

Intelligent DevSecOps Automation Framework Based on AI-Driven Security Analytics for Agent-Based Jenkins Pipelines

Risko^{1*)}, Agung Budi Susanto²⁾, Arya Adhyaksa Waskita³⁾

¹⁾²⁾³⁾Program Studi Magister Teknik Informatika, Program Pascasarjana, Universitas Pamulang

^{*)}Correspondence author: risko.alexander@gmail.com, Tangerang Selatan, Indonesia

DOI: <https://doi.org/10.37012/jtik.v12i2.3615>

Abstract

The adoption of Continuous Integration/Continuous Delivery (CI/CD) has accelerated the implementation of DevSecOps as an approach for integrating security throughout the software development lifecycle. Although various automated security tools are widely available, their outputs remain fragmented, require extensive manual analysis, and provide limited support for security-driven decision-making within CI/CD pipelines. This gap creates the objective of this study, which is to design and evaluate an Intelligent DevSecOps Automation Framework that combines AI-Driven Security Analytics with an Agent-Based Jenkins Pipeline to integrate multiple security tools, normalize heterogeneous findings, enforce a unified security quality gate, and generate contextual security advisories through Large Language Models (LLMs). The study employed an applied research method with a prototype development and experimental approach. The proposed framework integrates Ruff, Semgrep, SonarQube, and Trivy, and is supported by a finding normalization and consolidation mechanism, risk-based prioritization, security quality gate evaluation, and AI-driven advisory generation orchestrated through LangChain. The framework was evaluated using a deliberately vulnerable payment-service repository to examine its capability in detecting vulnerabilities, consolidating cross-tool findings, classifying risk, and producing mitigation recommendations together with a release decision. The results show that the framework successfully unified four security tools within a single pipeline, producing 160 raw findings that were normalized and consolidated into 37 unique findings, of which 27 were categorized as Critical or Major. Fifteen of the highest-priority findings were forwarded to the AI-Driven Security Analytics component, yielding an Overall Risk and Exploitation Risk of Critical, 14 items classified as release blockers, and an overall release recommendation of Hold. These findings indicate that the proposed framework effectively supports early vulnerability detection, cross-tool consolidation, risk-based prioritization, security quality gate enforcement, and AI-assisted security advisory generation to strengthen security decision-making in modern DevSecOps pipelines.

Keywords: DevSecOps, CI/CD Pipeline, Artificial Intelligence, Security Automation, Jenkins Pipeline

Abstrak

Penerapan Continuous Integration/Continuous Delivery (CI/CD) mendorong berkembangnya DevSecOps sebagai pendekatan untuk mengintegrasikan keamanan ke dalam seluruh siklus pengembangan perangkat lunak. Meskipun berbagai security tools telah tersedia untuk mengotomasi pengujian keamanan, implementasinya masih menghasilkan temuan yang terfragmentasi, membutuhkan analisis manual, dan belum mampu mendukung pengambilan keputusan keamanan pada pipeline CI/CD secara terintegrasi. Berdasarkan kesenjangan tersebut, penelitian ini bertujuan merancang dan mengevaluasi Intelligent DevSecOps Automation Framework yang menggabungkan AI-Driven Security Analytics dengan Agent-Based Jenkins Pipeline untuk mengintegrasikan berbagai security tools, menormalisasi temuan yang heterogen, menerapkan security quality gate yang seragam, serta menghasilkan security advisory kontekstual melalui Large Language Model (LLM). Penelitian menggunakan metode penelitian terapan dengan pendekatan pengembangan prototipe dan eksperimen. Framework yang dikembangkan mengintegrasikan Ruff, Semgrep, SonarQube, dan Trivy, dilengkapi mekanisme normalisasi dan konsolidasi temuan, prioritas berbasis risiko, evaluasi security quality gate, serta pembangkitan security advisory berbasis AI yang diorkestrasi menggunakan LangChain. Framework dievaluasi menggunakan repositori uji payment-service yang secara sengaja mengandung berbagai kerentanan untuk menguji kemampuan mendeteksi kerentanan, mengkonsolidasikan temuan lintas tools, mengklasifikasikan risiko, serta menghasilkan rekomendasi mitigasi dan keputusan rilis. Hasil penelitian menunjukkan bahwa framework berhasil menyatukan empat security tools dalam satu pipeline dan

