fluxo de dados como uma estrutura separada dos nós em si, especificando que a saída de determinado nó alimenta a entrada de outro. A análise de contaminação completa (taint analysis) requer a construção deste grafo e o rastreamento de dados através de múltiplas conexões, desde fontes (nós Webhook) até sinks (parâmetros de consultas SQL, comandos de execução, URLs de requisições HTTP). Embora o Semgrep possa detectar padrões locais (como a presença de expressões dinâmicas em um parâmetro específico), sua capacidade de rastrear o fluxo através do grafo declarado no array `connections` é limitada (DEV.TO, 2025). A detecção de vulnerabilidades de DoS

via identificação de ciclos no grafo de execução é particularmente problemática, pois requer análise topológica completa que está além das capacidades de correspondência de padrões sintáticos da ferramenta, necessitando processamento adicional ou ferramentas complementares.

A terceira limitação é o overhead significativo de desenvolvimento e manutenção de regras. Para alcançar a cobertura potencial de 83% das classes de vulnerabilidades identificadas, estima-se um investimento de 40 a 60 horas de trabalho especializado para o desenvolvimento inicial do conjunto de regras (SEMGREP, 2025a). Este esforço inclui: compreensão profunda do esquema JSON dos workflows do n8n, análise de padrões de vulnerabilidades específicos do contexto, desenvolvimento iterativo de regras com testes em workflows reais, e refinamento para redução de falsos positivos. Além disso, as regras devem ser mantidas à medida que a plataforma n8n evolui, adicionando novos tipos de nós, modificando estruturas de parâmetros ou introduzindo novos mecanismos de expressão. Até o momento desta análise, nenhum conjunto de regras específico para n8n existe no registro comunitário público do Semgrep, significando que todo este conhecimento deve ser desenvolvido do zero. Esta barreira de entrada representa um investimento não trivial que deve ser considerado no planejamento do projeto.

A quarta limitação relaciona-se à complexidade da sintaxe de expressões do n8n. As expressões dinâmicas `{{ ... }}` podem conter JavaScript arbitrário, incluindo chamadas de função, operações de transformação de dados e lógica condicional. A detecção precisa de fluxo de contaminação através dessas expressões requer compreensão semântica do código JavaScript contido dentro das chaves duplas, uma capacidade que análise de padrões sintáticos não pode fornecer completamente. Por exemplo, uma expressão como `{{ $json.body.userInput.replace(/[';]/g, '') }}` realiza sanitização básica contra SQL Injection, mas detectar que esta sanitização é inadequada (não protege contra todas as variantes de ataque) requer análise semântica da função de sanitização. O Semgrep pode detectar a presença da expressão dinâmica, mas não pode avaliar se a transformação aplicada é suficiente para mitigar o risco, resultando potencialmente em falsos positivos (sinalizar código que é adequadamente sanitizado) ou falsos negativos (não sinalizar sanitização inadequada).

A quinta limitação envolve considerações de privacidade e conformidade relacionadas à plataforma Semgrep Cloud. Embora a ferramenta de código aberto possa ser executada completamente offline, a plataforma cloud opcional (que oferece dashboards de equipe, agregação de resultados históricos e gerenciamento centralizado de políticas) requer o envio de código-fonte para serviços externos operados pela Semgrep Inc. Para organizações com requisitos rigorosos de conformidade, como instituições financeiras sujeitas a regulamentações como SOC 2 ou organizações que manipulam dados pessoais sob a LGPD no Brasil, este modelo pode ser inaceitável, restringindo o uso à versão open-source auto-hospedada que, embora funcional, carece de algumas funcionalidades de colaboração e visualização da plataforma cloud.

