

**DEVSECOPS INTELIGENTE:
INTEGRAÇÃO DE AGENTES
AUTÔNOMOS DE
INTELIGÊNCIA ARTIFICIAL
NA DETECÇÃO,
PRIORIZAÇÃO E CORREÇÃO
DE VULNERABILIDADES EM
PIPELINES DE INTEGRAÇÃO
E ENTREGA CONTÍNUAS**

**INTELLIGENT DEVSECOPS: INTEGRATING AUTONOMOUS ARTIFICIAL
INTELLIGENCE AGENTS INTO VULNERABILITY DETECTION,
PRIORITIZATION, AND REMEDIATION IN CONTINUOUS INTEGRATION AND
DELIVERY PIPELINES**

Ciências Exatas e da Terra, Engenharias • 30/10/2026

REGISTRO DOI: [10.70773/revistatopicos/793245533](https://doi.org/10.70773/revistatopicos/793245533)

Márcio Rodrigo Elias Carvalho¹

Karinne Rocha Queiroz Menezes²

Jailson da Silva dos Santos³

Samuel Santana de Almeida⁴

Natália José Pereira Rodrigues⁵

Frederico Augusto Siqueira Gentil⁶

Raimundo Araujo de Almeida Junior⁷

RESUMO

A intensificação das práticas de integração e entrega contínuas ampliou a velocidade de mudança dos sistemas e tornou mais complexa a gestão de vulnerabilidades na cadeia de desenvolvimento. A integração de agentes autônomos de inteligência artificial aos pipelines de CI/CD oferece possibilidades de contextualizar alertas, relacionar dependências, recomendar correções e automatizar etapas controladas de validação. Este artigo analisa criticamente essas possibilidades, seus pressupostos arquiteturais e seus limites de segurança. Trata-se de revisão sistematizada qualitativa com síntese narrativa de documentação técnica institucional, padrões e literatura sobre desenvolvimento seguro, segurança de pipelines, gestão de risco em inteligência artificial e sistemas agênticos. A análise distingue a capacidade de gerar sugestões de código da autorização para executar mudanças, propõe uma arquitetura de autonomia graduada e examina riscos de injeção de instruções, comprometimento da cadeia de suprimentos, exposição de credenciais, regressões e fragilidade da atribuição de responsabilidades. Conclui-se que a adoção responsável depende de políticas verificáveis, segregação de funções, revisão humana proporcional ao risco, execução isolada, testes reproduzíveis, trilhas de auditoria e indicadores de qualidade que não se limitem à velocidade de remediação. O modelo apresentado constitui proposta conceitual, cuja efetividade exige validação empírica em diferentes contextos organizacionais.

Palavras-chave: DevSecOps; Agentes Autônomos; Inteligência Artificial; CI/CD; Gestão de Vulnerabilidades; Segurança da Cadeia de Suprimentos.

ABSTRACT

The expansion of continuous integration and delivery has increased

software change velocity while making vulnerability management across the development supply chain more complex. Integrating autonomous artificial intelligence agents into CI/CD pipelines creates opportunities to contextualize findings, connect dependencies, recommend patches, and automate controlled verification activities. This paper critically examines these opportunities, their architectural assumptions, and their security boundaries. It adopts a qualitative systematized review with narrative synthesis of institutional technical guidance, standards, and literature concerning secure development, pipeline security, AI risk management, and agentic systems. The analysis differentiates the ability to propose code changes from the authority to execute them, proposes graduated levels of autonomy, and discusses prompt injection, supply-chain compromise, credential exposure, regressions, and accountability gaps. Responsible adoption requires enforceable policies, separation of duties, isolated execution, risk-proportionate human review, reproducible tests, audit trails, and quality indicators beyond remediation speed. The proposed architecture is conceptual and requires empirical validation across organizational contexts.

Keywords: DevSecOps; Autonomous Agents; Artificial Intelligence; CI/CD; Vulnerability Management; Software Supply Chain Security.

1. INTRODUÇÃO

A integração contínua e a entrega contínua alteraram a dinâmica operacional da engenharia de software. Alterações frequentes passam por compilação, testes, análise de dependências, criação de artefatos e implantação em ambientes heterogêneos. A aceleração desse ciclo não elimina o tempo necessário para compreender riscos de segurança, sobretudo quando diferentes ferramentas emitem

alertas incompletos, duplicados ou descontextualizados. O desafio não é simplesmente acrescentar mais verificações ao pipeline: trata-se de incorporar decisões de segurança suficientemente informadas e auditáveis sem transformar os mecanismos de proteção em gargalos arbitrários. O Secure Software Development Framework do NIST oferece uma referência para integrar práticas seguras ao ciclo de desenvolvimento, enquanto a OWASP chama atenção para os próprios pipelines como superfícies críticas de ataque (Souppaya; Scarfone; Dodson, 2022; OWASP, s.d.-a).

A problemática pode ser delimitada em torno de uma tensão entre velocidade de entrega, qualidade da evidência e autoridade operacional. Scanners produzem achados cuja severidade técnica nem sempre corresponde ao risco observado em um serviço concreto; um agente pode correlacionar tais achados com dependências, arquitetura e histórico de alterações, mas sua análise continua sujeita a erros de interpretação. Em ambientes com múltiplas equipes, a cadeia de decisão inclui também critérios de aceitação de risco, responsabilidades contratuais, prioridades de negócio e obrigações regulatórias. Por isso, a pesquisa não presume que automatizar uma atividade signifique melhorar sua segurança. A hipótese relevante é mais estreita: agentes podem aumentar a capacidade de produzir propostas contextualizadas e verificáveis quando operam sob fronteiras explícitas de acesso e avaliação. Esse recorte diferencia a inovação funcional de sua validação operacional. Uma alteração de código aparentemente adequada pode falhar em execução, criar um novo caminho de ataque ou depender de um pacote não confiável. A avaliação precisa acompanhar todo esse percurso, desde a descoberta até a implantação, e distinguir sugestões úteis de correções que efetivamente reduzem exposição (Souppaya; Scarfone; Dodson, 2022; Autio et al., 2024).

Também é preciso explicitar o significado de inteligência no DevSecOps. Ele não corresponde a transferir indiscriminadamente ao modelo a função de autoridade técnica, mas à possibilidade de combinar informação estruturada e não estruturada em apoio a decisões verificáveis. A arquitetura pode usar um agente para explicar uma regra de análise estática, recuperar o contexto da dependência, examinar um teste que falhou e construir uma proposta de correção. Nenhuma dessas operações exige que o modelo tenha acesso de escrita a repositórios protegidos ou a credenciais de publicação. Em especial, a decisão de aceitar risco deve continuar vinculada a uma pessoa ou instância organizacional identificável. Essa distinção torna o problema cientificamente testável: é possível observar separadamente qualidade de triagem, utilidade da explicação, aceitação de propostas, regressões e violações de política. Sem essa decomposição, indicadores globais de produtividade podem ocultar erros graves de autorização. O ganho de rapidez interessa apenas quando acompanhado de preservação ou aumento da confiabilidade do processo. A análise assim orientada evita tratar autonomia como um fim em si mesmo e a situa como variável de projeto que precisa ser justificada pela evidência.

A emergência de agentes de inteligência artificial capazes de utilizar ferramentas, planejar sequências de ações e produzir alterações de código cria uma nova categoria de automação. Diferentemente de um classificador de achados ou de um assistente que apenas sugere alterações, o agente pode consultar repositórios, interpretar resultados de scanners, acionar testes e propor solicitações de mudança. Essa amplitude exige uma distinção decisiva: autonomia cognitiva para formular hipóteses não equivale a privilégio operacional para modificar ativos, acessar segredos ou promover

uma versão à produção. A OWASP (2025a) identifica superfícies específicas desses sistemas, como ferramentas, identidade, memória e supervisão humana, que precisam ser consideradas como parte da arquitetura de segurança.

O problema norteador do estudo consiste em investigar como agentes autônomos de inteligência artificial podem ser integrados à detecção, priorização e correção de vulnerabilidades em pipelines CI/CD sem reduzir a verificabilidade dos controles de segurança. O objetivo geral é propor um modelo conceitual de integração governada, capaz de apoiar decisões técnicas e preservar a responsabilidade humana por intervenções críticas. Como objetivos específicos, analisam-se as etapas da gestão de vulnerabilidades, os mecanismos de contextualização do risco, a arquitetura de permissões e aprovações, as ameaças aos agentes e os indicadores apropriados para futura avaliação empírica. A contribuição consiste em sistematizar requisitos operacionais e limites de autonomia, e não em afirmar desempenho superior não demonstrado.