menghasilkan 160 raw findings yang dinormalisasi serta dikonsolidasikan menjadi 37 unique findings, dengan 27 di antaranya berkategori Critical atau Major. Sebanyak 15 temuan berprioritas tertinggi diteruskan ke komponen AI-Driven Security Analytics, menghasilkan Overall Risk dan Exploitation Risk pada kategori Critical, 14 temuan diklasifikasikan sebagai release blocker, dan rekomendasi rilis keseluruhan berupa Hold. Hasil tersebut menunjukkan bahwa framework yang diusulkan mampu mendukung deteksi dini kerentanan, konsolidasi temuan lintas alat, prioritas berbasis risiko, penerapan security quality gate, serta penyusunan security advisory berbantuan AI untuk memperkuat pengambilan keputusan keamanan pada pipeline DevSecOps modern.

Kata Kunci: DevSecOps, CI/CD Pipeline, Kecerdasan Buatan, Otomasi Keamanan, Jenkins Pipeline

PENDAHULUAN

Perkembangan perangkat lunak modern ditandai oleh meluasnya penerapan Continuous Integration dan Continuous Delivery/Deployment (CI/CD) yang memungkinkan organisasi mempercepat proses integrasi, pengujian, dan distribusi perangkat lunak secara otomatis. Didukung oleh pendekatan cloud-native dan praktik DevOps, siklus rilis yang sebelumnya berlangsung dalam hitungan minggu atau bulan kini dapat dijalankan beberapa kali dalam sehari. Kondisi ini meningkatkan kecepatan inovasi dan responsivitas terhadap kebutuhan bisnis, tetapi juga meningkatkan kompleksitas pengelolaan keamanan aplikasi. Proses keamanan yang masih dilakukan secara manual atau ditempatkan pada tahap akhir pengembangan tidak lagi mampu mengimbangi laju perubahan perangkat lunak, sehingga kerentanan berpotensi lolos hingga ke lingkungan produksi. Insiden keamanan berskala besar, seperti serangan rantai pasok SolarWinds dan eksploitasi kerentanan Log4j, menunjukkan bahwa satu kelemahan pada rantai pasok perangkat lunak dapat berdampak luas terhadap banyak organisasi, sehingga keamanan perlu menjadi bagian yang terintegrasi dalam seluruh siklus pengembangan perangkat lunak.

Sebagai respons terhadap tantangan tersebut, DevSecOps berkembang sebagai evolusi DevOps yang mengintegrasikan keamanan ke dalam setiap tahapan Software Development Life Cycle (SDLC) melalui prinsip shift-left security dan security as code. Melalui pendekatan ini, aktivitas keamanan tidak lagi diperlakukan sebagai proses terpisah setelah pengembangan selesai, melainkan di otomasi dan dijalankan secara berkelanjutan pada pipeline CI/CD, umumnya memanfaatkan berbagai security tools untuk analisis kualitas kode, deteksi kerentanan aplikasi, pemeriksaan software dependency, dan pemindaian container image. Integrasi berbagai mekanisme pengujian tersebut diharapkan meningkatkan kemampuan deteksi dini terhadap kerentanan sekaligus menekan biaya perbaikan pada tahap akhir pengembangan. Meskipun demikian, penelitian terdahulu

menunjukkan bahwa implementasi DevSecOps masih menghadapi tantangan pada aspek integrasi alat keamanan, interoperabilitas, otomasi proses, serta penyajian hasil analisis yang mudah dipahami oleh pengembang (Zhao et al., 2024; Prates & Pereira, 2025).