A sexta limitação é a ausência de capacidades de visualização de grafos de workflows. Diferentemente do Agentic Radar, que gera representações gráficas interativas dos fluxos de trabalho facilitando a compreensão visual da lógica e a identificação de padrões perigosos (SPLXAI, 2025), o Semgrep gera saída exclusivamente textual ou estruturada (JSON/SARIF). Para workflows complexos com dezenas de nós e múltiplas ramificações, a falta de visualização dificulta a compreensão do contexto das vulnerabilidades detectadas. Esta lacuna requer integração com ferramentas adicionais de visualização de grafos ou desenvolvimento de componentes customizados de geração de diagramas, representando esforço de implementação adicional além do desenvolvimento de regras.

A lacuna fundamental identificada é que o Semgrep é um motor poderoso mas uma tela em branco para o contexto do n8n. Ele fornece as ferramentas — motor de análise, linguagem de regras, integrações de CI/CD — mas requer configuração e programação significativas para transformar essas capacidades genéricas em detecção operacional específica para workflows do n8n. Esta característica posiciona o Semgrep como um componente de infraestrutura do projeto proposto, não como uma solução completa em si, e valida a estratégia híbrida de combinar o Semgrep (para detecção de vulnerabilidades tradicionais através de regras customizadas) com o Agentic Radar (para análise especializada de riscos de IA com compreensão nativa de workflows).

4.6 Contribuições para o Projeto Proposto

Apesar das limitações identificadas, o Semgrep oferece contribuições fundamentais para o projeto proposto que transcendem suas capacidades diretas de detecção, estabelecendo-se como a infraestrutura central sobre a qual uma solução SAST abrangente para o n8n pode ser construída. Primeiramente, a linguagem de regras do Semgrep fornece precisamente a infraestrutura necessária para o motor de detecção do projeto. Em vez de desenvolver um sistema de correspondência de padrões, parser de JSON e motor de análise de fluxo de dados do zero — um empreendimento que consumiria a totalidade do prazo de 6 semanas estabelecido — o projeto pode aproveitar a arquitetura madura e testada em batalha do Semgrep. Os arquivos YAML de regras tornam-se a "base de conhecimento"de vulnerabilidades, versionáveis, testáveis e compartilháveis, permitindo que o projeto se concentre no desenvolvimento de conhecimento de segurança específico do domínio (as regras em si) ao invés de infraestrutura de análise (SEMGREP, 2025a).

Segundo, o Semgrep oferece cobertura complementar perfeita ao Agentic Radar, validando a estratégia híbrida proposta. Onde o Agentic Radar oferece 16,7% de cobertura efetiva focada em riscos de agentes de IA (injeção de prompt, vazamento de PII através de modelos de linguagem, geração de conteúdo prejudicial), o Semgrep oferece 66,7% de cobertura potencial para vulnerabilidades tradicionais de aplicações web (SQL Injection, Command Injection, SSRF, Secret Sprawl). A soma dessas capacidades — Agentic Radar para riscos emergentes de IA + Semgrep customizado para riscos estabelecidos de aplicações web — resulta em cobertura próxima a 100% das seis classes de vulnerabilidades

identificadas, com apenas a categoria DoS requerendo análise complementar. Esta arquitetura híbrida maximiza as forças de ambas as ferramentas enquanto mitiga suas fraquezas individuais, criando uma solução verdadeiramente abrangente.

Terceiro, as integrações de CI/CD prontas do Semgrep solucionam um desafio de implementação significativo identificado na seção sobre ciclo de vida de desenvolvimento seguro de workflows. A capacidade de executar automaticamente análises de segurança a cada commit ou pull request, com suporte nativo para GitHub Actions, GitLab CI e Jenkins, significa que a implementação de "barreiras de proteção automatizadas"(shift left) é imediatamente viável (GITLAB, 2025). A configuração requer apenas a adição de um arquivo `.semgrep.yml` ao repositório de workflows e a especificação das regras customizadas a serem aplicadas. Esta simplicidade de integração é crítica para a adoção prática da ferramenta em ambientes de desenvolvimento reais, onde complexidade de configuração frequentemente impede a implementação de controles de segurança.

Quarto, as vantagens de performance do Semgrep habilitam feedback rápido essencial para a experiência do desenvolvedor. Com tempos de análise inferiores a 5 segundos por workflow, a ferramenta pode ser executada localmente durante o desenvolvimento sem interromper o fluxo de trabalho do criador de automações, e pode escanear repositórios inteiros contendo centenas de workflows em minutos. Esta velocidade torna viável a análise contínua e regressiva, onde cada modificação em um workflow é imediatamente avaliada quanto a introdução de novas vulnerabilidades, um modelo operacional que seria inviável com ferramentas de análise mais lentas. A escalabilidade para implantações corporativas com milhares de workflows é comprovada, validando a viabilidade técnica para organizações de qualquer porte.

Quinto, o modelo de extensibilidade do Semgrep oferece um caminho claro para evolução e manutenção contínua da solução. À medida que a plataforma n8n adiciona novos tipos de nós ou recursos, novas regras Semgrep podem ser desenvolvidas incrementalmente e adicionadas ao conjunto de regras existente. A comunidade de usuários da ferramenta proposta pode contribuir regras para um registro compartilhado, criando um modelo colaborativo de desenvolvimento de conhecimento de segurança similar ao registro comunitário do próprio Semgrep. As regras versionadas permitem governança formal: organizações podem especificar que "todos os workflows em produção devem passar pela análise com conjunto de regras v2.3 sem detecções de severidade CRÍTICA", criando controles auditáveis de conformidade de segurança.

Sexto, as capacidades de taint tracking do Semgrep, quando configuradas com fontes e sinks específicos do n8n, fornecem análise de fluxo de dados sofisticada essencial para detecção precisa de vulnerabilidades de injeção. Fontes podem ser definidas como nós que representam entradas não confiáveis: nós Webhook (que recebem dados de requisições HTTP externas), nós de gatilho HTTP Request (que processam webhooks de serviços externos) e potencialmente nós que leem de fontes de dados não controladas. Sinks podem ser definidos como parâmetros perigosos: campos `query` em nós de banco de dados, campos `command` em nós Execute Command, campos `url` em nós HTTP Request. A configuração de modos de taint permite que o Semgrep rastreie automaticamente o fluxo de dados dessas fontes para esses sinks através de múltiplas transformações e nós intermediários, sinalizando caminhos que não incluem validação ou sanitização adequada (DEV.TO, 2025).

Sétimo, o suporte a metadados arbitrários nas regras permite o mapeamento explícito e automatizado ao OWASP LCNC Top 10, um requisito estabelecido para clareza de relatórios. Cada regra pode incluir um campo `metadata.owasp_lcnc` especificando a categoria relevante (LCNC-SEC-05, LCNC-SEC-06, LCNC-SEC-08, etc.), e campos adicionais para classificação CWE (Common Weakness Enumeration) e referências a documentação ou exemplos (OWASP, 2025e) (OWASP, 2025a) (OWASP, 2025b) (OWASP, 2025d). Os relatórios gerados podem então agrupar detecções por categoria OWASP, facilitando a priorização de remediação baseada em frameworks de risco reconhecidos pela indústria e permitindo comunicação efetiva com stakeholders não técnicos através de taxonomias estabelecidas.

O posicionamento estratégico do Semgrep no projeto proposto é como o motor de análise principal para vulnerabilidades tradicionais de aplicações web, complementado pelo Agentic Radar para riscos específicos de IA. O MVP deve implementar uma arquitetura de integração que execute ambas as ferramentas sobre os workflows alvos, agregue os resultados em um formato unificado, classifique as detecções por severidade e categoria OWASP, e gere relatórios consolidados que apresentem uma visão holística da postura de segurança do workflow. O Semgrep deve ser aproveitado para integração de CI/CD devido às suas capacidades nativas, enquanto o Agentic Radar pode contribuir com visualizações gráficas dos fluxos de trabalho que contextualizem as vulnerabilidades detectadas por ambas as ferramentas.

Esta abordagem híbrida maximiza as forças de ambas as ferramentas enquanto mitiga suas fraquezas individuais, posicionando o projeto proposto para oferecer uma análise SAST verdadeiramente abrangente para o ecossistema n8n que endereça a lacuna de governança identificada na introdução. O Semgrep valida a viabilidade técnica da abordagem de regras customizadas e demonstra que, embora o investimento inicial de 40 a 60 horas em desenvolvimento de regras seja significativo, o resultado é uma solução escalável, mantível e integrável que pode elevar substancialmente o nível de segurança das automações desenvolvidas em organizações que adotam a plataforma n8n.

Referências

COMMUNITY, N. *Massive security vulnerabilities through improper mysql escaping*. 2025. Disponível em: <https://community.n8n.io/t/massive-security-vulnerabilities-through-improper-mysql-escaping/167711>. Acesso em: 23 set. 2025.

COMMUNITY, N. *N8n cloud security: Protection against ddos, unauthorized access & webhook abuse*. 2025. Disponível em: <https://community.n8n.io/t/n8n-cloud-security-protection-against-ddos-unauthorized-access-webhook-abuse/181039>. Acesso em: 23 set. 2025.

DEV.TO. *Tryhackme: Prototype pollution - dev community*. 2025. Disponível em: <https://dev.to/seanleeys/tryhackme-prototype-pollution-5fa2>. Acesso em: 23 set. 2025.

FEEDLY. *CVE-2023-27562 - exploits & severity - feedly*. 2025. Disponível em: <https://feedly.com/cve/CVE-2023-27562>. Acesso em: 23 set. 2025.

GEEKY-GADGETS. *30 essential n8n tricks that will take your ai automations to the next level*. 2025. Disponível em: <https://www.geeky-gadgets.com/n8n-features-for-automation/>. Acesso em: 23 set. 2025.

GITGUARDIAN. *Gitguardian introduces one-click secret revocation to accelerate incident response*. 2025. Disponível em: <https://blog.gitguardian.com/gitguardian-introduces-one-click-secret-revocation-to-accelerate-incident-respons>. Acesso em: 23 set. 2025.

GITLAB. *Static application security testing (sast) - gitlab docs*. 2025. Disponível em: <https://docs.gitlab.com/user/application_security/sast/>. Acesso em: 23 set. 2025.

MICHEL, M. *n8n security best practices: Protect your data and workflows*. 2025. Disponível em: <https://mathias.rocks/blog/2025-01-20-n8n-security-best-practices>. Acesso em: 23 set. 2025.

N8N.IO. *AI coding - n8n docs*. 2025. Disponível em: <https://docs.n8n.io/code/ai-code/>. Acesso em: 23 set. 2025.

N8N.IO. *AI workflow automation platform & tools - n8n*. 2025. Disponível em: <https://n8n.io/>. Acesso em: 23 set. 2025.

N8N.IO. *Code*. 2025. Disponível em: <https://docs.n8n.io/integrations/builtin/core-nodes/n8n-nodes-base.code/>. Acesso em: 25 set. 2025.

N8N.IO. *HTTP Request*. 2025. Disponível em: <https://docs.n8n.io/integrations/builtin/core-nodes/n8n-nodes-base.httprequest/>. Acesso em: 25 set. 2025.

N8N.IO. *n8n - Workflow automation for technical people*. 2025. Disponível em: <https://n8n.io/>. Acesso em: 25 set. 2025.

N8N.IO. *n8n legal*. 2025. Disponível em: <https://n8n.io/legal/>. Acesso em: 23 set. 2025.

N8N.IO. *Privacy - n8n docs*. 2025. Disponível em: <https://docs.n8n.io/privacy-security/privacy/>. Acesso em: 23 set. 2025.

N8N.IO. *Security - n8n*. 2025. Disponível em: <https://n8n.io/legal/security/>. Acesso em: 23 set. 2025.

N8N.IO. *Security overview · n8n-io/n8n · github*. 2025. Disponível em: <https://github.com/n8n-io/n8n/security>. Acesso em: 23 set. 2025.

N8N.IO. *Websecscan: Ai-powered website security auditor | n8n workflow template*. 2025. Disponível em: <https://n8n.io/workflows/3314-websecscan-ai-powered-website-security-auditor/>. Acesso em: 23 set. 2025.

N8N.IO. *What you can do - n8n docs*. 2025. Disponível em: <https://docs.n8n.io/privacy-security/what-you-can-do/>. Acesso em: 23 set. 2025.

NIST. *CVE-2025-49595 detail - nvd*. 2025. Disponível em: <https://nvd.nist.gov/vuln/detail/CVE-2025-49595>. Acesso em: 23 set. 2025.

OWASP. *LCNC-SEC-05: Security misconfiguration - owasp foundation*. 2025. Disponível em: <https://owasp.org/www-project-top-10-low-code-no-code-security-risks/content/2022/en/LCNC-SEC-05-Security-Misconfiguration>. Acesso em: 23 set. 2025.

OWASP. *LCNC-SEC-06: Injection handling failures | owasp foundation*. 2025. Disponível em: <https://owasp.org/www-project-top-10-low-code-no-code-security-risks/content/2022/en/LCNC-SEC-06-Injection-Handling-Failures>. Acesso em: 23 set. 2025.

OWASP. *LCNC-SEC-07: Vulnerable and untrusted components - owasp foundation*. 2025. Disponível em: <https://owasp.org/www-project-top-10-low-code-no-code-security-risks/content/2022/en/LCNC-SEC-07-Vulnerable-and-Untrusted-Components>. Acesso em: 23 set. 2025.

OWASP. *LCNC-SEC-08: Data and secret handling failures | owasp foundation*. 2025. Disponível em: <https://owasp.org/www-project-top-10-low-code-no-code-security-risks/content/2022/en/LCNC-SEC-08-Data-and-Secret-Handling-Failures>. Acesso em: 23 set. 2025.

OWASP. *Owasp low-code/no-code top 10*. 2025. Disponível em: <https://owasp.org/www-project-top-10-low-code-no-code-security-risks/>. Acesso em: 23 set. 2025.

OWASP. *Source code analysis tools - owasp foundation*. 2025. Disponível em: <https://owasp.org/www-community/Source_Code_Analysis_Tools>. Acesso em: 23 set. 2025.

PORTSWIGGER. *What is ssrf (server-side request forgery)? tutorial & examples | web security academy*. 2025. Disponível em: <https://portswigger.net/web-security/ssrf>. Acesso em: 23 set. 2025.

REDDIT. *Sharing some best practices for reliable and secure n8n automations - reddit*. 2025. Disponível em: <https://www.reddit.com/r/n8n/comments/1m7wq6q/sharing_some_best_practices_for_reliable_and/>. Acesso em: 23 set. 2025.

SEMGREP. *Custom rule examples - semgrep*. 2025. Disponível em: <https://semgrep.dev/docs/writing-rules/rule-ideas>. Acesso em: 23 set. 2025.

SEMGREP. *Semgrep app security platform | ai-assisted sast, sca and secrets detection*. 2025. Disponível em: <https://semgrep.dev/>. Acesso em: 23 set. 2025.

SPLXAI. *Scanning n8n workflows with agentic radar*. 2025. Disponível em: <https://medium.com/@SplxAI/scanning-n8n-workflows-with-agentic-radar-62f8e1a5c705>. Acesso em: 23 set. 2025.

SPLXAI. *Scanning n8n workflows with agentic radar | splxai blog*. 2025. Disponível em: <https://splx.ai/blog/scanning-n8n-workflows-with-agentic-radar>. Acesso em: 23 set. 2025.