A lacuna de pesquisa reside na distância entre demonstrações isoladas de geração automática de correções e estudos que examinem um fluxo completo de entrega segura. Uma tarefa de benchmark pode considerar sucesso quando um patch satisfaz testes previamente definidos, enquanto um ambiente produtivo precisa lidar com dependências transitivas, políticas de aprovação, documentação de risco residual, mudanças concomitantes e possibilidade de reversão. Essas dimensões não são acessórios administrativos: elas determinam se um resultado de laboratório pode ser transportado para uma organização. Estudos empíricos sobre agentes em segurança de software têm reforçado que executar tarefas autênticas permanece difícil, mesmo quando os

sistemas apresentam resultados convincentes em avaliações mais simples (Lee et al., 2025). A proposta apresentada neste artigo procura preencher uma lacuna conceitual mediante a especificação de componentes, controles e critérios que possam orientar comparações futuras. Não se afirma aqui implementação concluída, ensaio controlado ou melhoria percentual observada. O valor científico da arquitetura depende, ao contrário, de tornar refutáveis seus pressupostos e de explicitar em quais condições a intervenção poderia ser considerada suficientemente segura.

2. METODOLOGIA

Esta pesquisa adota abordagem qualitativa de natureza exploratória e propositiva, operacionalizada como revisão sistematizada qualitativa com síntese narrativa. Foram examinadas fontes institucionais públicas efetivamente consultadas, destacando-se o NIST SP 800-218, o perfil de inteligência artificial generativa do NIST, a documentação da OWASP sobre riscos de CI/CD e aplicações agênticas e a especificação SLSA. A seleção foi orientada pela pertinência direta à pergunta de pesquisa, pela explicitação de responsabilidades de segurança e pela utilidade para construção de uma arquitetura de referência. Não houve busca exaustiva em todas as bases bibliográficas nem triagem auditável de estudos; portanto, o trabalho não é apresentado como revisão sistemática formal e não atribui quantidades fictícias a registros identificados, excluídos ou incluídos.

O corpus documental foi organizado a partir de publicações institucionais passíveis de identificação, com consulta direta aos documentos do NIST, da OWASP, da especificação SLSA e às páginas oficiais da FIRST e da CISA relacionadas à avaliação de

vulnerabilidades. Incorporou-se ainda o estudo SEC-bench, publicado em anais científicos, como contraponto empírico às orientações normativas. A seleção teve caráter intencional: buscou cobrir requisitos de desenvolvimento seguro, riscos de sistemas agênticos, integridade de artefatos, critérios de priorização e limitações de agentes na remediação. Não se reivindica cobertura exaustiva da literatura, tampouco a existência de uma busca protocolada em bases acadêmicas que não tenha ocorrido. Cada documento foi interpretado conforme sua natureza: normas e guias fornecem recomendações e categorias de controle; especificações técnicas descrevem propriedades e garantias; pesquisas experimentais informam desempenho apenas nos cenários avaliados. Essa separação impede que uma diretriz de segurança seja indevidamente apresentada como evidência causal de redução de incidentes. O procedimento de síntese consistiu em confrontar os documentos com as etapas propostas da arquitetura, registrando pontos de convergência, tensões e questões não resolvidas.

Como estratégia de apreciação crítica, as afirmações foram examinadas segundo três perguntas: qual fenômeno a fonte efetivamente documenta; qual nível de inferência ela permite; e que controle ou indicador seria necessário para testar a aplicação proposta. Por exemplo, a existência de um catálogo de exploração conhecida não prova que qualquer critério automatizado de priorização seja correto, mas demonstra a disponibilidade de evidências externas que podem complementar a severidade de uma vulnerabilidade. De forma semelhante, a existência de uma especificação de proveniência não comprova que todo artefato atestado seja seguro; seu papel é fortalecer a rastreabilidade do processo de construção. Essa matriz interpretativa foi utilizada para evitar uma síntese meramente descritiva de documentos. Persistem

limitações importantes, entre elas ausência de avaliação sistemática de risco de viés de todos os estudos, possibilidade de atualização das páginas consultadas e heterogeneidade entre plataformas. Os resultados devem ser lidos como análise técnica fundamentada e desenho conceitual passível de teste, não como avaliação de produto comercial ou como metanálise de eficácia.

A análise organizou o material em cinco eixos interdependentes: governança da cadeia de desenvolvimento; obtenção e contextualização de achados; priorização orientada a risco; remediação e testes sob controle; e segurança, supervisão e mensuração dos próprios agentes. Em cada eixo, diferenciaram-se disposições explicitamente descritas por normas e guias, inferências técnicas compatíveis com tais orientações e propostas originais para pesquisa futura. As recomendações foram avaliadas por sua coerência arquitetural e capacidade de serem testadas, não como prova de eficácia. A ausência de experimentos controlados, métricas observadas, comparações entre modelos e registros de implantação constitui limitação assumida desde o delineamento.

3. FUNDAMENTOS DE DevSecOps E SEGURANÇA EM CI/CD

O DevSecOps pretende distribuir a responsabilidade pela segurança ao longo das etapas de planejamento, codificação, construção, teste, implantação e operação, em vez de concentrá-la em uma revisão tardia. Na prática, essa integração combina análise estática de código, análise dinâmica quando aplicável, identificação de segredos, composição de software, verificações de infraestrutura como código e políticas de implantação. O desafio aumenta porque cada instrumento observa aspectos distintos do sistema e pode apresentar resultados sem contexto suficiente. O NIST (Souppaya;

Scarfone; Dodson, 2022) recomenda que práticas de desenvolvimento seguro sejam adaptadas ao processo específico, inclusive para prevenir recorrências pela análise das causas fundamentais das falhas.

A integração de segurança ao desenvolvimento exige governança sobre a origem dos requisitos e sobre a resolução dos achados. Regras de análise de código precisam ser versionadas e justificadas em função das ameaças relevantes, sob pena de gerar bloqueios arbitrários ou de silenciar defeitos por conveniência. Em um processo bem definido, a equipe estabelece um conjunto mínimo de verificações antes de a mudança ser proposta e distingue exceções temporárias de desvios aceitos com fundamentação. O agente pode apoiar a análise das exceções, reunindo evidências de impacto e sugerindo ações mitigatórias, mas não deveria alterar a política que torna a exceção necessária. O SSDF articula práticas de preparação organizacional, proteção do software, produção de software seguro e resposta a vulnerabilidades, oferecendo uma base mais abrangente que a simples adição de ferramentas ao pipeline (Souppaya; Scarfone; Dodson, 2022). Em organizações heterogêneas, isso implica mapear responsabilidades entre desenvolvedores, segurança, operações e responsáveis pelo risco de negócio. A distribuição de responsabilidades não equivale à ausência de responsáveis: cada decisão relevante deve possuir autor, justificativa, prazo para revisão e evidência de execução.

Um pipeline não é apenas condutor neutro das análises. Seus executores podem dispor de credenciais, permissões para publicar pacotes e capacidade de acessar ambientes sensíveis. A OWASP (s.d.-a) organiza riscos como controles inadequados de fluxo, falhas de identidade e acesso, abuso da cadeia de dependências, execução

envenenada, higiene insuficiente de segredos, validação precária da integridade de artefatos e ausência de visibilidade. Acrescentar um agente com acesso amplo a esse ambiente potencializa riscos já existentes. Por isso, a arquitetura de automação deve considerar o pipeline como um ativo a ser defendido, e não somente como o lugar em que as vulnerabilidades de aplicações são descobertas.

A superfície de ataque do CI/CD merece análise independente da segurança do software produzido. Arquivos de configuração, ações de terceiros, runners compartilhados e processos de cache podem atuar como vias de comprometimento antes que o aplicativo final seja construído. Por isso, uma vulnerabilidade em um pipeline pode ter impacto desproporcional: credenciais de assinatura, tokens de publicação e permissões de implantação podem converter um problema pontual em comprometimento da cadeia de distribuição. Os riscos catalogados pela OWASP abrangem controle de fluxo insuficiente, gestão inadequada de identidades, cadeias de dependência abusadas e ausência de controles de integridade de artefatos, entre outros (OWASP, s.d.-a; OWASP, s.d.-b). Um agente com capacidade de alterar arquivos de workflow deve ser tratado como participante privilegiado de uma infraestrutura sensível. Mesmo quando seu propósito declarado consiste em corrigir segurança, a execução deve ocorrer com acesso mínimo e sem possibilidade de mudar unilateralmente regras de aprovação, localização de segredos ou critérios de confiança dos executores.

A proteção da cadeia de suprimentos complementa a segurança de código. A especificação SLSA descreve progressões de garantias relacionadas à origem de componentes, processos de construção e atestações, permitindo que organizações estabeleçam objetivos mensuráveis de integridade. Um agente que recomenda atualização

de biblioteca precisa verificar mais do que a redução de um alerta: deve examinar procedência, compatibilidade, mudanças transitivas, manutenção do projeto e consistência do artefato construído. Uma atualização aparentemente corretiva pode introduzir dependência maliciosa ou quebrar garantias de reprodutibilidade. O princípio de defesa em profundidade requer conservar verificações independentes das saídas do modelo (SLSA, 2025).

A qualidade da informação sobre componentes é outro requisito da automação confiável. Um inventário de dependências e um software bill of materials podem ajudar a relacionar um alerta a versões efetivamente distribuídas, mas sua completude depende da etapa e da configuração em que foram gerados. Informações produzidas apenas sobre arquivos declarativos podem não capturar componentes incorporados ao binário final; por sua vez, uma enumeração detalhada sem vínculo com a versão implantada reduz seu valor de investigação. O NIST recomenda atenção à geração e ao consumo de SBOMs em formatos interoperáveis e à relação entre a lista de componentes e o processo real de construção (NIST, 2022). Agentes podem comparar mudanças de SBOM antes e depois de uma atualização, identificar dependências transitivas afetadas e solicitar documentação ausente. Entretanto, não devem inferir segurança a partir da mera presença de inventário. Proveniência, autenticidade, verificação de integridade e testes funcionais são controles complementares. Na SLSA 1.2, as trilhas relacionadas à fonte e à construção esclarecem propriedades distintas da cadeia de suprimentos; não constituem certificação geral de inexistência de vulnerabilidades (SLSA, 2025).

4. AGENTES AUTÔNOMOS: CAPACIDADES, LIMITES E MODELOS DE AUTONOMIA

No escopo deste artigo, um agente é um componente computacional que recebe objetivos delimitados, consulta fontes de contexto, seleciona ferramentas autorizadas e produz ações ou propostas sujeitas a políticas. Essa definição evita tratar qualquer chamada a um modelo de linguagem como autonomia efetiva. O ciclo pode incluir interpretação de relatórios, decomposição de tarefas, execução de consultas read-only, preparação de alterações em uma ramificação isolada e análise dos resultados de testes. Entretanto, modelos generativos podem produzir inferências incorretas, introduzir dependências frágeis, interpretar instruções maliciosas como ordens válidas ou apresentar confiança inadequada. O perfil de risco de IA generativa do NIST recomenda incorporar governança, medição e gerenciamento continuado desses riscos (Autio et al., 2024).

O desenho de agentes precisa especificar separadamente capacidade de compreender uma tarefa, permissão para consultar dados e autoridade para praticar atos. Esses três eixos nem sempre evoluem juntos. Um sistema pode ser suficientemente competente para identificar uma configuração insegura e, ainda assim, não receber a chave que permitiria corrigi-la diretamente. Pode também executar testes em ambiente descartável sem possuir qualquer privilégio de publicar alterações. Essa separação reduz a possibilidade de uma falha de interpretação se converter automaticamente em incidente operacional. Perfis de risco de IA generativa do NIST recomendam governança e avaliação contínua dos sistemas; o perfil de desenvolvimento seguro para modelos de IA acrescenta preocupações específicas sobre os artefatos e componentes dessa tecnologia (Autio et al., 2024; Booth et al., 2024). Aplicadas ao DevSecOps, essas orientações exigem que o mecanismo de autorização permaneça externo ao texto produzido

pelo modelo, com políticas testáveis e identidades de serviço adequadas. O agente não deve criar, modificar ou aprovar sozinho as regras que restringem suas ações.

Propõe-se a distinção entre quatro níveis funcionais de autonomia, entendidos como categorias analíticas e não como classificação de uma norma externa. No nível consultivo, o agente apenas explica alertas e sugere hipóteses. No nível propositivo, produz uma modificação isolada e solicita revisão. No nível de execução restrita, efetua correções predefinidas em ambientes descartáveis, com testes obrigatórios e abertura de solicitação de alteração. No nível de operação sensível, qualquer promoção para produção, mudança de políticas, concessão de privilégios ou acesso a segredos exige controles independentes e autorização explícita. O avanço de um nível para outro depende da criticidade do ativo e da evidência acumulada de confiabilidade, não de uma impressão subjetiva de maturidade.

A gradação de autonomia também deve ser tratada de forma contingente ao risco e não como escala rígida de maturidade organizacional. A mesma instituição pode permitir atualizações automáticas de documentação não sensível e exigir dupla revisão para alterações em autenticação ou autorização. A classificação operacional deve levar em conta criticidade dos ativos, reversibilidade da mudança, abrangência dos testes, acesso a dados confidenciais e possibilidade de escalonamento de privilégios. Um recurso especialmente importante é a separação entre planejar e executar: o plano pode ser reavaliado por um serviço determinístico antes que ferramentas sejam acionadas. Além de reduzir exposição, esse controle torna comparáveis propostas de diferentes modelos, porque cada uma passa pelas mesmas verificações. O

acompanhamento longitudinal deve registrar quantas propostas são rejeitadas por violação de política, quantas são aprovadas após revisão humana e quantas precisam de reversão. Sem essa observabilidade, seria fácil confundir uma taxa elevada de ações concluídas com governança eficiente.

A decomposição de tarefas entre agentes especializados pode reduzir a complexidade de cada etapa, mas não garante, por si, a redução de riscos. Um agente de classificação pode interpretar um alerta, outro estimar impacto no sistema, um terceiro sugerir correção e um quarto verificar a conformidade da mudança. Se compartilham contexto contaminado, privilégios excessivos ou uma memória não validada, a multiplicação de agentes também multiplica as superfícies de ataque. A OWASP (2025b) destaca ameaças envolvendo raciocínio, identidade, ferramentas e interações entre agentes. Assim, a especialização deve acompanhar contratos explícitos de entrada e saída, validação de evidências e barreiras entre zonas de confiança.

A colaboração entre agentes especializados introduz questões adicionais de consistência e confiança. Um componente encarregado de recuperar contexto pode transmitir informações incompletas a outro responsável pela correção; um resumo incorreto torna-se então premissa de uma mudança subsequente. Para evitar esse efeito em cadeia, as mensagens entre agentes devem preservar referências às evidências originais, escopo temporal, identificadores de arquivo e graus explícitos de incerteza. A camada de orquestração precisa validar tipos de entrada e saída, impedir comandos ocultos em campos textuais e exigir reconfirmação quando uma decisão depende de contexto sensível. Não é suficiente dividir um processo em papéis distintos se todos compartilham uma

credencial administrativa ou se os resultados de um são tratados como verdade incontroversa pelos demais. A OWASP chama atenção para superfícies de identidade, ferramentas e interações entre agentes que ampliam os riscos de confusão de autoridade (OWASP, 2025b). A especialização, nesse sentido, representa uma escolha de arquitetura cuja eficácia precisa ser medida e não uma garantia automática de segurança.

5. DETECÇÃO E CORRELAÇÃO CONTEXTUAL DE VULNERABILIDADES

A primeira camada da arquitetura recebe evidências de ferramentas tradicionais: achados de análise estática, dependências vulneráveis, segredos expostos, configurações inadequadas, contêineres e ativos em execução. Cada registro deve preservar origem, versão da ferramenta, compromisso de código, caminho, evidência reproduzível e momento da coleta. Um agente pode auxiliar a normalização sem substituir o scanner ou a prova técnica. O objetivo da correlação consiste em identificar possíveis duplicações, vínculos entre componentes e pré-condições de exploração. É fundamental manter os dados originais imutáveis, de forma que divergências entre a interpretação algorítmica e o achado primário possam ser reexaminadas posteriormente.

A integridade do fluxo de evidências é decisiva porque relatórios de análise estática, varredura de dependências e execução de testes têm estruturas e pressupostos diferentes. Um resultado de SAST pode indicar um fluxo de dados potencialmente perigoso, ao passo que uma falha encontrada por DAST depende de condições de execução observadas; um alerta em biblioteca pode estar relacionado a código que nem sequer é acionado no caminho

vulnerável. Normalizar registros não deve apagar essas distinções. O agente poderia estabelecer relações entre identificadores CVE, categorias CWE, versões afetadas e trechos do repositório, mas precisaria manter o vínculo com o achado original e com a ferramenta que o produziu. Uma recomendação rastreável explicita se a ligação foi verificada, inferida ou ainda não examinada. Essa taxonomia reduz o risco de decisões baseadas em uma linguagem confiante que esconde dados insuficientes. A reprodutibilidade, aqui, não significa necessariamente obter o mesmo texto probabilístico em cada execução; significa que a evidência técnica e as condições de aceitação da correção possam ser recuperadas e testadas independentemente.

A contextualização depende do conhecimento de arquitetura e da função do ativo: exposição à internet, alcance dos privilégios, dados tratados, controles compensatórios, criticidade de negócio e presença efetiva da dependência no caminho de execução. Uma biblioteca com vulnerabilidade conhecida pode representar risco distinto conforme a função vulnerável seja alcançável ou a aplicação se encontre isolada; contudo, ausência de prova de exploração não é prova de inexistência de risco. O agente deve registrar quais informações foram verificadas, quais são hipóteses e quais permanecem indisponíveis. Quando o contexto é incompleto, a recomendação responsável consiste em elevar a necessidade de investigação, não em atribuir artificialmente baixa severidade.

A noção de explorabilidade contextual exige prudência metodológica. A análise de alcançabilidade de uma função vulnerável, por exemplo, pode oferecer evidências para modular prioridade, mas depende da cobertura do programa analisado, de mecanismos dinâmicos e de características do ambiente de

execução. Se esses elementos não forem observados, a conclusão adequada é de incerteza e não de imunidade. O mesmo raciocínio vale para serviços expostos apenas em redes internas, pois credenciais comprometidas, cadeias de ataque e configurações transitórias podem modificar o cenário. Na elaboração de um caso de risco, o agente deveria explicitar premissas de exposição, impacto esperado, dependências e medidas compensatórias, permitindo auditoria por um analista. A decisão de interromper uma implantação pode exigir evidências diferentes daquelas necessárias para abrir um ticket de investigação. Essa diferença de consequências recomenda políticas de aprovação proporcionais, com critérios predefinidos de urgência, possibilidade de revisão e registro da decisão responsável.

Uma boa detecção assistida por IA precisa resistir à contaminação de fontes. Repositórios, arquivos README, relatórios de scanner e comentários de código podem conter texto arbitrário, inclusive instruções adversariais destinadas ao modelo. O sistema deve tratá-los como dados não confiáveis, estabelecer precedência das políticas internas e impedir que conteúdo lido em um repositório conceda novas permissões. A triagem também deve ser reexecutável: um analista autorizado precisa recuperar as entradas, reproduzir as verificações determinísticas e justificar eventuais divergências. Essa separação entre evidência técnica e interpretação probabilística sustenta a auditabilidade, especialmente quando um erro de classificação poderia permitir a publicação de uma versão insegura (OWASP, 2025a).

A proteção contra manipulação de informações exige separar os canais que recebem conteúdo de terceiros daqueles que autorizam operações. Em particular, comentários inseridos em uma solicitação

de alteração, trechos de documentação recuperados automaticamente e mensagens de execução de testes são entradas que podem conter instruções adversariais. Essas instruções precisam ser tratadas como dados referentes à tarefa, não como comandos legítimos para o modelo executar. O controle técnico inclui esquemas fechados para chamadas de ferramentas, validação de parâmetros, bloqueio de destinos de rede não autorizados e recusa de ações fora do escopo da tarefa. A atribuição de proveniência ao conteúdo recuperado também é importante: sem identificar repositório, versão e origem, um resumo previamente gerado pode adquirir autoridade indevida em uma execução futura. A OWASP situa a injeção de instruções e o abuso de ferramentas entre as ameaças que exigem modelagem própria em aplicações agênticas (OWASP, 2025a). A solução, portanto, não pode depender apenas de uma instrução textual pedindo ao agente que ignore conteúdo malicioso.

6. PRIORIZAÇÃO ORIENTADA A RISCO E DECISÃO EXPLICÁVEL

O volume de alertas pode induzir equipes a ordenar vulnerabilidades apenas por severidade padronizada ou pela data de abertura. Embora esses elementos sejam úteis, a priorização operacional exige considerar probabilidade plausível de exploração, superfície exposta, relevância para o negócio, facilidade de mitigação e efeitos de uma mudança corretiva. Um agente pode consolidar essas dimensões, apresentar cenários e recomendar filas de tratamento; entretanto, a decisão final de aceitar risco, adiar uma correção ou interromper uma implantação deve obedecer à política organizacional. Recomenda-se registrar explicitamente pesos, exceções, fontes e responsáveis, evitando escores opacos cujo significado não possa ser questionado por equipes técnicas.

A priorização deve diferenciar severidade intrínseca da vulnerabilidade, probabilidade de exploração, exposição efetiva do ativo e consequência de um incidente. O Common Vulnerability Scoring System, na versão 4.0, organiza métricas básicas, de ameaça, ambientais e suplementares, favorecendo descrições comparáveis de características técnicas; esse escore, entretanto, não substitui o julgamento específico de cada organização (FIRST, 2023). O Exploit Prediction Scoring System oferece estimativas populacionais de probabilidade de exploração em um horizonte de trinta dias, mas seus resultados não representam diagnóstico da configuração local nem medida direta de impacto (FIRST, s.d.). O catálogo Known Exploited Vulnerabilities, por sua vez, identifica vulnerabilidades com evidências de exploração conhecida e fornece um sinal relevante de priorização (CISA, s.d.). Um agente pode reunir esses insumos e indicar convergências e divergências entre eles. A política deve impedir, contudo, que uma pontuação externa seja convertida automaticamente em ordem para corrigir ou ignorar determinada falha. O contexto institucional e as evidências de implementação permanecem indispensáveis.

Em vez de produzir uma pontuação única com aparência excessiva de precisão, a arquitetura pode organizar situações em classes justificadas: intervenção imediata, remediação planejada, investigação contextual e aceitação formal com prazo de revisão. Uma recomendação deve esclarecer o motivo de determinada vulnerabilidade ter prioridade, por exemplo, por combinar exposição externa, permissões amplas e ausência de contenção; também deve permitir reclassificação diante de novos fatos. As escolhas precisam ser contestáveis e versionadas, pois tanto o cenário operacional quanto as informações públicas sobre ameaças se modificam. A IA

auxilia a ordenar trabalho humano, mas não elimina incerteza nem transforma critérios de risco em verdades matemáticas.

Ao construir uma fila de tratamento, convém separar situações nas quais há informação suficiente para uma ação daquelas em que ainda é preciso investigar. Uma classificação artificialmente exata transmite impressão indevida de certeza quando a versão implantada, a superfície de exposição ou a existência de controles compensatórios não foram confirmadas. Um agente responsável pode, nesses casos, sugerir experimentos ou consultas adicionais, como recuperar o inventário do ambiente, reproduzir a condição vulnerável ou obter documentação de mudança. A urgência operacional também deve ser sensível à dependência temporal entre tarefas: um patch que exige atualização de esquema de dados pode demandar sequência diferente daquela de uma substituição isolada de biblioteca. Por isso, a justificativa de risco deveria informar não só o que corrigir, mas o que precisa ocorrer antes de a correção ser segura. A utilidade organizacional da análise decorre da capacidade de apoiar decisões contestáveis, e não da produção de rótulos invariavelmente corretos.

Há ainda um conflito estrutural entre rapidez e estabilidade. Corrigir imediatamente uma dependência crítica pode diminuir uma exposição e introduzir regressão em serviço essencial. Adiar indefinidamente essa atualização também pode ser inadequado. A decisão precisa confrontar ambas as possibilidades, prever testes direcionados, janelas de implantação e mecanismos de reversão. Um agente que só maximize a quantidade de alertas encerrados pode estimular alterações cosméticas, supressões indevidas ou movimentação de risco para outros componentes. Portanto, a governança exige indicadores de recorrência, falhas pós-correção e

qualidade de justificativa, além dos prazos de atendimento (Souppaya; Scarfone; Dodson, 2022).

Há uma dimensão de governança frequentemente negligenciada na priorização automatizada: o custo de oportunidades perdidas. Se o sistema induz a equipe a tratar primeiro um grande volume de achados de baixo impacto porque são facilmente corrigíveis, vulnerabilidades difíceis e relevantes podem permanecer expostas. Avaliações deveriam, portanto, observar o conjunto de exposições abertas ao longo do tempo, a criticidade dos ativos associados e a distribuição do esforço de engenharia. Convém ainda distinguir o encerramento administrativo de um alerta da remoção verificável de sua causa. Um modelo que propõe desabilitar uma regra de scanner para elevar indicadores de produtividade falhou, mesmo que o número de alertas tenha diminuído. Políticas de mensuração precisam vincular a remediação a evidências independentes e examinar recorrências. Essa perspectiva está alinhada à preocupação do SSDF com causas fundamentais e permite formular resultados empíricos que não premiem apenas volume de atividades executadas (Souppaya; Scarfone; Dodson, 2022).

7. CORREÇÃO AUTOMATIZADA, VALIDAÇÃO E PROMOÇÃO DE ALTERAÇÕES

A remediação assistida deve ser construída como processo reversível e com fronteiras claras. Depois da triagem, o agente prepara uma proposta em ramificação temporária, limita arquivos que pode editar, descreve a vulnerabilidade e explicita a hipótese de correção. Em seguida, executa testes estáticos, testes de unidade, integração e segurança pertinentes, sem obter permissões desnecessárias. A solicitação de alteração deve apresentar diferença de código,

resultado das verificações, dependências modificadas e riscos residuais. Um patch que elimina o alerta do scanner não basta como prova de segurança: é necessário avaliar a causa fundamental, o comportamento preservado e a compatibilidade com as políticas do repositório.

Um fluxo robusto de correção começa com a definição verificável do comportamento indesejado e de uma condição de aceitação. Para vulnerabilidades reproduzíveis, pode-se registrar teste que falha antes da mudança e passa depois dela, preservando ainda testes de regressão do comportamento esperado. Quando não é seguro ou viável reproduzir a exploração, a avaliação precisa explicitar os limites da evidência e recorrer a verificações alternativas. A alteração proposta pelo agente deve ser suficientemente pequena para permitir revisão, mas não tão superficial que apenas neutralize um sintoma detectado pelo scanner. Uma dependência vulnerável, por exemplo, pode requerer substituição de API, adaptação de configurações e novos testes para manter propriedades de segurança. O processo de aprovação deveria verificar a correspondência entre a causa da falha, o patch e os critérios de aceitação, em vez de simplesmente confirmar que o pipeline concluiu sem alertas. Um resultado positivo em testes existentes não demonstra ausência universal de vulnerabilidades, sobretudo quando a cobertura do teste é limitada.

Correções simples e determinísticas podem ser candidatas à execução controlada, como atualização de configuração para valor já aprovado ou substituição de uma versão segundo política explícita. Alterações em autenticação, criptografia, autorização, persistência de dados, políticas de rede ou dependências centrais demandam revisão humana reforçada. Esse desenho não reduz o humano a um

clique simbólico: a aprovação precisa fornecer evidências compreensíveis e tempo suficiente para contestação. Para impedir autoaprovação, a identidade que produz o patch não deve ser a única capaz de modificar proteções de branch, executar o processo de build confiável e autorizar a publicação.

A revisão humana efetiva exige que o artefato produzido seja compreensível. Isso envolve apresentar o trecho modificado, a hipótese que justifica a mudança, as alternativas rejeitadas, a evidência de testes executados e as possíveis consequências sobre interfaces e dados. A presença formal de um aprovador não melhora a segurança quando o volume de propostas impede exame substantivo. Por isso, indicadores de carga de revisão e de retrabalho devem acompanhar os de tempo de correção. Para mudanças críticas, uma organização pode exigir múltiplas aprovações segregadas ou execução de controles independentes de especialistas, sem depender da classificação emitida pelo próprio agente. A rastreabilidade da decisão também é central para incidentes posteriores: deve ser possível distinguir erro de sugestão, erro de implementação, falha de teste e autorização inadequada. Essa estrutura transforma o processo em cadeia auditável de responsabilidades e reduz a opacidade introduzida pela geração probabilística de código.

O mecanismo de promoção para homologação e produção deve permanecer separado da etapa generativa. Artefatos precisam ser construídos em executores confiáveis, identificados por versões ou resumos criptográficos e acompanhados de evidências de proveniência, quando aplicável. A especificação SLSA fornece referências relevantes para fortalecer garantias de construção e atestação, enquanto a OWASP ressalta o perigo de validação

insuficiente da integridade de artefatos. Dessa forma, mesmo que um agente gere uma alteração tecnicamente plausível, o produto publicado deve passar por controles que não dependam da própria afirmação do agente acerca de sua correção (OWASP, s.d.-a; SLSA, 2025).

A promoção de artefatos deve estar vinculada à identidade exata do código, das dependências e do ambiente de construção. Mesmo uma correção correta na ramificação analisada pode tornar-se insegura se o artefato distribuído resultar de outro estado do repositório ou incorporar etapas de construção não verificadas. Assinaturas, atestações e verificações de proveniência ajudam a reduzir essa distância, desde que sejam efetivamente validadas pelo consumidor do artefato. A SLSA 1.2 separa garantias sobre fonte e construção, permitindo pensar em controles específicos para autorar, revisar, produzir e distribuir software (SLSA, 2025). Um agente que produziu o patch não deve ser a única fonte de declaração da integridade do build; a atestação deve provir de mecanismos confiáveis e sujeitos a políticas de verificação. O mesmo princípio se aplica à reversão: um rollback precisa recuperar uma versão conhecida e controlada, sem reproduzir o componente vulnerável em situação que possa ampliar risco. A arquitetura de implantação, portanto, constitui parte integrante da decisão de remediar.

8. AMEAÇAS ESPECÍFICAS E CONTROLES DE DEFESA

Uma ameaça particularmente relevante é a injeção de instruções. Um invasor que consiga alterar documentação, metadados ou arquivos analisados pelo agente pode tentar induzi-lo a executar comandos não autorizados, revelar dados ou ignorar verificações. A

defesa requer tratamento explícito de conteúdo externo como informação não confiável, listas de ferramentas autorizadas, esquemas rígidos de chamadas e validação de ações antes da execução. Isolamento em contêineres ou ambientes efêmeros ajuda a restringir efeitos, mas não substitui revisão das capacidades concedidas. A OWASP (2025a; 2025b) ressalta que riscos agênticos surgem precisamente na passagem da interpretação de dados para a tomada de ações.

A modelagem de ameaças deve especificar o que um atacante controla. Um invasor pode modificar conteúdo de uma biblioteca externa, abrir uma solicitação de alteração, manipular documentação ou contaminar um registro de análise posteriormente recuperado pelo agente. Também pode tentar explorar permissões de ferramentas concedidas à instância de execução ou induzir decisões com informações factualmente falsas. O impacto potencial varia conforme o agente esteja restrito à leitura, possa modificar uma ramificação ou disponha de privilégios para publicar código. Um plano de avaliação de segurança precisa cobrir essas capacidades separadamente. Não basta medir a taxa de sucesso em tarefas benignas: é necessário construir cenários em que o comportamento correto seja negar uma chamada, limitar um acesso, solicitar revisão ou interromper a execução. Testes adversariais devem inspecionar vazamento de dados, manipulação do repositório, contorno de gates de aprovação e degradação da qualidade dos registros. Esse enfoque se alinha às superfícies identificadas pela OWASP para aplicações agênticas (OWASP, 2025a; OWASP, 2025b).

Credenciais de curta duração, identidade específica por tarefa, permissões mínimas e segmentação entre ambientes reduzem

oportunidades de escalonamento. O agente de revisão de código não deveria acessar chaves de implantação; o agente responsável por testes não deveria reconfigurar os critérios de aprovação. Também é necessário controlar o envio de código e dados para serviços externos, considerando confidencialidade, contratos, retenção e jurisdição. Logs devem permitir reconstituir quem iniciou uma tarefa, qual versão do agente operou, quais ferramentas foram acionadas e que artefatos resultaram; simultaneamente, esses registros precisam evitar armazenar segredos e dados pessoais sem necessidade.

Privilégios mínimos precisam ser materializados em controles de infraestrutura e não somente em declarações de política. Credenciais efêmeras e vinculadas à tarefa, segregação de ambientes, isolamento de processos e acesso restrito a recursos de rede reduzem a possibilidade de escalonamento. Também é aconselhável separar a identidade que solicita uma execução daquela que a autoriza e registrar as credenciais utilizadas em cada etapa sem expor seus valores. O modelo pode justificar a necessidade de uma ferramenta, mas o serviço que executa a chamada deve verificar parâmetros e escopo. Quando uma tarefa demanda informação confidencial, políticas devem definir limites de retenção, envio externo e acesso por fornecedores. Essas medidas não dependem de o agente ser mais ou menos persuasivo: constituem controles tradicionais cuja importância cresce à medida que sistemas generativos passam a operar de forma ativa. O NIST destaca a natureza transversal da governança de riscos de IA, incluindo documentação, medição e gerenciamento continuado (Autio et al., 2024).

A persistência de memória de agentes constitui outro vetor de risco. Resumos gerados anteriormente podem carregar conclusões incorretas ou conteúdo contaminado que reaparece em execuções futuras. Recomenda-se vincular memória e recuperação documental ao repositório, à versão e ao nível de confiança, com expiração e possibilidade de correção. Ataques à cadeia de dependências também precisam ser contemplados: a recomendação de instalar um pacote deve passar por política de procedência e inspeções independentes, mesmo quando o nome do componente pareça legítimo. A arquitetura defensiva combina controles clássicos de desenvolvimento seguro com mecanismos específicos para modelos e ferramentas (Autio et al., 2024; OWASP, 2025b).

A segurança da memória e da recuperação de documentos precisa ser avaliada como questão de integridade de dados. Um resumo antigo pode continuar circulando depois que uma dependência foi removida, ou um arquivo não confiável pode inserir uma instrução persistente em uma base de conhecimento. A arquitetura deve permitir invalidar material quando a versão do repositório muda, diferenciar fontes aprovadas de conteúdo fornecido por terceiros e revalidar evidências críticas antes de ações sensíveis. Outro risco é o de dependência excessiva de respostas do próprio agente para fiscalizar seu comportamento. Verificações de política, resultados de build, análise de integridade e registros de acesso devem ser controlados por componentes independentes. Essa independência não elimina falhas de configuração, mas reduz a probabilidade de que uma única interpretação incorreta comprometa análise, aprovação e auditoria ao mesmo tempo. Para pesquisa aplicada, tais requisitos podem ser transformados em testes de injeção,

envenenamento de memória e resistência à alteração indevida de permissões.

9. ARQUITETURA CONCEITUAL DE REFERÊNCIA

O modelo proposto distribui responsabilidades entre uma camada de coleta de evidências, um serviço de normalização e correlação, agentes especializados, um motor independente de políticas e um mecanismo de execução restrita. A coleta recebe eventos e relatórios produzidos por scanners e pelo pipeline. A normalização mantém identificadores de ocorrência, proveniência e contexto. Os agentes produzem análises ou patches, mas suas solicitações de ação atravessam um ponto obrigatório de aplicação de políticas que confere identidade, escopo, criticidade do ativo e autorização vigente. Essa separação impede que a saída textual do modelo seja interpretada automaticamente como ordem privilegiada. Os resultados ficam registrados em trilha auditável, associada a uma versão de código e a testes reproduzíveis.

A arquitetura conceitual deve incluir uma camada de identidade e autorização que antecede qualquer execução com efeitos persistentes. A tarefa recebida precisa carregar identificador do solicitante, escopo de repositórios, prazo de validade, recursos autorizados e política aplicável. O orquestrador pode organizar consultas e propostas, mas não alterar a identidade em nome da qual atua. As saídas do agente, por sua vez, devem ser representadas em estruturas passíveis de validação, distinguindo hipóteses, evidências e ações solicitadas. Esse desenho dificulta que conteúdo livre produzido pelo modelo seja interpretado como autorização. Um motor independente de políticas verifica se a ação pertence à lista permitida e se existem condições necessárias, como aprovação

registrada, testes concluídos e estado de repositório correspondente. A trilha resultante liga a decisão humana às entradas utilizadas, aos testes e ao artefato final. A proposta constitui especificação para implementação futura e não descrição de infraestrutura já implantada.

O processo de decisão inicia com identificação do achado, passa pela análise de contexto e chega a uma recomendação de prioridade acompanhada de justificativa. Havendo possibilidade de remediação, o agente trabalha em cópia isolada e abre proposta de alteração. A verificação combina testes determinísticos, inspeção de qualidade, revisão dos impactos e comparação de evidências anteriores e posteriores. Em casos previamente autorizados e de risco limitado, uma política pode permitir determinadas tarefas automáticas; em casos críticos, a aprovação permanece humana e segregada. A função do motor de políticas é estabelecer limites objetivos que continuem válidos mesmo quando um modelo produzir uma recomendação persuasiva, mas incorreta.

A avaliação de uma solicitação de mudança pode ser concebida como sequência de garantias independentes. Primeiro, deve-se preservar a identificação do achado e sua relação com o estado do código. Depois, exige-se uma explicação do risco contextual e uma hipótese verificável de correção. Em seguida, a proposta passa por checagens de política, testes pertinentes, inspeção do diff e confirmação de sua proveniência. Somente então mecanismos autorizados podem produzir um artefato para homologação, deixando a promoção condicionada ao nível de risco. Essa ordem importa porque a execução antecipada de ferramentas com privilégios elevados pode transformar um erro de interpretação em evento irreversível. Para facilitar investigações, as decisões não

devem ser sobrescritas por resumos mais recentes do agente: as evidências devem permanecer localizáveis pelo identificador da execução. Um registro útil expõe não apenas o resultado aprovado, mas também exceções, falhas de testes e motivos de recusa.

Uma implantação piloto deveria começar por repositórios sem acesso a produção e por classes bem caracterizadas de vulnerabilidades. Antes de ampliar escopo, seria necessário documentar riscos aceitos, definir responsáveis, testar a resistência a instruções maliciosas e avaliar falhas de reversão. O ambiente de teste precisa conter casos negativos, nos quais o comportamento correto seja recusar uma ação, solicitar evidência adicional ou encaminhar a decisão. A arquitetura não pressupõe um fornecedor, modelo de linguagem ou plataforma CI/CD particular; essa neutralidade busca facilitar estudos comparativos, embora a implementação concreta dependa das condições técnicas e de governança de cada instituição.

Uma implementação gradual permite testar hipóteses de confiabilidade antes de ampliar o domínio de atuação do agente. Inicialmente, podem ser selecionados repositórios não produtivos e classes de mudanças reversíveis, mantendo a possibilidade de comparar recomendações do sistema com avaliações independentes de especialistas. A expansão deve depender de indicadores previamente definidos, como preservação de comportamento funcional, ausência de violações críticas de política e documentação reprodutível de decisões. Quando surgirem falhas, a equipe precisa distinguir problemas do modelo, da integração com ferramentas, dos dados de contexto ou das políticas que deveriam restringir a execução. Uma arquitetura neutra em relação a fornecedores facilita tal análise porque reduz o risco de tratar

características de um produto específico como leis gerais de sistemas agênticos. O estudo pode, assim, orientar desenho experimental com hipóteses falsificáveis e critérios de interrupção claros.

10. RESULTADOS DA SÍNTESE E DISCUSSÃO

A síntese documental revela convergência entre diretrizes de desenvolvimento seguro e recomendações de segurança para sistemas agênticos: a automação deve produzir evidências, respeitar limites de acesso e permanecer sujeita a governança. O NIST destaca integração das práticas de segurança ao ciclo de desenvolvimento e tratamento das causas de vulnerabilidades; a OWASP identifica fragilidades estruturais de CI/CD e riscos específicos de ações conduzidas por agentes; a SLSA oferece um vocabulário de garantias de procedência e integridade. Essas referências não demonstram experimentalmente que agentes reduzem incidentes em pipelines, mas sustentam critérios de arquitetura para que a hipótese seja avaliada de modo mais controlado (Souppaya; Scarfone; Dodson, 2022; OWASP, s.d.-a; OWASP, 2025a; SLSA, 2025).

A literatura experimental recente introduz um contraponto útil às expectativas de automação ampla. No trabalho SEC-bench, Lee et al. (2025) propõem uma estrutura de avaliação com tarefas de segurança de software em repositórios reais, incluindo reprodução de vulnerabilidades e geração de patches verificáveis. Os resultados relatados pelos autores revelam dificuldades importantes para agentes avaliados nas condições do benchmark, o que desaconselha inferir capacidade de remediação geral a partir de exemplos demonstrativos bem-sucedidos. O estudo é relevante precisamente por buscar tarefas mais próximas de situações autênticas de

engenharia, mas seus resultados não podem ser transportados automaticamente para qualquer agente, repositório ou período. Tampouco representam medição do modelo conceitual aqui desenvolvido. A interpretação adequada é que o desempenho de agentes constitui questão empírica em aberto, sujeita a propriedades da linguagem, qualidade dos testes, complexidade da vulnerabilidade e condições de execução. Essa evidência reforça a opção metodológica por separar proposta de arquitetura, requisitos de controle e demonstração experimental de eficácia.

A principal vantagem potencial está em conectar informações que frequentemente permanecem distribuídas entre ferramentas e equipes: alertas, contexto de execução, histórico de alterações, dependências e justificativas de risco. Essa função pode diminuir esforços manuais de contextualização e acelerar a produção de propostas de correção, mas também pode introduzir novos erros por inferência e automação. A discussão não permite afirmar ganho líquido sem mensuração de falsos positivos, falsos negativos, regressões, tempo de revisão, consumo de recursos e ocorrências de violação de políticas. A produtividade aparente de um sistema agêntico será insuficiente se o aumento de velocidade vier acompanhado de menor confiabilidade ou de perda de rastreabilidade.

Para avaliar ganho real de produtividade, os estudos precisam contabilizar trabalho transferido entre etapas. Um agente pode reduzir o tempo de triagem e aumentar a carga de revisão de patches inadequados; também pode gerar explicações úteis sem melhorar a taxa final de correções aprovadas. A comparação deveria incluir períodos de preparação, infraestrutura consumida, atividades humanas exigidas e ocorrências de regressão após a implantação.

Convém medir separadamente a qualidade da priorização, a correção funcional do código e a conformidade com políticas de segurança. Um indicador agregado que mistura essas dimensões dificulta identificar onde o agente contribui ou prejudica. Da mesma forma, economia de tempo em tarefas repetitivas não equivale a menor risco de exploração se vulnerabilidades críticas continuarem abertas. A observação longitudinal deve relacionar recursos investidos ao conjunto de exposições efetivamente reduzidas, reconhecendo limites de atribuição causal em ambientes de mudanças contínuas.

Para pesquisa empírica futura, sugere-se comparação de pipelines equivalentes em tarefas definidas, com e sem assistência de agentes, controlando características dos projetos e dos achados. Indicadores pertinentes incluem tempo até triagem, tempo até correção aprovada, proporção de propostas aceitas após revisão, taxa de regressões, recorrência de vulnerabilidades, qualidade das justificativas e tentativas bloqueadas de ação indevida. Também é necessário acompanhar custo computacional, impacto na carga de trabalho de revisores e confiança calibrada das equipes. Os indicadores devem ser calculados com dados observados e protocolo transparente, sem pressupor resultados favoráveis. Estudos que avaliem apenas a quantidade de patches produzidos correm o risco de confundir atividade automatizada com redução efetiva de exposição.

A validade das comparações depende da construção cuidadosa do conjunto de tarefas. Vulnerabilidades sintáticas simples, falhas de configuração, problemas de autenticação e alterações arquiteturais mobilizam capacidades muito diferentes; uma média única pode ocultar desempenho inadequado justamente nos casos de maior

risco. Recomenda-se estratificar as amostras por linguagem, classe de vulnerabilidade, complexidade do patch e qualidade dos testes, com avaliações independentes e registro prévio dos critérios de sucesso. Testes negativos também devem fazer parte do protocolo: a capacidade de não executar uma ação perigosa ou de declarar insuficiência de evidência é componente relevante da qualidade de um agente. A participação de especialistas humanos não pode ser tratada como ruído a ser eliminado, pois sua intervenção integra o modelo operacional proposto. A avaliação deveria, portanto, investigar tanto o desempenho isolado dos agentes quanto sua contribuição quando inseridos em processos reais de decisão.

A comparação entre documentos ainda evidencia tensões não resolvidas. Uma política de mínima permissão protege ativos, mas pode limitar a capacidade de coletar contexto; uma revisão obrigatória aumenta controle, porém reduz velocidade; memórias persistentes podem favorecer continuidade de tarefas e simultaneamente elevar risco de contaminação. Não existe regra universal para solucionar essas tensões. O grau de autonomia aceitável depende da criticidade do sistema, capacidade de observação, maturidade dos testes e consequências de um erro. A contribuição da proposta reside em tornar essas escolhas explícitas e verificáveis, substituindo a promessa genérica de autocorreção por condições técnicas de autorização e responsabilização.

Há limites de generalização decorrentes da rápida evolução das ferramentas e dos próprios repositórios avaliados. O resultado de um sistema em determinada versão de linguagem, modelo ou scanner não define comportamento estável diante de alterações de contexto. Para isso, avaliações precisam conservar identificadores de versão, configuração de execução, fontes disponíveis e políticas em vigor,

permitindo distinguir mudanças de desempenho por atualização do modelo daquelas produzidas por alterações do ambiente. A qualidade de explicações também exige apreciação independente: uma justificativa plausível pode citar evidências incorretas e estimular confiança indevida. A triangulação entre teste determinístico, revisão humana e rastreamento do resultado é mais robusta que qualquer uma dessas fontes isoladamente. Nesse sentido, a contribuição central da arquitetura consiste em sustentar a contestabilidade das decisões e a possibilidade de investigar falhas, características necessárias à incorporação responsável da automação em sistemas críticos.

11. CONSIDERAÇÕES FINAIS

A integração de agentes autônomos de inteligência artificial ao DevSecOps deve ser compreendida como evolução das possibilidades de apoio ao desenvolvimento seguro, e não como substituição automática das políticas, ferramentas e responsabilidades já estabelecidas. O estudo respondeu à questão norteadora mediante a formulação de uma arquitetura de autonomia graduada, na qual detecção, contextualização, priorização e proposta de correção são articuladas por mecanismos independentes de autorização, testes e auditoria. A distinção entre recomendar e executar constitui seu eixo normativo: uma sugestão tecnicamente sofisticada ainda requer confirmação de evidências, delimitação de privilégios e avaliação de consequências. Assim, a qualidade de uma solução não pode ser inferida apenas da fluência da explicação produzida pelo agente ou da quantidade de alertas encerrados no pipeline.

A implicação principal para a pesquisa em segurança de software é que um agente deve ser avaliado como parte de um sistema sociotécnico e não apenas como mecanismo de geração textual. Seu comportamento depende de dados de contexto, regras organizacionais, privilégios efetivos, instrumentos de verificação e padrões de supervisão humana. O desenho proposto torna visível essa dependência ao instituir separação entre interpretação, recomendação, execução e promoção. Em consequência, a autonomia deve permanecer graduada conforme a criticidade dos ativos e a reversibilidade das alterações. Processos que toleram pequenas falhas funcionais podem aceitar maior automação de mudanças rotineiras, mas serviços com informações sensíveis ou ampla exposição necessitam controle reforçado e evidência independente. A maturidade do DevSecOps inteligente não pode ser reduzida à adoção de um modelo mais recente; ela requer capacidade organizacional de medir riscos, revisar políticas e reconstruir a história de cada decisão.

A revisão permite reconhecer que as práticas consolidadas de segurança da cadeia de desenvolvimento continuam pertinentes em ambientes agênticos. Identidades de serviço restritas, integridade de artefatos, controle de dependências, proteção de segredos, políticas de branch e testes reproduzíveis formam uma base que a inteligência artificial não dispensa. Ao contrário, quanto maior a capacidade de um componente de acionar ferramentas, maior a necessidade de impedir que informações não confiáveis sejam convertidas em comandos privilegiados. A segurança do agente é, portanto, inseparável da segurança do pipeline no qual atua. Essa perspectiva evita arquiteturas em que a automação recebe permissões administrativas sob a justificativa de facilitar correções, criando uma nova superfície de comprometimento

potencialmente mais sensível que as vulnerabilidades inicialmente investigadas.

A priorização orientada a risco foi identificada como campo promissor de apoio analítico, especialmente quando diferentes fontes de evidência precisam ser comparadas. Contudo, não se recomenda que um escore gerado por modelo funcione como decisão inquestionável. Exposição, impacto de negócio, características de exploração, limitações de teste e risco de regressão precisam ser explicitados de modo que especialistas possam contestar premissas e registrar justificativas. A transparência é particularmente importante para a aceitação de riscos e para adiamentos de remediação, pois essas decisões produzem consequências que extrapolam a camada de software. O modelo proposto preserva, por esse motivo, a revisão humana proporcional à criticidade, sem confundi-la com uma etapa meramente burocrática.

A articulação entre severidade, explorabilidade e contexto também oferece orientação para implementação. Fontes como CVSS, EPSS e KEV registram dimensões distintas do risco e precisam ser usadas de forma complementar, preservando incertezas e diferenças de finalidade. O agente pode ajudar o analista a localizar uma dependência relevante, verificar a versão implantada e relacionar sinais externos à exposição do serviço. Entretanto, aceitar ou postergar o tratamento exige critérios explícitos, responsáveis definidos e possibilidade de reavaliação. A explicabilidade, assim, não consiste apenas em produzir justificativas detalhadas, mas em permitir que outra pessoa confira premissas, fontes e consequências. A ausência de dados adequados deve resultar em investigação ou classificação provisória, e nunca em suposta certeza de inexistência

de ameaça. Essa disciplina previne a substituição de decisões técnicas contestáveis por uma pontuação de aparência objetiva e fundamentação frágil.

Outra conclusão refere-se à necessidade de incorporar resistência a ataques e comportamento seguro de recusa como objetivos de projeto. Agentes podem entrar em contato com comentários, arquivos e relatórios controlados por terceiros, e nem todo conteúdo aparentemente técnico deve orientar suas ações. Injeções de instrução, envenenamento de memória, manipulação de dependências e uso indevido de ferramentas precisam ser contemplados em avaliações antes da implantação. Uma abordagem madura deve demonstrar não apenas a capacidade de reparar um problema, mas também a capacidade de interromper atividades quando a evidência é insuficiente, a tarefa excede privilégios autorizados ou a integridade do ambiente é questionável. O reconhecimento explícito da incerteza constitui requisito de confiabilidade, e não deficiência do sistema.

As limitações metodológicas condicionam o alcance das conclusões. A pesquisa realizou síntese qualitativa de documentação institucional e literatura técnica selecionada; não efetuou comparação experimental entre agentes, não operou pipelines produtivos e não mediu redução de vulnerabilidades ou tempo de correção. Consequentemente, a arquitetura apresentada deve ser entendida como proposição testável, não como solução cuja eficácia foi demonstrada. Estudos subsequentes poderão investigar diferentes classes de falhas, modelos, linguagens, plataformas e políticas de autonomia mediante protocolos reproduzíveis. Especial atenção deverá ser dedicada às diferenças entre correções sintáticas

simples e mudanças arquiteturais de segurança, para que avaliações agregadas não ocultem níveis de risco muito distintos.

A agenda de pesquisa futura precisa incluir validação externa e mensuração de efeitos adversos. Um piloto conduzido em um único repositório, com tarefa conhecida e equipe experiente, não garante que os mesmos resultados ocorram em ambientes com dependências extensas, dados regulados ou pipelines legados. Estudos multicenários devem observar não somente produtividade, mas a evolução do estoque de vulnerabilidades relevantes, erros de triagem, revisões rejeitadas, incidentes e tentativas de manipulação do agente. A comparação também deve considerar custos de infraestrutura, mecanismos de proteção, esforço de revisão e possibilidade de reversão. A publicação de resultados negativos é especialmente importante, pois sistemas que falham em certas classes de problemas podem ainda ser úteis em outras, desde que suas limitações sejam conhecidas. Sem esses parâmetros, discursos de transformação do desenvolvimento seguro permanecem pouco verificáveis.

Conclui-se que o DevSecOps inteligente tem maior potencial quando integra a capacidade de análise contextual dos agentes à disciplina de engenharia de segurança, preservando uma cadeia contínua de evidências, autorização e responsabilização. O sucesso não estará caracterizado apenas por entregar código mais rapidamente, mas por aumentar a capacidade organizacional de identificar exposições relevantes, decidir com justificativas, produzir correções confiáveis e reconhecer limites de autonomia. A agenda futura deverá investigar empiricamente como alcançar esse equilíbrio e em quais circunstâncias os ganhos de eficiência compensam os custos e riscos adicionais. Até que existam

resultados comparativos robustos, recomenda-se implantação gradual, observável e reversível, guiada por políticas de acesso estritas e verificação independente.

Do ponto de vista ético e institucional, a responsabilidade pelo tratamento de vulnerabilidades permanece com agentes organizacionais identificáveis, mesmo quando a análise preliminar é automatizada. Sistemas agênticos não devem operar como instâncias de aceitação de risco sem controle independente; tampouco convém utilizar saídas probabilísticas como justificativa suficiente para alterar regras de autorização, suprimir logs ou ignorar testes. A retenção de informações sensíveis e o compartilhamento de código com fornecedores exigem políticas próprias, compatíveis com o contexto da instituição. O princípio orientador é proporcionalidade: ampliar autonomia somente quando exista evidência robusta de que os benefícios não transferem riscos ocultos para outras partes do ciclo de desenvolvimento. A adoção responsável combina salvaguardas de engenharia, acompanhamento de desempenho e formação das equipes para interpretar os limites dos agentes. Essas condições oferecem base para evolução incremental sem converter expectativas tecnológicas em garantias cientificamente não demonstradas.

REFERÊNCIAS BIBLIOGRÁFICAS

AUTIO, Chloe et al. Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. Gaithersburg: National Institute of Standards and Technology, 2024. NIST AI 600-1. DOI: <https://doi.org/10.6028/NIST.AI.600-1>.

BOOTH, Harold et al. Secure Software Development Practices for Generative AI and Dual-Use Foundation Models: An SSDF Community Profile. Gaithersburg: National Institute of Standards and Technology, 2024. NIST SP 800-218A. DOI: <https://doi.org/10.6028/NIST.SP.800-218A>.

CYBERSECURITY AND INFRASTRUCTURE SECURITY AGENCY (CISA). Known Exploited Vulnerabilities Catalog. Washington, DC: CISA, s.d. Disponível em: <https://www.cisa.gov/known-exploited-vulnerabilities-catalog>. Acesso em: 22 out. 2026.

FORUM OF INCIDENT RESPONSE AND SECURITY TEAMS (FIRST). Common Vulnerability Scoring System Version 4.0: User Guide. [S. l.]: FIRST, 2023. Disponível em: <https://www.first.org/cvss/v4.0/user-guide>. Acesso em: 22 out. 2026.

FORUM OF INCIDENT RESPONSE AND SECURITY TEAMS (FIRST). Exploit Prediction Scoring System (EPSS): Frequently Asked Questions. [S. l.]: FIRST, s.d. Disponível em: <https://www.first.org/epss/faq>. Acesso em: 22 out. 2026.

LEE, Hwiwon et al. SEC-bench: Automated Benchmarking of LLM Agents on Real-World Software Security Tasks. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 38., 2025. Proceedings [...]. [S. l.]: NeurIPS, 2025. DOI: <https://doi.org/10.52202/085713-3878>.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST). Software Security in Supply Chains: Software Bill of Materials (SBOM). Gaithersburg: NIST, 2022. Disponível em: <https://www.nist.gov/itl/executive-order-14028-improving-nations->

[cybersecurity/software-supply-chain-security-guidance-20](#). Acesso em: 22 out. 2026.

OPEN WORLDWIDE APPLICATION SECURITY PROJECT (OWASP). Agentic AI: Threats and Mitigations. [S. l.]: OWASP, 2025a. Disponível em: <https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/>. Acesso em: 22 out. 2026.

OPEN WORLDWIDE APPLICATION SECURITY PROJECT (OWASP). CI/CD Security Cheat Sheet. [S. l.]: OWASP, s.d.-b. Disponível em: https://cheatsheetseries.owasp.org/cheatsheets/CI_CD_Security_Cheat_Sheet.html. Acesso em: 22 out. 2026.

OPEN WORLDWIDE APPLICATION SECURITY PROJECT (OWASP). OWASP Gen AI Security Project: Agentic Threats Navigator. [S. l.]: OWASP, 2025b. Disponível em: <https://genai.owasp.org/resource/owasp-gen-ai-security-project-agentic-threats-navigator/>. Acesso em: 22 out. 2026.

OPEN WORLDWIDE APPLICATION SECURITY PROJECT (OWASP). OWASP Top 10 CI/CD Security Risks. [S. l.]: OWASP, s.d.-a. Disponível em: <https://owasp.org/projects/top-10-cicd-security-risks>. Acesso em: 22 out. 2026.

SOFTWARE SUPPLY CHAIN LEVELS FOR SOFTWARE ARTIFACTS (SLSA). SLSA Specification: Version 1.2. [S. l.]: SLSA, 2025. Disponível em: <https://slsa.dev/spec/v1.2/>. Acesso em: 22 out. 2026.

SOUPPAYA, Murugiah; SCARFONE, Karen; DODSON, Donna. Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities. Gaithersburg: National Institute of Standards and

¹ Especialização em Cybersecurity: Detecção de Ameaças e Proteção de Sistemas de Informática e Comunicações – FANS; Especialização em Redes de Computadores – FANESE; Especialização em Inteligência Artificial – FANS; Graduação em Sistemas de Informação – UNIT; Graduação em Administração – ETEP. E-mail: [acesse o artigo original para visualizar o e-mail](#). Lattes: <http://lattes.cnpq.br/8752905296064861>.

² Mestrado em Ciências da Educação – INTERAMERICANA; Especialização em Redes de Computadores – FANESE; Especialização em Metodologia e Didática de Nível Superior – SERIGY; Graduação em Gestão de Tecnologia da Informação – FANESE. E-mail: [acesse o artigo original para visualizar o e-mail](#). Lattes: <http://lattes.cnpq.br/0606923050760749>.

³ Especialização em Engenharia de Software – FOCUS; Especialização em Gestão de Riscos e Cibersegurança – FOCUS; Especialização em Tecnologia da Informação – UNIFAVENI; Graduação em Pedagogia – FSLF. E-mail: [acesse o artigo original para visualizar o e-mail](#). Lattes: <http://lattes.cnpq.br/8302147764638511>.

⁴ Mestrado em Ciência da Computação – UFS; Especialização em Big Data, Business Intelligence e Inteligência Artificial – FOCUS; Especialização em Matemática para o Mundo do Trabalho – UFPI; Pós-graduação em Docência para Educação Profissional e Tecnológica – IFPB; Licenciatura em Matemática – ESTÁCIO;

Bacharelado em Administração – ESTÁCIO; Tecnólogo em Gestão de Tecnologia da Informação – ESTÁCIO. E-mail: [acesse o artigo original para visualizar o e-mail](#). Lattes: <http://lattes.cnpq.br/7780092391740205>. ORCID: <https://orcid.org/0000-0001-8662-1673>.

⁵ Especialização em Linguagens, suas Tecnologias e o Mundo do Trabalho – UFPI; Especialização em Metodologia do Ensino da Língua Inglesa – FAVENI; Especialização em MIT em Engenharia de Software (.NET/C#) – INFNET; Segunda Licenciatura em Informática – UNIASSELVI; Graduação em Letras – Inglês – ESTÁCIO; Graduação em Análise e Desenvolvimento de Sistemas – UNICARIOCA. E-mail: [acesse o artigo original para visualizar o e-mail](#). Lattes: <https://lattes.cnpq.br/9898138931949263>. ORCID: <https://orcid.org/0009-0005-3263-7879>.

⁶ Especialização em Administração Hospitalar – Centro Universitário São Camilo; Especialização em Sistema de Informação para WEB – FANESE; Graduação em Administração com Ênfase em Análise de Sistemas – UNIT; Formação Pedagógica para Formadores de Educação Profissional – UNISUL. E-mail: [acesse o artigo original para visualizar o e-mail](#). Lattes: <https://lattes.cnpq.br/6247498924148402>. ORCID: <https://orcid.org/0000-0001-6676-575X>.

⁷ Mestrado em Sistemas e Computação – UNIFACS; MBA em Gestão de Tecnologia – FTE; Graduação em Sistemas de Informação – UCSAL. E-mail: [acesse o artigo original para visualizar o e-mail](#). Lattes: <http://lattes.cnpq.br/2972683702153025>.