Pada praktiknya, organisasi umumnya menggunakan lebih dari satu security tool untuk memperoleh cakupan deteksi kerentanan yang lebih komprehensif. Alat seperti SonarQube, Semgrep, Trivy, dan linter tools memiliki kemampuan yang saling melengkapi dalam mendeteksi berbagai jenis kelemahan keamanan, tetapi setiap alat menghasilkan format keluaran, klasifikasi tingkat keparahan, dan karakteristik temuan yang berbeda, sehingga proses analisis keamanan masih dilakukan secara parsial (siloed) (Nocera et al., 2025). Kondisi ini menyebabkan hasil pemindaian menjadi terfragmentasi, munculnya temuan yang saling tumpang tindih (duplicate findings), serta meningkatnya kebutuhan analisis manual untuk menentukan prioritas mitigasi dan keputusan rilis perangkat lunak. Akibatnya, tujuan utama DevSecOps untuk menghadirkan otomasi keamanan secara menyeluruh belum sepenuhnya tercapai. Permasalahan tersebut menegaskan kesenjangan penelitian (research gap), yaitu belum tersedianya integrasi multi-security tools dalam satu pipeline DevSecOps yang terpadu, sekaligus belum optimalnya proses analisis hasil pemindaian secara otomatis.

Perkembangan Artificial Intelligence (AI), khususnya Large Language Model (LLM), membuka peluang baru untuk meningkatkan efektivitas analisis keamanan pada pipeline DevSecOps. Kemampuan LLM dalam memahami konteks, menganalisis informasi dari berbagai sumber, serta menghasilkan rekomendasi berbasis bahasa alami memungkinkan AI dimanfaatkan untuk mendukung vulnerability triage, reduksi false positive, dan penyusunan rekomendasi mitigasi yang lebih kontekstual (Bedoya et al., 2024; Camacho, 2024). Pemanfaatan framework orkestrasi seperti LangChain memungkinkan layanan AI diintegrasikan ke dalam pipeline DevSecOps sehingga hasil pemindaian keamanan tidak hanya disajikan sebagai laporan teknis, tetapi juga diterjemahkan menjadi security advisory yang lebih mudah dipahami oleh pengembang. Meskipun demikian, pemanfaatan AI pada penelitian maupun praktik DevSecOps saat ini masih cenderung parsial dan belum terintegrasi dengan mekanisme konsolidasi temuan keamanan dari berbagai security tools dalam satu alur kerja yang terpadu (Binbeshr & Imam, 2025). Tinjauan literatur sistematis oleh Fu, Pasuksmit, dan Tantithamthavorn (2025) turut menegaskan bahwa penerapan AI

pada tahapan DevOps masih terfragmentasi menurut fase SDLC dan belum banyak dievaluasi secara end-to-end pada satu pipeline yang utuh.

Beberapa penelitian sebelumnya telah membahas integrasi AI dalam DevSecOps maupun otomasi keamanan pada pipeline CI/CD. Kokku (2024) mengkaji pemanfaatan AI dan machine learning untuk otomasi pengujian keamanan pada DevOps, sedangkan Malik dan Sachs (2025) mengusulkan integrasi threat intelligence untuk memitigasi risiko sebelum kode mencapai lingkungan produksi. Kiran Kumar Reddy Yelkoti (2025) mengembangkan kerangka arsitektur security as code untuk mitigasi risiko otomatis pada pipeline DevSecOps. Kağızmandere dan Arslan (2024) mengusulkan model pemindaian kerentanan otomatis berbasis Software Bill of Materials (SBOM) untuk pipeline CI/CD, sedangkan Aljohani dan Alqahtani (2023) mengembangkan kerangka kerja terpadu untuk otomasi analisis keamanan perangkat lunak pada DevSecOps. Namun, penelitian-penelitian tersebut umumnya berfokus pada satu aspek tertentu, seperti otomasi pengujian, analisis intelijen ancaman, atau pemindaian dependensi, dan belum menyediakan mekanisme terpadu yang menggabungkan konsolidasi temuan lintas-tools dengan analisis keamanan berbasis LLM secara end-to-end pada satu pipeline. Kesenjangan inilah yang menjadi novelty penelitian ini, yaitu integrasi multi-security tools, normalisasi dan konsolidasi temuan, security quality gate, serta AI-Driven Security Analytics dalam satu Agent-Based Jenkins Pipeline yang terpadu.

Berdasarkan uraian tersebut, penelitian ini mengajukan rumusan masalah: bagaimana rancangan Intelligent DevSecOps Automation Framework berbasis Agent-Based Jenkins Pipeline dapat mengintegrasikan berbagai security tools, mengkonsolidasikan hasil pemindaian lintas alat, serta memanfaatkan AI untuk menghasilkan rekomendasi mitigasi dan keputusan rilis secara otomatis. Penelitian ini bertujuan merancang, mengimplementasikan, dan mengevaluasi framework tersebut dengan mengintegrasikan Ruff, Semgrep, SonarQube, dan Trivy sebagai mekanisme deteksi kerentanan, memanfaatkan LangChain sebagai orkestrator AI-Driven Security Advisory, serta menerapkan Large Language Model (LLM) untuk analisis hasil pemindaian dan rekomendasi mitigasi. Kontribusi penelitian ini terletak pada integrasi end-to-end antara multi-security scanning, konsolidasi temuan, security quality gate, dan AI-assisted security analytics dalam satu alur CI/CD, sehingga dapat mendukung otomasi keamanan,

mempercepat penyampaian security feedback, dan membantu pengambilan keputusan keamanan pada pipeline DevSecOps.

METODE PENELITIAN

Penelitian ini menggunakan metode eksperimen (experimental method) untuk merancang, mengimplementasikan, dan mengevaluasi Intelligent DevSecOps Automation Framework pada lingkungan CI/CD berbasis Agent-Based Jenkins Pipeline. Eksperimen dilakukan melalui implementasi framework pada lingkungan pengujian yang mengintegrasikan empat alat analisis keamanan, yaitu Ruff, Semgrep, SonarQube, dan Trivy, dengan mekanisme analisis berbasis Large Language Model (LLM) yang diorkestrasi menggunakan LangChain. Metode eksperimen digunakan untuk mengamati kemampuan framework dalam menjalankan proses keamanan secara terintegrasi dan otomatis, mulai dari eksekusi multi-security tools, pengolahan hasil pemindaian, analisis keamanan berbasis AI, hingga pembentukan security advisory dan rekomendasi keamanan.

Desain Eksperimen

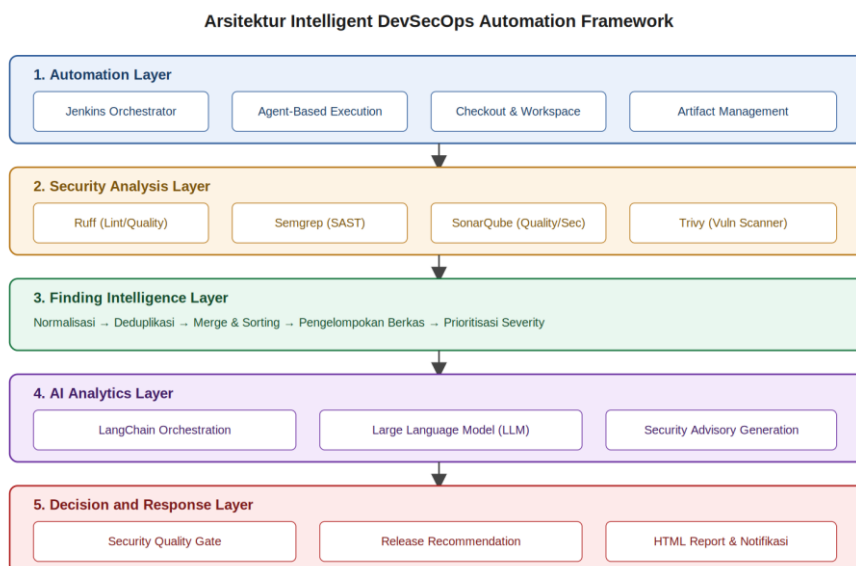
Desain eksperimen disusun untuk mengevaluasi implementasi Intelligent DevSecOps Automation Framework dalam mengotomasi proses analisis keamanan pada pipeline CI/CD. Objek eksperimen berupa aplikasi uji yang diproses melalui pipeline Jenkins dan dianalisis menggunakan beberapa alat keamanan dengan fungsi yang saling melengkapi. Setiap eksekusi pipeline menghasilkan data keamanan yang selanjutnya diproses untuk membentuk informasi risiko dan security advisory. Eksperimen dilaksanakan melalui tahapan persiapan kode sumber dan lingkungan pengujian, eksekusi pipeline, pelaksanaan analisis keamanan menggunakan multi-security tools, pengumpulan hasil pemindaian, pemrosesan temuan keamanan, analisis berbasis AI, serta evaluasi terhadap keluaran framework. Pendekatan ini memungkinkan keseluruhan alur framework diamati secara end-to-end, mulai dari perubahan kode sumber hingga dihasilkannya informasi yang mendukung proses mitigasi dan keputusan keamanan.

Aspek yang diamati dalam eksperimen meliputi kemampuan integrasi alat keamanan dalam satu pipeline, hasil deteksi yang dihasilkan setiap alat, proses pengolahan temuan keamanan, keluaran analisis berbasis AI, rekomendasi mitigasi, status evaluasi keamanan, dan

rekomendasi rilis. Data eksperimen diperoleh dari keluaran pipeline, laporan masing-masing alat keamanan, hasil analisis AI, serta artefak laporan yang dihasilkan selama proses pengujian.

Arsitektur Intelligent DevSecOps Automation Framework

Penelitian ini mengusulkan Intelligent DevSecOps Automation Framework sebagai arsitektur terintegrasi untuk mengotomasi proses analisis keamanan pada pipeline CI/CD. Framework menghubungkan proses orkestrasi pipeline, analisis keamanan menggunakan beberapa alat, pengolahan hasil pemindaian, analisis berbasis AI, serta mekanisme pengambilan keputusan dan distribusi hasil dalam satu alur kerja. Arsitektur framework tersusun atas lima lapisan fungsional. Automation Layer menempatkan Jenkins sebagai orkestrator utama yang mengendalikan eksekusi pipeline pada agent, mengelola proses pengambilan source code, persiapan environment, eksekusi pengujian dan analisis keamanan, serta pengelolaan artefak hasil pipeline. Pendekatan berbasis agent memungkinkan proses eksekusi dijalankan pada lingkungan yang telah dikonfigurasi sesuai kebutuhan setiap tahapan pipeline.



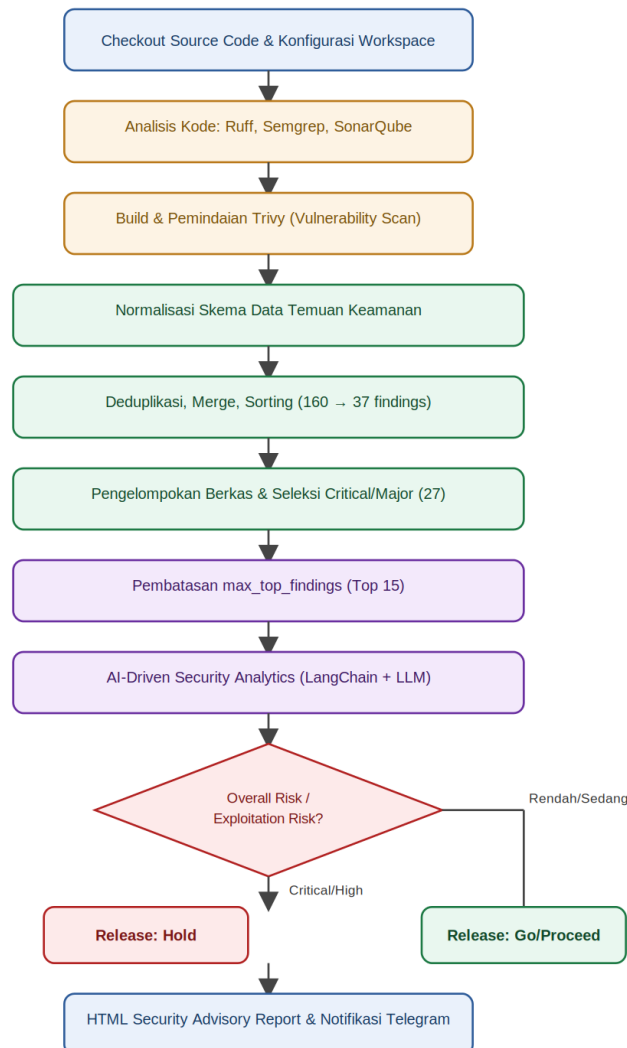
Gambar 1. Arsitektur Intelligent DevSecOps Automation Framework yang diusulkan Security Analysis Layer mengintegrasikan Ruff, Semgrep, SonarQube, dan Trivy sebagai alat analisis dengan cakupan pemeriksaan yang saling melengkapi. Ruff digunakan untuk

pemeriksaan kualitas dan konsistensi kode, Semgrep untuk analisis keamanan kode secara statis, SonarQube untuk analisis kualitas dan keamanan kode, sedangkan Trivy digunakan untuk pemindaian kerentanan pada komponen yang menjadi objek analisis. Hasil analisis dari berbagai alat selanjutnya diteruskan ke Finding Intelligence Layer untuk dinormalisasi dan dikonsolidasikan sebelum digunakan pada tahap analisis berikutnya. Data keamanan yang telah diproses menjadi masukan bagi AI Analytics Layer, yang menggunakan LangChain sebagai komponen orkestrasi interaksi dengan LLM untuk menghasilkan security advisory dan rekomendasi mitigasi. Hasil tersebut selanjutnya digunakan pada Decision and Response Layer untuk mendukung evaluasi keamanan dan rekomendasi rilis, dengan keluaran akhir berupa laporan HTML Security Advisory, notifikasi, serta artefak hasil pipeline. Pemisahan fungsi antarlapisan dirancang untuk menjaga independensi antara proses deteksi, pemrosesan temuan, analisis berbasis AI, dan pengambilan keputusan keamanan.

Implementasi Agent-Based Jenkins Pipeline

Framework diimplementasikan menggunakan Jenkins sebagai automation server dan orkestrator utama pipeline. Proses eksekusi dijalankan pada Jenkins agent yang menyediakan lingkungan untuk menjalankan tahapan analisis kode, pemindaian keamanan, pemrosesan hasil, dan integrasi dengan komponen AI. Pipeline dimulai ketika source code tersedia pada repositori dan proses eksekusi Jenkins dipicu sesuai mekanisme yang dikonfigurasi. Tahap awal mencakup proses checkout, persiapan lingkungan, dan konfigurasi workspace. Setelah lingkungan eksekusi siap, pipeline menjalankan Ruff, Semgrep, dan SonarQube untuk memperoleh informasi kualitas kode dan potensi permasalahan keamanan pada kode sumber. Pada tahap berikutnya, proses build dilakukan dan Trivy digunakan untuk memindai kerentanan sesuai objek pemindaian yang diterapkan dalam eksperimen. Hasil dari setiap alat disimpan dalam format yang konsisten agar dapat diproses lebih lanjut pada tahap normalisasi.

Alur Implementasi Agent-Based Jenkins Pipeline



Gambar 2. Alur implementasi Agent-Based Jenkins Pipeline

Tahap normalisasi mengonversi keluaran seluruh security tools ke dalam skema data yang seragam, mengingat setiap alat memiliki struktur keluaran, aturan identifikasi, dan klasifikasi severity yang berbeda. Proses ini bertujuan mengurangi fragmentasi hasil analisis yang muncul akibat perbedaan format keluaran dari setiap alat. Temuan yang telah dinormalisasi selanjutnya digunakan untuk membentuk gambaran kondisi keamanan aplikasi berdasarkan tingkat keparahan dan kriteria evaluasi yang diterapkan dalam framework. Hasil tersebut menjadi masukan bagi mekanisme security quality gate dan analisis berbasis AI, sehingga evaluasi keamanan tidak hanya bergantung pada keluaran satu alat.

Lingkungan dan Skenario Eksperimen

Eksperimen dilakukan pada lingkungan yang dirancang untuk merepresentasikan proses pengembangan perangkat lunak berbasis CI/CD dengan integrasi pengujian keamanan secara otomatis. Jenkins digunakan sebagai orkestrator utama untuk mengendalikan eksekusi pipeline pada agent, sedangkan Ruff, Semgrep, SonarQube, dan Trivy digunakan untuk menganalisis aspek keamanan dan kualitas perangkat lunak. LangChain digunakan sebagai lapisan orkestrasi untuk menghubungkan hasil analisis keamanan dengan LLM, sedangkan hasil akhir disajikan melalui laporan HTML Security Advisory dan notifikasi Telegram. Konfigurasi lingkungan eksperimen ditunjukkan pada Tabel 1.

Tabel 1.
Konfigurasi Lingkungan Eksperimen

Kategori	Komponen	Konfigurasi/Teknologi	Fungsi dalam Eksperimen
Orkestrasi	Jenkins	Agent-based pipeline	Mengorkestrasi seluruh tahapan pipeline
Eksekusi	Jenkins Agent	Lingkungan eksekusi terisolasi	Menjalankan build, analisis, dan pemrosesan hasil
Analisis kode	Ruff	Linter dan static analysis	Mendeteksi permasalahan kualitas dan konsistensi kode
Analisis keamanan	Semgrep	SAST berbasis aturan	Mendeteksi pola kerentanan pada kode sumber
Analisis kualitas dan keamanan	SonarQube	Static code analysis dan quality gate	Menganalisis kualitas dan keamanan kode
Analisis kerentanan	Trivy	Vulnerability scanner	Mendeteksi kerentanan pada artefak yang dipindai
Orkestrasi AI	LangChain	LLM orchestration framework	Menyiapkan dan mengelola alur analisis berbasis AI
Analisis AI	Large Language Model	Model LLM yang digunakan dalam eksperimen	Menghasilkan analisis dan rekomendasi mitigasi
Pelaporan	HTML Report	Security advisory report	Menyajikan hasil analisis keamanan secara terstruktur
Notifikasi	Telegram	Bot/API	Menyampaikan ringkasan hasil pipeline

Setelah lingkungan dikonfigurasi, eksperimen dijalankan dengan memproses aplikasi uji melalui keseluruhan alur Intelligent DevSecOps Automation Framework, mulai dari pemrosesan kode sumber pada Jenkins Pipeline, analisis menggunakan multi-security tools, pengumpulan dan pemrosesan hasil pemindaian, analisis keamanan berbasis AI, hingga

pembentukan security advisory dan rekomendasi keamanan. Pendekatan ini digunakan untuk mengevaluasi kemampuan framework secara end-to-end, bukan hanya keberhasilan masing-masing alat secara individual. Skenario eksperimen framework disajikan pada Tabel 2.

Tabel 2.
Skenario Eksperimen Framework

Tahap Eksperimen	Proses yang Diuji	Data yang Dihasilkan	Tujuan Evaluasi
Eksekusi pipeline	Menjalankan seluruh tahapan pada Jenkins Agent	Status dan log eksekusi	Memverifikasi keberhasilan otomasi end-to-end
Analisis keamanan	Menjalankan Ruff, Semgrep, SonarQube, dan Trivy	Temuan dari setiap alat	Mengevaluasi integrasi multi-security tools
Pemrosesan temuan	Mengolah hasil dari berbagai alat keamanan	Temuan keamanan terintegrasi	Mengevaluasi pengolahan hasil lintas alat
Analisis berbasis AI	Memproses konteks keamanan melalui LangChain dan LLM	Analisis dan rekomendasi mitigasi	Mengevaluasi pembentukan security advisory
Evaluasi keamanan	Mengevaluasi hasil analisis dan kriteria keamanan	Status risiko dan rekomendasi rilis	Mengevaluasi dukungan pengambilan keputusan
Distribusi hasil	Menghasilkan laporan HTML dan notifikasi	Laporan dan ringkasan hasil	Memverifikasi penyampaian umpan balik keamanan

Metode Evaluasi

Evaluasi dilakukan berdasarkan hasil eksperimen untuk menilai kemampuan framework dalam menjalankan proses keamanan secara terintegrasi dan otomatis. Evaluasi difokuskan pada aspek yang berkaitan langsung dengan tujuan penelitian, yaitu integrasi multi-security tools, otomasi pembentukan security advisory, pemanfaatan LLM dalam analisis hasil keamanan dan rekomendasi mitigasi, serta efektivitas keseluruhan framework, sebagaimana disajikan pada Tabel 3.

Tabel 3.
Metode Evaluasi

Aspek Evaluasi	Fokus Evaluasi
Integrasi multi-security tools	Kemampuan pipeline menjalankan dan mengintegrasikan alat keamanan
Otomasi security advisory	Kemampuan framework menghasilkan laporan keamanan secara otomatis
Analisis berbasis LLM	Kemampuan menghasilkan interpretasi dan rekomendasi mitigasi

Aspek Evaluasi	Fokus Evaluasi
Efektivitas framework	Hasil keseluruhan proses berdasarkan indikator eksperimen

Data evaluasi diperoleh dari keluaran aktual selama eksperimen, seperti hasil pemindaian masing-masing alat, jumlah dan kategori temuan, tingkat keparahan, status quality gate, hasil analisis AI, rekomendasi mitigasi, dan rekomendasi rilis. Analisis hasil dilakukan secara deskriptif dan komparatif sesuai rancangan eksperimen, sehingga hasil dari masing-masing skenario dapat digunakan untuk menilai bagaimana framework memproses temuan keamanan dari tahap deteksi hingga menghasilkan keluaran yang mendukung proses mitigasi dan pengambilan Keputusan

HASIL DAN PEMBAHASAN

Hasil Implementasi Intelligent DevSecOps Automation Framework

Hasil implementasi menunjukkan bahwa Intelligent DevSecOps Automation Framework berhasil diterapkan pada lingkungan CI/CD berbasis Agent-Based Jenkins Pipeline. Framework mengintegrasikan proses analisis kualitas kode, analisis keamanan statis, pemindaian kerentanan, pemrosesan hasil keamanan, analisis berbasis AI, dan penyampaian security advisory dalam satu alur eksekusi otomatis. Pipeline dijalankan melalui Jenkins dengan eksekusi pada agent, di mana setiap tahapan memiliki fungsi terpisah namun menghasilkan keluaran yang diteruskan ke tahapan berikutnya: Ruff untuk pemeriksaan kualitas kode, Semgrep untuk mendeteksi pola keamanan melalui analisis statis, SonarQube untuk mengevaluasi kualitas dan keamanan kode, dan Trivy untuk memindai kerentanan pada artefak yang menjadi objek pengujian. Hasil keseluruhan proses tersebut kemudian digunakan sebagai masukan bagi mekanisme analisis keamanan dan pembentukan security advisory. Implementasi ini menunjukkan bahwa proses keamanan yang sebelumnya dijalankan secara terpisah dapat diorkestrasi dalam satu alur pipeline, sehingga hasil analisis dari beberapa alat keamanan tersedia dalam satu siklus eksekusi dan proses identifikasi temuan serta penyampaian umpan balik kepada pengembang dapat dilakukan secara lebih sistematis.

Hasil Multi-Security Scanning dan Normalisasi

Hasil eksperimen menunjukkan bahwa proses pemindaian keamanan menghasilkan 160 raw findings yang berasal dari empat security tools. Trivy menghasilkan jumlah temuan terbesar, yaitu 115 temuan (71,88%), diikuti oleh Ruff sebanyak 33 temuan (20,63%), Semgrep sebanyak 8 temuan (5,00%), dan SonarQube sebanyak 4 temuan (2,50%), sebagaimana disajikan pada Tabel 4. Perbedaan jumlah temuan tersebut mencerminkan cakupan dan karakteristik deteksi yang berbeda pada setiap alat: Trivy lebih dominan dalam mendeteksi kerentanan pada dependensi perangkat lunak dan komponen container image, sedangkan Ruff, Semgrep, dan SonarQube lebih berfokus pada analisis kode sumber, kualitas kode, serta pola kerentanan aplikasi.

Tabel 4.

Hasil Temuan Keamanan Sebelum Proses Deduplication

Security Tool	Domain Analisis	Jumlah Raw Findings	Persentase
Trivy	Software Composition Analysis (SCA) / Container Security	115	71,88%
Ruff	Static Code Analysis / Linting	33	20,63%
Semgrep	Static Application Security Testing (SAST)	8	5,00%
SonarQube	Static Code Analysis / Code Quality & Security	4	2,50%
Total	Multi-tool Security Analysis	160	100%

Meskipun berasal dari alat yang berbeda, seluruh hasil pemindaian dikonversi ke dalam skema data yang seragam melalui proses normalisasi, karena masing-masing security tool memiliki struktur keluaran, aturan identifikasi, klasifikasi severity, dan format atribut yang berbeda. Proses ini menghasilkan 160 temuan terstandarisasi, sehingga seluruh temuan dapat diproses menggunakan mekanisme konsolidasi dan penilaian risiko yang sama. