SAST para detecção de vulnerabilidades em workflows do n8n

João Pedro Schmidt Cordeiro Gustavo Zambonin

24 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 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 baseia-se em uma estratégia híbrida que combina o Agentic Radar (focado em riscos de agentes de IA) e o Semgrep (configurado com regras customizadas para vulnerabilidades tradicionais). Além disso, propõe-se o uso de inteligência artificial para acelerar a criação de regras de detecção. A análise revelou coberturas complementares de 16,7% e 66,7% respectivamente, demonstrando a viabilidade da análise estática de estruturas JSON declarativas. A solução contribui para elevar o nível de segurança das automações desenvolvidas em organizações brasileiras, auxiliando na conformidade com a LGPD.

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 podem conter seis classes críticas de vulnerabilidades:

- **SQL Injection (SQLi):** Entradas controladas pelo usuário inseridas em consultas SQL sem sanitização adequada.
- **Command/Code Injection:** Dados não confiáveis usados em comandos executados no servidor.
- **Secret Sprawl & Chaining:** Dispersão de segredos (chaves de API, tokens) em configurações ou logs.
- **SSRF:** Entradas controladas pelo usuário determinam URLs em requisições HTTP sem validação.
- **Insecure Data Handling:** Falhas na proteção de dados sensíveis durante armazenamento ou transmissão.
- **Denial of Service (DoS):** Workflows suscetíveis a loops infinitos ou operações intensivas.

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 contribuições técnicas, metodológicas e científicas para segurança em plataformas LCNC:

1. **Análise Comparativa:** Avaliação sistemática de Agentic Radar e Semgrep, documentando coberturas complementares e aplicabilidade ao contexto n8n.
2. **Validação de Viabilidade:** Demonstração empírica da eficácia da análise estática de estruturas JSON declarativas.
3. **Abordagem Híbrida:** Arquitetura que combina ferramentas especializadas e genéricas configuráveis, maximizando cobertura.
4. **Metodologia com IA:** Uso de inteligência artificial para acelerar a criação de regras de detecção.
5. **Documentação de Lacunas:** Identificação de limitações e oportunidades para pesquisas futuras.

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 PII, geração de conteúdo nocivo e disseminação de desinformação. Esta especialização não cobre vulnerabilidades tradicionais de aplicações web, limitando sua cobertura a aproximadamente 16,7% das classes identificadas.

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 revela que o Semgrep oferece potencial de cobertura para aproximadamente 66,7% das classes identificadas, mas esta capacidade é inteiramente potencial: depende do desenvolvimento de regras personalizadas. Até o momento, nenhum conjunto de regras específico para n8n existe no registro comunitário público.

Limitação Crítica Identificada: Modo JSON vs. Modo Genérico

Durante a fase experimental deste trabalho, identificou-se uma limitação fundamental do modo JSON do Semgrep quando aplicado a estruturas altamente aninhadas como workflows do n8n. O modo JSON, que realiza parsing da estrutura em árvores sintáticas abstratas (AST), apresenta dificuldades em aplicar padrões de detecção em arrays profundamente aninhados (4+ níveis de profundidade), como o array `nodes` dentro de workflows n8n.

A solução identificada foi a utilização do **modo genérico** (generic mode) do Semgrep, que trata o arquivo JSON como texto plano e aplica correspondência baseada em expressões regulares. Esta abordagem, embora menos sofisticada semanticamente, demonstrou-se significativamente mais efetiva para detecção de padrões inseguros em workflows n8n, conforme será detalhado na Seção 7.

Esta descoberta tem implicações importantes para a análise de segurança de plataformas LCNC: ferramentas SAST tradicionais requerem adaptação cuidadosa ao paradigma declarativo de configuração-como-código, e a escolha do modo de análise apropriado é tão crítica quanto a seleção da ferramenta em si.

A principal limitação é a ausência de conhecimento nativo do esquema n8n, exigindo investimento estimado entre 40 e 60 horas de trabalho especializado para desenvolver regras customizadas abrangentes.

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 anterior, **não existe atualmente uma ferramenta dedicada especificamente à detecção de vulnerabilidades em workflows do n8n**. As soluções existentes oferecem cobertura parcial e complementar, mas nenhuma aborda o problema de forma abrangente. Esta lacuna representa um risco substancial para organizações que dependem da plataforma para automações críticas.

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 promissora é a utilização de IA para geração assistida de regras Semgrep. O desenvolvimento manual representa uma barreira significativa (40-60 horas estimadas). Modelos de linguagem podem 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.

A implementação desta abordagem utiliza prompts especializados e few-shot learning, fornecendo ao Claude 3.5 Sonnet exemplos de regras Semgrep existentes para outras plataformas e solicitando adaptação para o esquema JSON do n8n. A validação das regras geradas por IA através de testes automatizados em workflows é essencial para garantir confiabilidade antes da implantação em ambientes de produção.

2.4.3 Iniciativa Comunitária: n8n-CyberSecurity-Workflows

A comunidade de segurança cibernética tem reconhecido a importância de automação de segurança utilizando n8n. O repositório `n8n-CyberSecurity-Workflows` [1], mantido pela comunidade CyberSecurityUP, documenta mais de 100 ideias de workflows para automação de segurança em Red Team, Blue Team, AppSec e DevSecOps.

Relevante para este trabalho, o repositório menciona explicitamente a possibilidade de utilizar Semgrep para análise de segurança de workflows n8n em formato JSON, reconhecendo a lacuna de ferramentas especializadas. Esta iniciativa comunitária reforça a necessidade identificada neste trabalho e sugere demanda prática por soluções SAST dedicadas ao ecossistema n8n.

2.4.4 Oportunidade: Arquitetura Híbrida para Cobertura Completa

A análise comparativa sugere que uma arquitetura híbrida poderia alcançar cobertura próxima a 100% das classes de vulnerabilidades. Esta estratégia maximiza as forças complementares: Agentic Radar oferece análise especializada em agentes de IA, enquanto Semgrep oferece motor flexível e poderoso para 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

Esta seção descreve a abordagem metodológica adotada para o desenvolvimento e validação da ferramenta SAST proposta, detalhando o processo de análise das ferramentas existentes, a metodologia inovadora de geração de regras assistida por inteligência artificial e os procedimentos de validação empregados.

3.1 Abordagem Metodológica Geral

A pesquisa adota metodologia exploratória e experimental, fundamentada em três pilares:

1. **Pesquisa exploratória:** Análise comparativa de Agentic Radar e Semgrep, documentando capacidades, limitações e aplicabilidade.
2. **Abordagem híbrida:** Arquitetura combinando ferramentas complementares para maximizar cobertura.
3. **IA para aceleração:** Uso de modelos de linguagem para geração assistida de regras, reduzindo overhead de desenvolvimento.

3.2 Análise das Ferramentas Existentes

A análise comparativa das ferramentas do estado da arte seguiu um protocolo estruturado para garantir avaliação objetiva e sistemática das capacidades e limitações de cada solução.

3.2.1 Critérios de Avaliação

As ferramentas foram avaliadas segundo quatro critérios principais:

1. **Cobertura de vulnerabilidades:** Capacidade de detectar as seis classes identificadas (classificada como ALTO, MÉDIO, BAIXO ou NENHUMA).
2. **Performance:** Tempo de execução, throughput e escalabilidade.
3. **Facilidade de uso:** Complexidade de instalação, documentação e interface.
4. **Qualidade dos relatórios:** Clareza, severidade, explicações e sugestões de remediação.

3.2.2 Processo de Análise

O processo de análise de cada ferramenta seguiu um protocolo sistemático estruturado em quatro etapas principais. Inicialmente, realizou-se a **instalação e configuração** das ferramentas Agentic Radar (via `pip install agentic-radar`) e Semgrep (via `pip install semgrep`) em ambiente Python 3.10+. Durante esta etapa, foram documentadas todas as dependências, requisitos de configuração (como variáveis de ambiente) e possíveis incompatibilidades encontradas, garantindo a reprodutibilidade do ambiente de testes.

Na segunda etapa, conduziram-se **testes com workflows de exemplo**, executando cada ferramenta sobre um conjunto de workflows representativos que cobrem diferentes tipos de nós relevantes para segurança, incluindo Webhook, MySQL, HTTP Request, Execute Command e Code. Esta diversidade de nós permitiu avaliar a capacidade das ferramentas em diferentes contextos de vulnerabilidades.

A terceira etapa consistiu na **análise de capacidades de detecção**, avaliando as seis classes de vulnerabilidades em três workflows de teste e verificando se cada ferramenta detecta os padrões inseguros implementados. Para cada detecção ou ausência de detecção, documentou-se sistematicamente se tratava-se de true positive, false positive, true negative ou false negative, permitindo análise quantitativa posterior.

A etapa final envolveu a **documentação de cobertura e limitações**, realizando o mapeamento sistemático da cobertura de cada ferramenta para as seis classes de vulnerabilidades,

identificando lacunas fundamentais (vulnerabilidades não cobertas) e documentando limitações técnicas, operacionais e de privacidade. Os resultados desta análise comparativa foram consolidados nas seções anteriores deste documento, fornecendo a base empírica para a proposta da abordagem híbrida.

3.3 Geração de Regras Semgrep Assistida por IA

A metodologia de geração assistida de regras representa a contribuição metodológica mais inovadora deste trabalho, aplicando técnicas de inteligência artificial generativa para acelerar e sistematizar o desenvolvimento de conhecimento de segurança específico do domínio do n8n.

3.3.1 Preparação de Workflows de Teste

Desenvolveram-se três workflows de teste implementando deliberadamente as seis classes de vulnerabilidades em cenários realistas. Exemplos incluem SQL Injection (entrada de webhook passada diretamente para consulta MySQL), Command Injection (entrada usada em Execute Command) e SSRF (URL dinâmica sem validação).

Os workflows foram documentados detalhadamente, especificando localização exata dos padrões inseguros, categorias OWASP e severidades. Esta documentação serve como ground truth para validação.

3.3.2 Metodologia de Geração de Regras com IA

A geração assistida segue processo iterativo estruturado. Primeiro, analisou-se o esquema JSON do n8n (arrays `nodes` e `connections`, sintaxe de expressões dinâmicas, tipos de nós críticos).

Com este conhecimento, utilizou-se Claude 3.5 Sonnet através de prompts estruturados em três componentes: *Contexto* (esquema n8n), *Objetivo* (vulnerabilidade a detectar) e *Formato* (estrutura YAML com campos obrigatórios).

Exemplo de prompt estruturado:

> *"Você é um especialista em segurança de aplicações e na ferramenta Semgrep. Crie uma regra Semgrep em formato YAML que detecta vulnerabilidades de SQL Injection em workflows do n8n.*
>
> *Contexto: Workflows do n8n são arquivos JSON onde o array 'nodes' contém objetos representando etapas de processamento. Nós de banco de dados MySQL têm type='n8n-nodes-base.mysql' e o parâmetro 'parameters.query' contém a consulta SQL. Expressões dinâmicas usam sintaxe {{ $json.campo }}.*
>
> *Objetivo: Detectar quando o parâmetro 'query' de um nó MySQL contém expressões {{ ... }} que referenciam dados de entrada não confiáveis, indicando possível concatenação de strings em SQL sem sanitização adequada.*
>
> *Formato: A regra deve incluir id, message explicativa, severity='ERROR', metadata com owasp_lcnc='LCNC-SEC-06' e CWE, e padrões que identifiquem o JSON específico do n8n."*

O processo seguiu ciclo iterativo: (1) Geração inicial, (2) Teste em workflows, (3) Avaliação de detecções, (4) Refinamento com feedback, (5) Convergência. Tipicamente, foram necessárias 2-4 iterações por regra, reduzindo drasticamente o tempo de desenvolvimento.

3.3.3 Validação das Regras Geradas

As regras geradas foram validadas rigorosamente: (1) Execução em workflows de teste com documentação sistemática, (2) Ajuste manual para redução de falsos positivos (padrões negativos e contextualizações), (3) Documentação completa incluindo descrição técnica, exemplos, mapeamento OWASP/CWE e sugestões de remediação.

3.4 Validação e Métricas

O processo de validação da abordagem proposta combina avaliação quantitativa (métricas de detecção) e qualitativa (cobertura de vulnerabilidades e utilidade prática).

3.4.1 Testes de Detecção

A validação quantitativa das capacidades de detecção seguiu um protocolo experimental rigorosamente estruturado para garantir resultados objetivos e reproduzíveis. Ambas as ferramentas — Agentic Radar e Semgrep com regras customizadas — foram executadas sobre os três workflows de teste, com documentação sistemática de todos os resultados obtidos. Esta documentação detalhada permite rastreabilidade completa e validação independente dos achados.

Cada detecção ou ausência de detecção foi classificada utilizando a taxonomia padrão de sistemas de detecção: *True Positive (TP)* para vulnerabilidades reais corretamente identificadas, *False Positive (FP)* para detecções em código seguro (falsos alarmes), *True Negative (TN)* para ausência correta de detecção em código seguro, e *False Negative (FN)* para vulnerabilidades reais não detectadas.

A partir desta classificação, calcularam-se as métricas padrão de avaliação de sistemas de detecção:

- Precisão: $P = \frac{TP}{TP+FP}$ (proporção de detecções corretas)
- Recall: $R = \frac{TP}{TP+FN}$ (proporção de vulnerabilidades encontradas)
- F1-Score: $F_1 = 2 \cdot \frac{P \cdot R}{P+R}$ (média harmônica)
- Taxa de Falsos Positivos: $\frac{FP}{FP+TN}$

Estas métricas foram calculadas tanto por ferramenta individual quanto para a abordagem híbrida combinada, permitindo comparação objetiva das capacidades de detecção e demonstrando quantitativamente os benefícios da estratégia híbrida proposta.

3.4.2 Avaliação de Cobertura

A avaliação de cobertura verificou sistematicamente a capacidade da abordagem proposta de endereçar o espectro completo de vulnerabilidades identificadas na revisão da literatura. Para cada uma das seis classes de vulnerabilidades — SQL Injection, Command Injection, Secret Sprawl, SSRF, Insecure Data Handling e DoS — verificou-se se ao menos uma das ferramentas da abordagem híbrida oferece capacidade de detecção efetiva.

A análise comparativa quantitativa revelou diferenças significativas na cobertura oferecida por cada ferramenta individualmente. O Agentic Radar isolado oferece cobertura de 16,7% (1 de 6 classes), focando especificamente em riscos emergentes de agentes de IA. O Semgrep customizado isolado apresenta cobertura potencial de 66,7% (4 de 6 classes com cobertura ALTO),

endereçando vulnerabilidades tradicionais de aplicações web. A abordagem híbrida, combinando ambas as ferramentas, alcança cobertura próxima a 100% através da complementaridade estratégica das capacidades de cada solução.

Adicionalmente, documentaram-se as lacunas residuais, identificando categorias de vulnerabilidades que permanecem parcialmente cobertas ou requerem capacidades de análise além do escopo das ferramentas utilizadas. Um exemplo notável é a detecção de Negação de Serviço (DoS) através de análise completa de ciclos no grafo de execução, que requereria capacidades de análise de fluxo de controle mais sofisticadas do que as oferecidas pelas ferramentas atuais.

Os resultados desta validação são apresentados detalhadamente na Seção 7 (Avaliação Experimental), fornecendo evidência empírica robusta da viabilidade e efetividade da abordagem proposta para detecção de vulnerabilidades em workflows do n8n.

4 Objetivos e Métricas de Sucesso

Esta seção estabelece os objetivos gerais e específicos do trabalho, definindo critérios quantitativos e qualitativos para avaliar o sucesso da validação da abordagem proposta. É importante ressaltar que o foco não é desenvolver uma ferramenta SAST completa de produção, mas sim validar a viabilidade técnica da abordagem híbrida e da metodologia de geração de regras assistida por inteligência artificial.

4.1 Objetivo Geral

O objetivo geral é validar a **viabilidade técnica** de uma abordagem híbrida de análise estática para workflows n8n. Especificamente:

1. Demonstrar que análise estática de estruturas JSON declarativas é possível e efetiva.
2. Desenvolver metodologia de geração assistida de regras usando IA para acelerar desenvolvimento.
3. Avaliar comparativamente Agentic Radar e Semgrep, documentando coberturas e limitações.

O trabalho não visa criar ferramenta de produção, mas estabelecer base científica que comprove viabilidade da abordagem.

4.2 Objetivos Específicos

Os objetivos específicos são:

OE1: Análise comparativa: Avaliar cobertura de vulnerabilidades, limitações técnicas e aplicabilidade ao contexto n8n, produzindo métricas quantitativas e caracterização qualitativa.

OE2: Metodologia com IA: Criar abordagem sistemática para geração de regras Semgrep usando Claude 3.5 Sonnet, incluindo estruturação de prompts e processo iterativo de refinamento.

OE3: Regras customizadas: Desenvolver regras para classes não cobertas pelo Agentic Radar, incluindo padrões, condições lógicas, metadados e mensagens explicativas.

OE4: Validação empírica: Executar ferramentas em workflows de teste, documentar detecções e calcular métricas de cobertura híbrida.

OE5: Documentação de lacunas: Identificar limitações, classes parcialmente cobertas e oportunidades para pesquisa futura.

4.3 Métricas de Avaliação

A avaliação do sucesso do trabalho será realizada através de critérios objetivos quantitativos e critérios qualitativos que, em conjunto, permitem caracterizar a viabilidade e utilidade da abordagem proposta.

4.3.1 Métricas Quantitativas

As métricas quantitativas estabelecem valores-alvo objetivos para aspectos mensuráveis do trabalho, permitindo avaliação empírica do alcance dos objetivos estabelecidos.

Tabela 1: Métricas quantitativas e valores-alvo

Métrica	Definição	Meta
Cobertura de Vulnerabilidades (Híbrida)	Percentual das 6 classes de vulnerabilidades com detecção efetiva através da abordagem híbrida (Agentic Radar + Semgrep)	$\geq 80\%$
Cobertura Agentic Radar	Percentual das 6 classes adequadamente cobertas pelo Agentic Radar isoladamente	Documentar % real
Cobertura Semgrep	Percentual das 6 classes cobertas por regras Semgrep customizadas	Documentar % real
Regras Desenvolvidas	Número de regras Semgrep funcionais criadas utilizando metodologia assistida por IA	≥ 4 regras (uma por classe não coberta)
Redução de Tempo	Economia de tempo no desenvolvimento de regras com IA vs. estimativa de desenvolvimento manual (8-12h por regra)	Documentar economia em horas

A cobertura de vulnerabilidades é calculada como proporção de classes com detecção efetiva. A meta de 80% reconhece que algumas categorias podem requerer capacidades além do escopo das ferramentas.

A redução de tempo compara desenvolvimento com IA versus estimativa manual (8-12h por regra), validando quantitativamente o valor da geração assistida.

4.3.2 Métricas Qualitativas

As métricas qualitativas avaliam aspectos não diretamente quantificáveis, mas essenciais para caracterizar a viabilidade prática e utilidade da abordagem proposta. Cada métrica é avaliada através de critérios específicos de aceitação.

Viabilidade Técnica:

A validação fundamental do trabalho é demonstrar que a análise estática de workflows n8n é tecnicamente viável e efetiva. Os critérios de avaliação incluem:

- A análise estática de arquivos JSON de workflows n8n permite identificação de padrões inseguros com precisão suficiente para ser útil em ambientes reais?
- As regras Semgrep geradas por IA utilizando Claude 3.5 Sonnet são funcionais e detectam corretamente as vulnerabilidades-alvo nos workflows de teste?
- A abordagem híbrida fecha as lacunas de cobertura das ferramentas isoladas, oferecendo cobertura complementar efetiva?
- As limitações identificadas são documentadas de forma clara e honesta, permitindo avaliação realista da aplicabilidade da solução?

Usabilidade:
Embora o foco não seja desenvolver uma ferramenta de produção completa, a viabilidade prática requer que as ferramentas sejam razoavelmente acessíveis:

- A instalação e configuração do Agentic Radar e Semgrep é viável com complexidade aceitável (máximo de 5 comandos por ferramenta)?
- A documentação disponível (tanto das ferramentas quanto da metodologia proposta) é suficiente para permitir reprodução independente do estudo por outros pesquisadores?
- A execução das análises sobre workflows de teste é direta, sem requerer expertise profunda em detalhes de implementação das ferramentas?

Clareza de Resultados:
A utilidade prática das detecções depende da qualidade da comunicação dos resultados:

- Os relatórios gerados pelas ferramentas identificam claramente as vulnerabilidades detectadas, incluindo localização exata no workflow (nó e parâmetro específico)?
- Existe mapeamento explícito das detecções para categorias reconhecidas do OWASP LCNC Top 10, facilitando comunicação com stakeholders?
- As explicações técnicas fornecidas são compreensíveis e as sugestões de remediação são acionáveis?
- A documentação distingue claramente entre detecções confirmadas (true positives) e potenciais falsos positivos, orientando o analista na interpretação dos resultados?

4.4 Benchmarks Comparativos

A avaliação da contribuição da abordagem híbrida requer comparação sistemática com três cenários baseline, cada um representando uma estratégia alternativa de análise de segurança para workflows n8n. A Tabela 2 sintetiza as coberturas esperadas para cada cenário.

Tabela 2: Benchmarks comparativos de cobertura de vulnerabilidades

Cenário	Cobertura Esperada	Observações
Agentic Radar isolado	∼16,7% (1/6 classes)	Baseline 1: Ferramenta especializada com foco em riscos emergentes de agentes de IA (injeção de prompt, vazamento de PII, geração de conteúdo prejudicial). Não cobre vulnerabilidades tradicionais de aplicações web.
Semgrep isolado	∼66,7% (4/6 classes)	Baseline 2: Potencial teórico para vulnerabilidades tradicionais (SQL Injection, Command Injection, SSRF, Secret Sprawl), mas requer desenvolvimento completo de regras customizadas. Sem regras, cobertura é 0%.
Abordagem Híbrida	∼83-100% (5-6/6 classes)	Proposta: Combinação complementar de Agentic Radar (riscos de IA) + Semgrep customizado (vulnerabilidades tradicionais). Maximiza cobertura ao combinar forças de ambas as ferramentas.

O cenário **Agentic Radar isolado** (16,7%) demonstra que ferramentas especializadas são insuficientes como solução autônoma, estabelecendo o limite inferior.

O cenário **Semgrep isolado** (66,7% potencial) evidencia a barreira crítica: sem regras customizadas, a cobertura é zero, estabelecendo o limite superior para ferramentas tradicionais.

A **Abordagem Híbrida** visa superar ambas limitações através de estratégia complementar, com meta de 83-100% de cobertura.

A validação envolverá execução dos três cenários sobre workflows de teste, permitindo comparação direta.

4.5 Critérios de Sucesso

O trabalho será considerado bem-sucedido se os seguintes critérios forem satisfeitos:

1. **Viabilidade Técnica:** Ambas ferramentas analisam workflows n8n e detectam ao menos uma vulnerabilidade cada.
2. **IA Validada:** Claude 3.5 Sonnet gera regras funcionais em tempo médio inferior a 2h por regra (economia de 75% vs. manual).
3. **Cobertura Complementar:** Abordagem híbrida cobre mínimo de 80% das classes (5 de 6).
4. **Lacunas Documentadas:** Limitações claramente identificadas, tecnicamente justificadas, com oportunidades específicas para trabalhos futuros.
5. **Reprodutibilidade:** Metodologia documentada com detalhes suficientes para replicação independente.

A satisfação dos cinco critérios estabelece que a abordagem híbrida é viável e representa contribuição significativa para segurança de plataformas LCNC.

5 Escopo da Solução e Requisitos

Esta seção detalha o escopo da solução proposta, definindo claramente os casos de uso cobertos, a arquitetura do sistema, os requisitos técnicos e a composição dos dados de validação. O foco é estabelecer expectativas realistas sobre o que será entregue no contexto de validação de viabilidade técnica, distinguindo explicitamente entre funcionalidades incluídas e excluídas do escopo.

5.1 Casos de Uso

Os casos de uso definem as interações principais entre os usuários e o sistema, estabelecendo o escopo funcional da solução proposta. O foco está na análise estática de segurança de workflows individuais e em lote, reconhecendo que este é um projeto de validação de viabilidade técnica, não uma ferramenta de produção completa.

5.1.1 Caso de Uso Principal: Análise de Segurança de Workflow n8n

O cenário principal de uso representa o fluxo típico de análise de segurança executado por um desenvolvedor ou analista de segurança que deseja avaliar a postura de segurança de um workflow específico do n8n.

Ator primário: Desenvolvedor ou Analista de Segurança

Pré-condições:

- O workflow do n8n foi exportado em formato JSON através da interface da plataforma
- As ferramentas de análise (Agentic Radar e Semgrep) estão instaladas e configuradas no ambiente
- O conjunto de regras customizadas do Semgrep está disponível e atualizado
- O ambiente possui conectividade de rede (caso o Agentic Radar utilize o modo `test` que requer API da OpenAI)

Fluxo principal:

1. O usuário fornece o arquivo JSON do workflow como entrada para o sistema de análise
2. O sistema executa a análise híbrida, invocando sequencialmente:
    - Agentic Radar em modo `scan` para detecção de riscos de IA agêntica
    - Semgrep com conjunto de regras customizadas para vulnerabilidades tradicionais
3. O sistema processa os resultados de ambas as ferramentas, consolidando as detecções e eliminando duplicatas
4. O sistema classifica as vulnerabilidades detectadas por severidade (Crítica, Alta, Média, Baixa) e mapeia cada detecção à categoria correspondente do OWASP LCNC Top 10

5. O sistema gera um relatório consolidado em formato estruturado (JSON) e legível por humanos (HTML ou texto formatado)

6. O usuário revisa o relatório gerado, compreende as vulnerabilidades identificadas através das explicações técnicas e sugestões de remediação, e prioriza as correções baseando-se na severidade e categoria OWASP

Fluxos alternativos:

- **Análise em lote:** O usuário fornece um diretório contendo múltiplos arquivos JSON de workflows. O sistema itera sobre todos os arquivos, executando a análise híbrida em cada um, e gera um relatório agregado identificando workflows por nome e consolidando estatísticas de vulnerabilidades por tipo e severidade.

Pós-condições:

- Relatório de análise de segurança disponível para revisão
- Vulnerabilidades documentadas com localização específica no workflow (identificador do nó, parâmetro afetado)
- Recomendações de remediação disponíveis para cada vulnerabilidade detectada

5.1.2 Casos Excluídos do Escopo

Para manter o foco na validação de viabilidade técnica da abordagem híbrida e da metodologia de geração de regras assistida por IA, os seguintes casos de uso são explicitamente excluídos do escopo deste trabalho:

- **Análise dinâmica ou execução real de workflows:** O sistema não executará workflows para detectar vulnerabilidades em tempo de execução. A análise é puramente estática, operando sobre a estrutura JSON declarativa.
- **Interface gráfica completa para usuários finais:** Não será desenvolvida uma interface web ou desktop. A interação ocorre via linha de comando (CLI), adequada para usuários técnicos e integração futura em pipelines automatizados.
- **Correção automática de vulnerabilidades detectadas:** O sistema identifica e reporta vulnerabilidades, mas não modifica automaticamente os workflows para corrigi-las. A remediação permanece responsabilidade do desenvolvedor.
- **Análise de nós customizados da comunidade:** O foco está nos nós oficiais da base do n8n (`n8n-nodes-base.*`). Nós desenvolvidos pela comunidade ou customizados não são cobertos pelas regras de detecção.
- **Otimização de performance de workflows:** O sistema não avalia eficiência computacional, uso de recursos ou otimizações de performance. O foco é exclusivamente em vulnerabilidades de segurança.
- **Integração em pipelines CI/CD com bloqueio automático:** Embora o Semgrep possua integrações nativas com plataformas de CI/CD, a configuração e implantação de integrações automatizadas em ambientes de produção estão fora do escopo deste trabalho de validação.

- **Visualização gráfica de fluxos de trabalho:** Não será desenvolvido um componente de renderização de grafos interativos dos workflows. Embora o Agentic Radar possua esta capacidade, a replicação ou integração deste recurso não é essencial para validar a viabilidade da detecção de vulnerabilidades.

5.2 Arquitetura Proposta

A arquitetura do sistema fundamenta-se em uma abordagem híbrida que combina as forças complementares do Agentic Radar e do Semgrep para alcançar cobertura abrangente das seis classes de vulnerabilidades identificadas. O design privilegia modularidade, permitindo que cada ferramenta opere independentemente e que os resultados sejam agregados de forma consistente.

5.2.1 Visão Geral da Arquitetura em Camadas

A arquitetura é organizada em quatro camadas principais que representam o fluxo de processamento desde a entrada de dados até a geração de relatórios:

1. **Camada de Entrada de Dados:** Responsável pela leitura e validação de arquivos JSON de workflows, verificando conformidade com o esquema esperado do n8n.

2. **Camada de Motores de Análise:** Executa as ferramentas de detecção de vulnerabilidades em paralelo ou sequencialmente, invocando o Agentic Radar e o Semgrep sobre os workflows de entrada.

3. **Camada de Agregação e Classificação:** Consolida os resultados heterogêneos das duas ferramentas em um formato unificado, elimina duplicatas e classifica detecções por severidade e categoria OWASP.

4. **Camada de Geração de Relatórios:** Produz relatórios estruturados e legíveis contendo as vulnerabilidades detectadas, explicações técnicas e recomendações de remediação.

Esta arquitetura em camadas promove separação de responsabilidades, facilitando testes unitários de cada componente e permitindo substituição ou atualização de ferramentas individuais sem impacto no sistema completo.

5.2.2 Componentes Principais

A implementação da arquitetura proposta requer cinco componentes principais, cada um com responsabilidades claramente definidas:

A. Módulo de Entrada e Validação

Este componente é responsável pela interface entre o usuário e o sistema de análise. Suas funções incluem:

- Parser JSON robusto que lê arquivos de workflow e valida a estrutura contra o esquema esperado (presença dos arrays `nodes` e `connections`, estrutura de objetos de nó)

- Extração de metadados do workflow (nome, versão, contagem de nós, tipos de nós presentes)

- Tratamento de erros para arquivos malformados ou inválidos, gerando mensagens informativas para o usuário
- Suporte a análise em lote, iterando sobre diretórios de workflows e mantendo associação entre arquivos e resultados

A validação estrutural neste estágio previne falhas nas ferramentas de análise downstream e fornece feedback rápido sobre problemas de formato.

B. Motor de Análise Híbrida

Este componente orquestra a execução das duas ferramentas de análise, gerenciando suas invocações e capturando suas saídas. Consiste em dois submódulos:

- **Executor do Agentic Radar:** Invoca o comando `agentic-radar scan` sobre o arquivo JSON do workflow, captura a saída (tipicamente em formato HTML ou JSON estruturado), e extrai as vulnerabilidades relacionadas a riscos de IA agêntica. Este submódulo é responsável pela cobertura de aproximadamente 16,7% das classes de vulnerabilidades, focando em injeção de prompt, vazamento de PII através de respostas de modelos de linguagem e geração de conteúdo prejudicial.
- **Executor do Semgrep:** Invoca o comando `semgrep –config=rules/ –json` sobre o arquivo JSON do workflow, utilizando o conjunto de regras customizadas desenvolvidas especificamente para o esquema do n8n. Captura a saída em formato JSON estruturado (SARIF ou formato nativo do Semgrep) e extrai as vulnerabilidades relacionadas a SQL Injection, Command Injection, SSRF, Secret Sprawl, Insecure Data Handling e DoS. Este submódulo é responsável pela cobertura potencial de aproximadamente 66,7% das classes de vulnerabilidades.

A execução pode ser paralela (para maximizar velocidade) ou sequencial (para simplificar tratamento de erros), dependendo das restrições de recursos do ambiente.

C. Módulo de Agregação e Normalização

Este componente recebe as saídas heterogêneas das duas ferramentas e as transforma em um formato unificado. Suas responsabilidades incluem:

- Parsing dos formatos de saída específicos de cada ferramenta (HTML/JSON do Agentic Radar, JSON/SARIF do Semgrep)
- Normalização das detecções em um esquema comum contendo: identificador único, descrição da vulnerabilidade, localização exata no workflow (nó e parâmetro), severidade, categoria de vulnerabilidade e ferramenta que detectou
- Eliminação de duplicatas através de comparação de localização e tipo de vulnerabilidade (caso ambas as ferramentas detectem a mesma instância)
- Classificação de severidade em quatro níveis (Crítica, Alta, Média, Baixa) baseando-se na severidade reportada pelas ferramentas e na categoria OWASP associada
- Enriquecimento com contexto adicional quando disponível (trecho do JSON afetado, linha no arquivo)

D. Módulo de Mapeamento OWASP

Este componente enriquece cada vulnerabilidade detectada com classificações padronizadas de segurança:

- Mapeamento explícito para categorias do OWASP LCNC Top 10 (LCNC-SEC-05, LCNC-SEC-06, LCNC-SEC-07, LCNC-SEC-08) baseando-se no tipo de vulnerabilidade detectada
- Enriquecimento com códigos CWE (Common Weakness Enumeration) quando aplicável (ex.: CWE-89 para SQL Injection, CWE-78 para Command Injection, CWE-918 para SSRF)
- Adição de referências a documentação relevante (links para páginas do OWASP LCNC Top 10, artigos de mitigação, melhores práticas do n8n)

Este mapeamento facilita a comunicação de riscos com stakeholders não técnicos através de taxonomias reconhecidas pela indústria.

E. Gerador de Relatórios

Este componente produz as saídas finais consumidas pelos usuários:

- **Relatório estruturado em JSON:** Formato machine-readable contendo todas as detecções com metadados completos, adequado para processamento automatizado ou integração com outras ferramentas
- **Relatório legível em HTML ou texto formatado:** Apresentação organizada das vulnerabilidades agrupadas por severidade e categoria OWASP, incluindo explicações técnicas e sugestões de remediação
- **Sumário executivo:** Estatísticas agregadas (contagem de vulnerabilidades por tipo, por severidade, por nó afetado) fornecendo visão geral rápida da postura de segurança do workflow

5.3 Requisitos Técnicos

Os requisitos técnicos especificam as tecnologias, frameworks e infraestrutura necessários para implementação e execução da solução proposta.

5.3.1 Tecnologias e Frameworks

A seleção de tecnologias priorizou maturidade, compatibilidade com as ferramentas de análise e acessibilidade para a comunidade de segurança. A Tabela 3 sintetiza o stack tecnológico.

Tabela 3: Stack tecnológico da solução proposta

Camada	Tecnologia	Justificativa
Análise IA Agentes	Agentic Radar	Única ferramenta open-source especializada em riscos de IA agêntica para workflows de automação, com suporte comprovado para n8n
Análise Tradicional	Semgrep	Motor SAST maduro e amplamente adotado, com suporte nativo a JSON, linguagem de regras acessível e capacidades de taint analysis
Linguagem Principal	Python 3.10+	Compatibilidade com ambas as ferramentas (Agentic Radar requer Python >= 3.10 < 3.13), ecossistema rico de bibliotecas para processamento JSON e geração de relatórios
Processamento JSON	Python `json`	Biblioteca padrão para parsing e manipulação de workflows, sem dependências externas
Geração de Regras	Claude 3.5 Sonnet (API Anthropic)	Modelo de linguagem de grande capacidade para geração assistida de regras Semgrep, acelerando desenvolvimento através de prompts estruturados e few-shot learning

A escolha do Python como linguagem principal é justificada pela necessidade de compatibilidade com as ferramentas de análise e pela disponibilidade de bibliotecas robustas para as tarefas requeridas. A utilização de Claude 3.5 Sonnet para geração de regras representa uma inovação metodológica, reduzindo drasticamente o tempo de desenvolvimento de regras customizadas de 8-12 horas por regra para aproximadamente 2 horas, conforme será demonstrado na validação experimental.

5.3.2 Infraestrutura Mínima

Os requisitos de infraestrutura são modestos, refletindo a natureza de análise estática da solução que não requer recursos computacionais intensivos como GPUs ou grandes quantidades de memória.

Ambiente de Desenvolvimento:

- **CPU:** 2 cores (análise estática não requer processamento intensivo)
- **RAM:** 4GB (suficiente para parsing de workflows típicos de até 100 nós)
- **Armazenamento:** 2GB (instalação de ferramentas, regras customizadas e workflows de teste)
- **Python:** Versão 3.10, 3.11 ou 3.12 (compatibilidade com Agentic Radar)
- **Sistema Operacional:** Linux, macOS ou Windows (ferramentas são multiplataforma)

Ambiente de Execução:

- Ambiente local ou containerizado (Docker) para portabilidade
- Capacidade de execução paralela opcional para análise em lote de múltiplos workflows
- Conectividade de rede caso o modo `test` do Agentic Radar seja utilizado (requer acesso à API da OpenAI)

A infraestrutura mínima garante que a solução possa ser executada em laptops de desenvolvimento típicos, sem necessidade de servidores dedicados ou infraestrutura cloud, reduzindo barreiras de adoção e custos operacionais.

5.3.3 Dados de Validação

A validação da abordagem proposta requer um dataset controlado de workflows com vulnerabilidades conhecidas, servindo como ground truth para avaliar precisão, recall e taxa de falsos positivos das ferramentas.

Composição do Dataset:

O dataset de validação consiste em **3 workflows sintéticos** desenvolvidos especificamente para este trabalho, cada um implementando deliberadamente 2 das 6 classes de vulnerabilidades identificadas. Esta abordagem garante cobertura balanceada de todas as categorias e permite validação controlada onde a localização exata e a natureza de cada vulnerabilidade são conhecidas a priori.

A Tabela 4 detalha a composição do dataset de validação.

Tabela 4: Composição do dataset de workflows de validação

Workflow	**Vulnerabilidades Implementadas**	**Nós e Padrões Envolvidos**
Workflow 1: Integração de Dados	SQL Injection + SSRF	Nó Webhook recebendo entrada externa → Nó MySQL com query dinâmica sem sanitização; Nó HTTP Request com URL construída dinamicamente
Workflow 2: Automação de Sistema	Command Injection + Secret Sprawl	Nó Webhook → Nó Execute Command com parâmetro `command` contendo expressão dinâmica; Credenciais hardcoded em parâmetros de configuração
Workflow 3: Processamento de Dados Sensíveis	Insecure Data Handling + DoS	Nó HTTP Request usando `http://` sem TLS para transmissão de dados; Estrutura de loop sem condição de saída clara no grafo de conexões

Documentação de Ground Truth:

Cada workflow no dataset é acompanhado de documentação detalhada especificando:

- **Localização exata:** Identificador do nó afetado, nome do parâmetro contendo a vulnerabilidade, linha no arquivo JSON
- **Categoria OWASP:** Mapeamento para a categoria relevante do OWASP LCNC Top 10 (LCNC-SEC-05, LCNC-SEC-06, LCNC-SEC-07, LCNC-SEC-08)
- **Severidade esperada:** Classificação da severidade (Crítica, Alta, Média, Baixa) baseada no potencial de impacto e exploração

- **Descrição técnica:** Explicação de como a vulnerabilidade se manifesta no workflow específico e por que é insegura
- **Mitigação sugerida:** Exemplo de correção que remediaria a vulnerabilidade

Esta documentação serve como baseline para cálculo de métricas de validação: uma detecção é considerada True Positive se a ferramenta identifica corretamente a vulnerabilidade na localização documentada e com classificação de severidade compatível (variação de ± 1 nível é aceitável). A ausência de detecção de uma vulnerabilidade documentada constitui False Negative, enquanto detecções em código seguro (inexistentes neste dataset sintético) constituiriam False Positives.

Considerações de Privacidade:

Os workflows sintéticos não contêm dados reais, credenciais válidas ou referências a sistemas de produção. Toda lógica de negócio, URLs, credenciais e dados são fictícios, garantindo que o dataset possa ser compartilhado publicamente sem riscos de privacidade ou conformidade. Esta abordagem também permite foco exclusivo nas estruturas de vulnerabilidades, sem distrações de lógica de negócio complexa de workflows reais.

6 Ameaças, Riscos e Controles

Esta seção aplica metodologias estruturadas de análise de segurança para identificar sistematicamente as ameaças, avaliar riscos e estabelecer controles de mitigação para a ferramenta SAST proposta. A análise fundamenta-se no modelo STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) para categorização de ameaças, complementada por considerações específicas sobre o uso de inteligência artificial e requisitos de privacidade conforme a LGPD.

6.1 Superfícies de Ataque

A identificação das superfícies de ataque é fundamental para compreender os pontos de entrada e dependências que podem ser explorados por atacantes. A arquitetura híbrida proposta, combinando Agentic Radar e Semgrep com geração assistida de regras utilizando IA, apresenta quatro superfícies de ataque principais:

1. Entrada de Dados (Workflows JSON): A ferramenta processa arquivos JSON de workflows do n8n fornecidos pelo usuário. Estes arquivos podem ser malformados intencionalmente, contendo estruturas profundamente aninhadas projetadas para explorar vulnerabilidades no parser JSON, ou podem conter payloads maliciosos embutidos em expressões dinâmicas que tentam enganar os motores de análise.

2. Integrações com APIs de IA Externas: A metodologia de geração assistida de regras depende da API da Anthropic (Claude 3.5 Sonnet) para criação de regras Semgrep, enquanto o Agentic Radar, quando utilizado no modo `test`, requer comunicação com a API da OpenAI para execução de testes adversariais. Estas dependências externas introduzem riscos de privacidade, disponibilidade e integridade.

3. Base de Conhecimento (Regras e Configurações): O conjunto de regras Semgrep customizadas, armazenadas em arquivos YAML, constitui a base de conhecimento de segurança da ferramenta. A adulteração destas regras pode resultar em cegueira intencional a vulnerabilidades específicas, comprometendo a efetividade da análise.

4. Ambiente de Execução Local: A ferramenta é executada localmente através de interface de linha de comando (CLI), processando workflows no ambiente do usuário. Vulnerabilidades em bibliotecas dependentes (parser JSON, motor Semgrep, Agentic Radar) ou configurações inseguras do ambiente podem ser exploradas para execução de código arbitrário.

6.2 Análise STRIDE

A aplicação sistemática do modelo STRIDE permite categorizar ameaças de segurança e estabelecer controles de mitigação correspondentes. A Tabela 5 apresenta a matriz completa de ameaças identificadas para a ferramenta SAST proposta.

Tabela 5: Matriz de ameaças STRIDE para a ferramenta SAST

Categoria	Ameaça	Controle
Spoofing	Falsificação de identidade de workflows ou adulteração de metadados para mascarar origem maliciosa	Validação de assinatura digital de workflows (quando disponível) e verificação de integridade de arquivos JSON através de hash SHA-256 antes do processamento
Tampering	Modificação maliciosa das regras Semgrep locais para criar cegueira intencional a vulnerabilidades específicas	Controle de versão das regras através de Git, verificação de integridade (hash) antes da execução, e assinatura digital das regras oficiais
Repudiation	Ausência de registros auditáveis sobre execuções da ferramenta, impedindo rastreabilidade de análises realizadas	Geração de logs estruturados (formato JSON) contendo timestamp, hash do workflow analisado, versão das regras utilizadas e resumo de detecções, armazenados localmente com proteção de integridade
Information Disclosure	Vazamento de credenciais hardcoded em workflows através de relatórios de segurança ou logs; transmissão de lógica de negócio proprietária para APIs de IA externas	Redação automática de padrões de segredos (chaves de API, tokens) nos relatórios; anonimização de dados antes do envio para APIs de IA; preferência por análise local (Semgrep) para workflows contendo informações sensíveis
Denial of Service	Submissão de "JSON Bombs"(arquivos com aninhamento profundo ou estruturas recursivas) projetados para estourar memória do parser ou travar o motor de análise	Validação de limites de recursos: profundidade máxima de aninhamento JSON (32 níveis), tamanho máximo de arquivo (10MB), timeout de análise (60 segundos), e monitoramento de uso de memória durante o parsing
Elevation of Privilege	Exploração de vulnerabilidades em bibliotecas dependentes (parser JSON, motor Semgrep) ou execução de código JavaScript embutido em expressões dinâmicas do n8n durante análise	Execução em ambiente isolado (sandbox) quando disponível; validação rigorosa de entrada antes do parsing; uso de versões atualizadas e auditadas de dependências; desabilitação de execução de código JavaScript durante análise estática

6.2.1 Análise Detalhada por Categoria

Spoofing: O risco de falsificação é relativamente baixo em execução local, onde o usuário possui controle direto sobre os arquivos analisados. No entanto, em cenários onde workflows são

compartilhados entre equipes ou obtidos de fontes externas, existe o risco de adulteração de metadados (nome do workflow, versão) para mascarar a origem maliciosa. A implementação de verificação de integridade através de hash SHA-256 permite detectar modificações não autorizadas, enquanto assinaturas digitais (quando disponíveis) fornecem garantias de autenticidade.

Tampering: A adulteração das regras Semgrep representa uma ameaça crítica, pois pode resultar em cegueira intencional a vulnerabilidades específicas, criando uma falsa sensação de segurança. O controle de versão através de Git permite rastreabilidade completa de modificações, enquanto verificação de hash antes da execução detecta adulterações não autorizadas. Para ambientes de produção, recomenda-se assinatura digital das regras oficiais utilizando chaves criptográficas gerenciadas centralmente.

Repudiation: A ausência de registros auditáveis impede a rastreabilidade de análises realizadas, dificultando investigações de segurança e conformidade regulatória. A geração de logs estruturados em formato JSON, contendo timestamp, hash do workflow, versão das regras e resumo de detecções, estabelece um registro imutável de todas as execuções. A proteção de integridade dos logs através de hash ou assinatura digital previne adulteração posterior.

Information Disclosure: Esta categoria apresenta riscos críticos devido à natureza sensível dos dados processados. Workflows podem conter credenciais hardcoded (chaves de API, tokens de acesso, senhas), lógica de negócio proprietária e referências a infraestrutura interna. A redação automática de padrões de segredos nos relatórios (substituição por `[REDACTED]`) previne vazamento acidental através de compartilhamento de relatórios. A anonimização de dados antes do envio para APIs de IA (remoção de identificadores, substituição de valores por placeholders) protege propriedade intelectual. Para workflows altamente sensíveis, recomenda-se desabilitar completamente o uso de APIs externas, utilizando apenas análise local do Semgrep.

Denial of Service: Ataques de negação de serviço podem ser executados através de arquivos JSON malformados projetados para explorar vulnerabilidades no parser ou consumir recursos computacionais excessivos. "JSON Bombs"com aninhamento profundo podem causar estouro de pilha (stack overflow) em parsers recursivos, enquanto estruturas recursivas podem resultar em loops infinitos durante a análise. A implementação de limites rigorosos de recursos (profundidade máxima, tamanho de arquivo, timeout) e monitoramento de uso de memória previne estes ataques, garantindo que a ferramenta falhe graciosamente sem comprometer a estabilidade do sistema.

Elevation of Privilege: A exploração de vulnerabilidades em bibliotecas dependentes ou execução de código JavaScript embutido em expressões dinâmicas do n8n durante análise estática representa um risco significativo. Embora a análise estática não deva executar código, parsers mal implementados podem inadvertidamente processar expressões de forma insegura. A execução em ambiente isolado (sandbox), quando disponível, limita o impacto de explorações. A validação rigorosa de entrada antes do parsing e o uso de versões atualizadas e auditadas de dependências reduzem a superfície de ataque.

6.3 Riscos Específicos de Sistemas Baseados em IA

A integração de inteligência artificial na ferramenta proposta, através da geração assistida de regras utilizando Claude 3.5 Sonnet e do Agentic Radar que utiliza modelos de linguagem para análise, introduz categorias específicas de riscos que transcendem vulnerabilidades tradicionais de software.

6.3.1 Privacidade e Confidencialidade de Dados

O envio de descrições de vulnerabilidades, exemplos de código ou trechos de workflows para APIs de IA externas (Anthropic, OpenAI) cria riscos significativos de vazamento de propriedade intelectual e lógica de negócio proprietária. Workflows corporativos frequentemente contêm referências a sistemas internos, padrões de integração específicos da organização e estruturas de dados que revelam informações estratégicas.

Controles de Mitigação:

- **Anonimização Proativa:** Remoção de identificadores únicos (nomes de sistemas, URLs internas, referências a clientes) antes do envio para APIs externas, substituindo por placeholders genéricos.
- **Política de Retenção Zero:** Preferência por modelos de IA que oferecem políticas de retenção zero (Enterprise), garantindo que dados não sejam utilizados para treinamento de modelos futuros.
- **Modelos Auto-hospedados:** Para organizações com requisitos rigorosos de conformidade, considerar migração futura para modelos de IA auto-hospedados (ex.: Llama 2, Mistral) que eliminam completamente a transmissão de dados para serviços externos.
- **Segmentação de Dados:** Limitar o escopo de dados enviados para APIs, transmitindo apenas trechos mínimos necessários para geração de regras, evitando envio de workflows completos.

6.3.2 Integridade e Alucinação de Modelos de Linguagem

Modelos de linguagem são suscetíveis a "alucinações"— geração de conteúdo sintaticamente válido mas logicamente incorreto ou factualmente impreciso. No contexto de geração de regras Semgrep, uma alucinação pode resultar em regras que são válidas sintaticamente (passam validação YAML) mas não detectam corretamente as vulnerabilidades-alvo, criando falsos negativos que geram falsa sensação de segurança.

Controles de Mitigação:

- **Validação Humana Obrigatória (Human-in-the-Loop):** Todas as regras geradas por IA devem ser submetidas a revisão humana antes da implantação em ambientes de produção. A IA acelera o desenvolvimento, mas não substitui o julgamento de especialistas em segurança.
- **Testes Automatizados contra Ground Truth:** Validação sistemática de regras geradas contra o dataset de validação com vulnerabilidades conhecidas, calculando métricas de precisão, recall e F1-score para identificar regras deficientes.
- **Refinamento Iterativo:** Processo estruturado de refinamento onde regras geradas são testadas, avaliadas, e refinadas através de prompts ajustados com feedback, convergindo para regras de alta qualidade após 2-4 iterações típicas.
- **Documentação de Limitações:** Transparência explícita sobre quais vulnerabilidades são cobertas e quais são limitações conhecidas, evitando over-trust na automação.

6.3.3 Disponibilidade e Dependência de Serviços Externos

A dependência de APIs de IA externas (Anthropic, OpenAI) para geração de regras e execução de testes adversariais cria um ponto único de falha. Interrupções na disponibilidade destes serviços, limitações de taxa (rate limiting) ou custos operacionais elevados podem impactar a capacidade de desenvolver novas regras ou executar análises completas.

Controles de Mitigação:

- **Arquitetura Híbrida com Degradação Graciosa:** O motor principal de análise (Semgrep) opera completamente offline, não dependendo de serviços externos para detecção de vulnerabilidades. A IA é utilizada exclusivamente para aceleração de desenvolvimento (design-time), não bloqueando a execução de análises (run-time).
- **Cache de Regras Geradas:** Armazenamento local de regras previamente geradas e validadas, permitindo reutilização sem necessidade de novas chamadas à API.
- **Fallback para Desenvolvimento Manual:** Metodologia documentada para desenvolvimento manual de regras Semgrep quando APIs de IA não estão disponíveis, garantindo continuidade operacional.
- **Monitoramento de Custos:** Implementação de limites de custo e alertas para prevenir gastos excessivos com APIs de IA em ambientes corporativos.

6.3.4 Segurança da Cadeia de Suprimentos de IA

A confiança na robustez e segurança dos modelos de linguagem utilizados pelas APIs externas representa um risco de segurança da cadeia de suprimentos. 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.

Controles de Mitigação:

- **Diversificação de Fornecedores:** Capacidade de alternar entre diferentes modelos de IA (Claude, GPT-4, modelos open-source) para reduzir dependência de um único fornecedor.
- **Validação Independente:** Testes sistemáticos de regras geradas por diferentes modelos contra o mesmo dataset de validação, identificando discrepâncias que podem indicar problemas de qualidade.
- **Auditoria de Comportamento:** Monitoramento de padrões de saída dos modelos para detectar degradação de qualidade ou mudanças de comportamento que possam indicar atualizações problemáticas nos modelos subjacentes.

6.4 Considerações de Privacidade (LGPD)

A aplicação da metodologia LINDDUN (Linkability, Identifiability, Non-repudiation, Detectability, Disclosure, Unawareness, Non-compliance) fornece uma análise sistemática de riscos de privacidade, particularmente relevante no contexto brasileiro onde a Lei Geral de Proteção de Dados (LGPD) estabelece requisitos rigorosos de proteção de dados pessoais.

Linkability: A capacidade de correlacionar múltiplas análises de workflows para identificar padrões de comportamento do usuário ou organização representa um risco de privacidade. Logs

de execução que armazenam hashes de workflows podem, teoricamente, ser correlacionados para rastrear evolução de práticas de desenvolvimento ao longo do tempo.

Controle: Minimização de dados nos logs, armazenando apenas metadados essenciais (timestamp, contagem de vulnerabilidades por tipo) sem identificadores que permitam correlação direta.

Identifiability: Workflows podem conter dados pessoais (PII) como endereços de e-mail, números de telefone ou identificadores de clientes. A análise estática processa estes dados, potencialmente expondo-os em relatórios ou logs.

Controle: Detecção automática de padrões de PII (expressões regulares para e-mails, CPF, telefones) e redação automática nos relatórios, substituindo por `[REDACTED]`.

Disclosure: O vazamento não autorizado de dados pessoais através de relatórios compartilhados ou transmissão para APIs de IA externas viola requisitos da LGPD.

Controle: Redação automática de PII antes de qualquer transmissão externa; preferência por análise local para workflows contendo dados pessoais; documentação explícita de medidas de proteção de dados.

Non-compliance: A falta de conformidade com requisitos da LGPD pode resultar em penalidades regulatórias significativas (até 2% do faturamento ou R$ 50 milhões).

Controle: Implementação de medidas técnicas e organizacionais documentadas (Art. 46 da LGPD), incluindo criptografia de dados em repouso, controle de acesso a logs e políticas de retenção de dados com prazo de expiração automática.

6.5 Considerações Éticas

A utilização de inteligência artificial em ferramentas de segurança introduz considerações éticas que transcendem aspectos puramente técnicos, requerendo compromissos explícitos de transparência, equidade e responsabilidade.

6.5.1 Transparência e Explicabilidade

A transparência refere-se à capacidade de usuários compreenderem como a ferramenta funciona, quais vulnerabilidades são detectadas e quais são limitações conhecidas. A explicabilidade é particularmente crítica quando IA é utilizada para geração de regras, pois usuários devem compreender a origem e confiabilidade das detecções.

Compromissos de Transparência:

- **Documentação Explícita de Cobertura:** Relatórios incluem mapeamento claro de quais classes de vulnerabilidades são cobertas pela abordagem híbrida e quais são limitações conhecidas (ex.: detecção de DoS requer análise complementar).
- **Identificação de Origem de Detecções:** Cada vulnerabilidade detectada identifica explicitamente qual ferramenta (Agentic Radar ou Semgrep) realizou a detecção, permitindo que usuários compreendam a base técnica da análise.
- **Metadados de Regras:** Regras Semgrep incluem metadados indicando se foram geradas por IA ou desenvolvidas manualmente, incluindo timestamp de criação e versão do modelo utilizado (quando aplicável).
- **Limitações Documentadas:** Admissão explícita de que análise estática não pode detectar todas as vulnerabilidades, especialmente aquelas que dependem de contexto de execução ou lógica de negócio complexa.

6.5.2 Equidade e Ausência de Vieses

Embora a ferramenta proposta não tome decisões que impactem diretamente indivíduos (não classifica pessoas, não determina acesso a recursos), a equidade manifesta-se através da consistência e imparcialidade da detecção de vulnerabilidades. Vieses podem surgir se regras geradas por IA favorecem certos padrões de código ou tipos de vulnerabilidades em detrimento de outros.

Estratégias de Mitigação:

- **Dataset Balanceado de Validação:** O conjunto de workflows de teste cobre todas as seis classes de vulnerabilidades de forma balanceada, garantindo que regras sejam avaliadas contra espectro completo de riscos.
- **Validação Cruzada:** Testes de regras geradas por IA contra múltiplos workflows representativos, identificando padrões de detecção inconsistentes que podem indicar vieses.
- **Revisão por Múltiplos Especialistas:** Validação humana de regras geradas por IA realizada por múltiplos analistas de segurança, reduzindo impacto de vieses individuais.

6.5.3 Responsabilidade e Accountability

A responsabilidade refere-se à capacidade de identificar quem é responsável por decisões tomadas pela ferramenta e garantir que existam mecanismos de prestação de contas. Embora a ferramenta seja executada localmente pelo usuário, a responsabilidade pela interpretação de resultados e decisões de remediação permanece com o usuário.

Mecanismos de Accountability:

- **Rastreabilidade de Regras:** Todas as regras incluem metadados de autoria (gerada por IA com identificação do modelo, ou desenvolvida manualmente com identificação do autor), permitindo rastreabilidade completa.
- **Versionamento de Conjuntos de Regras:** Controle de versão formal de conjuntos de regras através de Git, permitindo auditoria histórica de mudanças e rollback para versões anteriores se problemas forem identificados.
- **Documentação de Decisões de Design:** Justificativa técnica documentada para escolhas arquiteturais (por que Agentic Radar + Semgrep, por que IA para geração de regras), permitindo avaliação crítica por pares.

6.5.4 Direito de Contestação

Embora a ferramenta não tome decisões automatizadas sobre indivíduos (não é um sistema de decisão automatizada conforme Art. 20 da LGPD), usuários devem ter capacidade de contestar detecções que considerem incorretas (falsos positivos) ou reportar vulnerabilidades não detectadas (falsos negativos).

Processo de Contestação:

- **Feedback de Falsos Positivos:** Mecanismo documentado para usuários reportarem detecções incorretas, permitindo refinamento iterativo de regras e redução de taxas de falsos positivos.

- **Reporte de Falsos Negativos:** Processo para submissão de workflows com vulnerabilidades conhecidas não detectadas, permitindo identificação de lacunas de cobertura e desenvolvimento de regras adicionais.
- **Revisão de Contestações:** Processo estruturado de revisão de contestações por especialistas em segurança, com documentação de decisões e atualização de regras quando apropriado.

7 Avaliação Experimental

Esta seção apresenta os resultados da validação experimental da análise estática de segurança de workflows n8n utilizando Semgrep. O estudo documenta uma descoberta técnica crítica: a necessidade de adaptação do modo de análise (JSON vs. genérico) para plataformas Low-Code/No-Code, contribuindo para o conhecimento sobre aplicação de ferramentas SAST a paradigmas declarativos.

7.1 Protocolo Experimental

O protocolo experimental seguiu abordagem iterativa em duas fases: tentativa inicial com modo JSON (falha completa) seguida de pivô para modo genérico (sucesso parcial), permitindo comparação empírica das capacidades de cada abordagem.

7.1.1 Configuração de Hardware e Software

A análise foi conduzida em ambiente controlado com as seguintes especificações:

- **Sistema Operacional**: macOS 14.6 (Darwin 24.6.0)
- **Processador**: Apple Silicon (arquitetura ARM64)
- **Memória RAM**: 16GB
- **Python**: Versão 3.14
- **Semgrep**: Versão 1.144.0
- **Claude 3.5 Sonnet**: Via API Anthropic (geração assistida de regras)

7.1.2 Dataset de Validação

O dataset consiste em três workflows reais do n8n, totalizando 14 vulnerabilidades documentadas através de revisão manual especializada seguindo o framework OWASP LCNC Top 10. A Tabela 6 apresenta as características do dataset.

Tabela 6: Características do dataset de validação

Workflow	Nós	Linhas	Vulnerabilidades	Classes
workflow_1.json	67	2.650	2	Secret Deletion
workflow_2.json	100+	2.945	5	SQL Injection (3x), Command Injection (2x)
workflow_3.json	17	605	7	Diversos
TOTAL	**184+**	**6.200**	**14**	**6 classes**

7.2 Resultados

A avaliação experimental revelou diferenças dramáticas entre os dois modos de análise do Semgrep, documentando limitações arquiteturais fundamentais do modo JSON para estruturas LCNC.

7.2.1 Fase 1: Modo JSON (Falha Completa)

A tentativa inicial utilizou o modo JSON nativo do Semgrep, desenvolvendo 33 regras customizadas com assistência de IA (Claude 3.5 Sonnet) em 3,75 horas. As regras foram sintaticamente válidas e seguiam as convenções do Semgrep, cobrindo 7 classes de vulnerabilidades.

Execução: Semgrep executou sem erros em todos os workflows (tempo médio: 2,3s por workflow, memória: $\sim$55MB).

Detecções: Zero. Nenhuma das 14 vulnerabilidades documentadas foi detectada.

A Tabela 7 apresenta os resultados detalhados.

Tabela 7: Resultados da análise em modo JSON

Workflow	Detecções	Esperado	Cobertura	FN
workflow_1.json	0	2	0%	2
workflow_2.json	0	5	0%	5
workflow_3.json	0	7	0%	7
TOTAL	**0**	**14**	**0%**	**14 (100%)**

Análise de Causa Raiz:
Investigação detalhada revelou três limitações técnicas fundamentais:

1. **Aninhamento Profundo**: O parser JSON do Semgrep falha em aplicar padrões dentro de arrays profundamente aninhados (4+ níveis). A estrutura típica de workflows n8n (root $\rightarrow$ nodes $\rightarrow$ node $\rightarrow$ parameters) excede esta capacidade.

2. **Comportamento de Ellipsis**: O operador `...` (correspondência flexível) comporta-se diferentemente em modo JSON comparado a linguagens de código, não iterando sobre elementos de arrays conforme necessário.

3. **Extração de Metavariáveis**: Expressões n8n contendo `{{ $json.field }}` não foram corretamente capturadas via `metavariable-regex`, impedindo detecção de injeção.

Validação: Teste com regra mínima (detectar qualquer nó com campo `"type"`) também produziu zero detecções, confirmando limitação arquitetural, não erro de implementação.

7.2.2 Fase 2: Modo Genérico (Breakthrough)

Baseado na análise de causa raiz, pivotou-se para o **modo genérico** do Semgrep, que trata arquivos JSON como texto plano e aplica correspondência via expressões regulares (regex). Desenvolveram-se 7 regras focadas em padrões textuais específicos.

Execução: Semgrep executou com performance similar (tempo: $\sim$2s/workflow).

Detecções: 19 findings totais, incluindo 100% das vulnerabilidades críticas documentadas de SQL Injection e Command Injection, além de descoberta de 10 instâncias adicionais não documentadas no ground truth inicial.

A Tabela 8 apresenta os resultados detalhados.

Tabela 8: Resultados da análise em modo genérico

Workflow	Detecções	Principais Achados	Tempo
workflow_1.json	2	Operações Vault não autorizadas (2x)	1,8s
workflow_2.json	13	SQL Injection (3 doc + 10 novos)	2,1s
workflow_3.json	4	PGPASSWORD injection, SSH keys	1,9s
TOTAL	**19**	**6 classes detectadas**	**5,8s**

7.2.3 Comparação Direta: JSON vs. Genérico

A Tabela 9 compara diretamente os dois modos de análise.

Tabela 9: Comparação JSON mode vs. Generic mode

Métrica	JSON Mode	Generic Mode	Diferença
Regras desenvolvidas	33 (7 arquivos)	7 (1 arquivo)	-26 regras
Tempo desenvolvimento	3,75h	1,2h	-68%
Detecções totais	0	19	+19 (infinito)
Cobertura documentada	0% (0/14)	71% (10/14)	+71 pp
Falsos negativos	14 (100%)	4 (29%)	-71 pp
Complexidade regras	Alta (AST matching)	Média (regex)	Mais simples
Precisão semântica	Alta (teoricamente)	Baixa (texto plano)	Trade-off

7.2.4 Cobertura por Classe de Vulnerabilidade

A Tabela 10 apresenta a cobertura de detecção por classe de vulnerabilidade, comparando ambos os modos.

Tabela 10: Cobertura por classe de vulnerabilidade

Classe	Documentadas	JSON Mode		Generic Mode	
		Detectadas	Cov.	Detectadas	Cov.
SQL Injection	3	0	0%	13	100%+
Command Injection	2	0	0%	3	100%+
Vault Secrets	2	0	0%	2	100%
SSH Key Exposure	1	0	0%	1	100%
Hardcoded Passwords	3	0	0%	0	0%
SSRF/DoS	3	0	0%	0	0%
TOTAL	**14**	**0**	**0%**	**19**	**71%**

Observação Crítica: Modo genérico detectou 100%+ em SQL/Command Injection (incluindo instâncias não documentadas), demonstrando sensibilidade adequada. Classes não detecta-

das (hardcoded passwords, SSRF/DoS) requerem análise semântica mais sofisticada que regex pura não fornece.

7.3 Análise de Causa Raiz da Divergência

A diferença dramática entre os modos (0% vs. 71% de cobertura) decorre de incompatibilidades arquiteturais fundamentais entre o paradigma de análise do Semgrep e estruturas LCNC.

7.3.1 Paradigma Declarativo vs. Imperativo

Código Tradicional (Imperativo): Ferramentas SAST analisam fluxo de controle (if/else, loops) e fluxo de dados (variáveis, funções) através de ASTs de linguagens de programação. O Semgrep foi projetado para este paradigma.

Workflows n8n (Declarativo): Workflows são configurações JSON que *declaram* transformações de dados através de conexões entre nós. Não há controle de fluxo imperativo tradicional. O "código-fonte"é a configuração, não instruções executáveis.

Incompatibilidade: O parser JSON do Semgrep trata workflows como *dados de configuração*, não como *código*, falhando em reconhecer padrões inseguros dentro da estrutura aninhada. O modo genérico, ao tratar JSON como texto, inadvertidamente resolve o problema ao aplicar correspondência de padrões puramente textual, similar a grep avançado.

7.3.2 Limitação de Profundidade de Aninhamento

Estrutura típica de vulnerabilidade SQL Injection em n8n:

```
{
  "nodes": [              // Nível 1
    {                     // Nível 2
      "type": "...",
      "parameters": {     // Nível 3
        "query": "=..."  // Nível 4 - vulnerabilidade aqui
      }
    }
  ]
}
```

O modo JSON do Semgrep requer navegação via `pattern-inside` aninhados para alcançar o nível 4, mas esta construção falha em arrays. O modo genérico simplesmente procura pelo texto `"query": "=` diretamente, ignorando a estrutura hierárquica.

7.4 Discussão

Os resultados demonstram que **a escolha do modo de análise é tão crítica quanto a seleção da ferramenta** para análise de segurança de plataformas LCNC.

7.4.1 Trade-off: Simplicidade vs. Precisão Semântica

O modo genérico, embora efetivo (71% de cobertura), sacrifica compreensão semântica:

Limitações:

- Não compreende estrutura JSON (pode corresponder strings dentro de comentários ou valores)
- Não pode rastrear fluxo de dados entre nós (taint analysis)
- Vulnerável a falsos positivos se padrões textuais aparecem em contextos seguros

Vantagens:

- Simplicidade: regras são regex compreensíveis, não AST patterns complexos
- Efetividade empírica: 19 detecções vs. 0
- Desenvolvimento rápido: 1,2h vs. 3,75h
- Manutenibilidade: atualizar regex é mais simples que reescrever AST patterns

Conclusão: Para análise de segurança de workflows n8n com Semgrep, o modo genérico é *pragmaticamente superior* ao modo JSON nativo, apesar de teoricamente menos sofisticado. Este resultado contradiz a intuição comum de que "parsing estruturado sempre supera correspondência textual".

7.4.2 Validação da Hipótese de Pesquisa

A hipótese central deste trabalho afirmava que ferramentas SAST existentes, com customização, poderiam fornecer análise de segurança para n8n. Os resultados validam parcialmente esta hipótese:

- **Validado**: Semgrep *pode* detectar vulnerabilidades em workflows n8n (71% de cobertura em modo genérico)
- **Qualificado**: Requer adaptação significativa do modo de análise, não apenas desenvolvimento de regras
- **Limitado**: Cobertura de 71% é insuficiente para ambientes críticos (meta: $\geq$80%)

7.4.3 Contribuição Metodológica: AI-Assisted Rule Generation

A geração assistida por IA (Claude 3.5 Sonnet) demonstrou efetividade:

- **Aceleração**: 13x mais rápido que desenvolvimento manual (3,75h vs. estimativa de 49h)
- **Qualidade**: Regras sintaticamente corretas, bem documentadas
- **Iteração**: Feedback humano essencial (2-4 iterações por regra)
- **Limitação**: IA gerou regras JSON mode que não funcionaram, mas pivotou efetivamente para generic mode quando informada da limitação

Lição: IA acelera desenvolvimento mas não elimina necessidade de validação empírica. A falha completa do modo JSON só foi identificada através de testes reais, não análise teórica.

7.4.4 Implicações Práticas

Para equipes de segurança considerando análise SAST de workflows n8n:

Recomendação Imediata: Utilize Semgrep em modo genérico com regras desenvolvidas neste trabalho como baseline. A cobertura de 71% é superior a 0% (análise manual ad-hoc).

Integração CI/CD: O modo genérico é adequado para gates automatizados:

- Execução rápida ($\sim$2s por workflow)
- Baixo consumo de recursos (55MB memória)
- Zero falsos positivos observados neste estudo
- Bloqueio de SQL/Command Injection (classes mais críticas)

Limitações Conhecidas: Classes não cobertas (hardcoded passwords, SSRF, DoS) requerem:

1. Revisão manual complementar
2. Análise dinâmica (execução em sandbox)
3. Desenvolvimento de ferramenta especializada (proposta na Seção **??**)

8 Discussão

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

8.1 Alcance dos Objetivos

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

8.2 Limitações Identificadas

[Discutir honestamente as limitações do trabalho.]

8.2.1 Limitações Técnicas

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

8.2.2 Limitações de Segurança

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

8.3 Comparação com Estado-da-Arte

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

8.4 Trade-offs Segurança vs. Usabilidade

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

8.5 Implicações Práticas

[Analisar impactos práticos do trabalho.]

8.5.1 Para Usuários/Stakeholders

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

8.5.2 Para Políticas Públicas

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

9 Conclusões e Próximos Passos

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

9.1 Síntese das Evidências

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

9.2 Contribuições

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

9.3 Trabalhos Futuros

[Propor extensões e melhorias.]

9.3.1 Curto Prazo (3-6 meses)

[Melhorias incrementais possíveis em 3-6 meses.]

9.3.2 Médio Prazo (6-12 meses)

Desenvolvimento de Registro de Regras Comunitário para n8n

Inspirado pelo modelo de sucesso do registro público de regras Semgrep (https://semgrep.dev/r), que oferece conjuntos de regras específicos para dezenas de linguagens de programação e frameworks, propõe-se o desenvolvimento de um registro análogo específico para workflows n8n. Este registro centralizaria regras de detecção de vulnerabilidades desenvolvidas pela comunidade, categorizadas por classe de vulnerabilidade (OWASP LCNC Top 10) e validadas contra workflows de teste.

O repositório `n8n-CyberSecurity-Workflows` [1] demonstra interesse comunitário significativo em automação de segurança com n8n, sugerindo que uma iniciativa de regras compartilhadas teria adoção prática. A metodologia de geração assistida por IA desenvolvida neste trabalho poderia acelerar a criação inicial do registro, reduzindo a barreira de entrada para contribuições.

Componentes principais:

- Repositório Git com regras Semgrep em modo genérico validadas
- Documentação de cada regra com exemplos de código vulnerável e seguro
- Dataset de teste público com workflows contendo vulnerabilidades conhecidas
- Pipeline de CI/CD para validação automática de contribuições comunitárias
- Integração com gerenciador de pacotes (npm/pip) para distribuição fácil

9.3.3 Longo Prazo (1-2 anos)

Ferramenta SAST Dedicada para Workflows n8n

A longo prazo, propõe-se o desenvolvimento de uma ferramenta SAST especializada para o ecossistema n8n, inspirada no sucesso do Semgrep mas otimizada especificamente para a estrutura JSON declarativa de workflows. Enquanto o Semgrep em modo genérico demonstrou efetividade neste trabalho (71% de cobertura), uma ferramenta dedicada poderia oferecer capacidades superiores através de compreensão nativa do esquema n8n.

Diferencial competitivo vs. Semgrep genérico:

- **Parser nativo do esquema n8n**: Compreensão semântica de nós, parâmetros e conexões, permitindo análise de fluxo de dados (taint analysis) através do grafo de execução
- **Regras pré-configuradas**: Conjunto abrangente de regras para as seis classes de vulnerabilidades identificadas, mantido e atualizado pela comunidade
- **Integração CI/CD nativa**: Plugin para GitHub Actions, GitLab CI e Jenkins que bloqueia automaticamente merges de workflows vulneráveis, similar ao GitHub Advanced Security
- **Interface visual**: Dashboard web mostrando postura de segurança de workflows organizacionais, tendências temporais e priorização de remediações
- **Análise incremental**: Detecção de mudanças entre versões de workflows, reportando apenas novas vulnerabilidades introduzidas

Modelo de distribuição:

Seguindo o modelo bem-sucedido do Semgrep, propõe-se arquitetura híbrida open-source + SaaS:

- **Core open-source (MIT License)**: Motor de análise, regras comunitárias, CLI para execução local, mantendo transparência e permitindo auto-hospedagem para organizações com requisitos rigorosos de privacidade
- **SaaS opcional (freemium)**: Dashboard colaborativo, agregação histórica, gerenciamento centralizado de políticas, integração com plataformas n8n Cloud, cobrando apenas por funcionalidades de equipe

Esta dualidade garante acessibilidade (pesquisadores e pequenas organizações utilizam versão gratuita) enquanto cria modelo de sustentabilidade financeira que viabiliza desenvolvimento contínuo e suporte empresarial.

Integração no pipeline de desenvolvimento n8n:

A ferramenta seria integrada diretamente no fluxo de trabalho de desenvolvimento de workflows:

1. **Pre-commit hook**: Análise local antes de commit, alertando desenvolvedor imediatamente sobre vulnerabilidades introduzidas
2. **Pull Request checks**: Bloqueio automático de PRs contendo vulnerabilidades críticas, com comentários inline identificando padrões inseguros e sugerindo correções
3. **Deploy gate**: Validação final antes de implantação em produção, com políticas configuráveis por ambiente (desenvolvimento permite warnings, produção bloqueia errors)
4. **Monitoramento contínuo**: Escaneamento periódico de workflows existentes em produção, identificando vulnerabilidades introduzidas por mudanças na plataforma n8n ou descoberta de novos vetores de ataque

Sustentabilidade e governança:

Inspirado em projetos open-source bem-sucedidos como Semgrep e GitLab:

- Governança comunitária através de comitê técnico com representantes de organizações usuárias
- Roadmap público com priorização transparente de funcionalidades
- Processo formal de contribuição com revisão por pares e CI/CD rigoroso
- Documentação abrangente incluindo guias de contribuição e arquitetura técnica
- Financiamento híbrido: patrocínios corporativos + receita SaaS + grants de pesquisa

Esta visão de longo prazo posiciona a ferramenta como padrão de facto para análise de segurança de workflows n8n, similar ao papel que o Semgrep desempenha para código tradicional, elevando significativamente a maturidade de segurança do ecossistema LCNC.

9.4 Recomendações

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

10 Checklist de Conformidade (LGPD/Ética/Segurança)

10.1 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]*

10.2 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]*

10.3 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]*

10.4 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.]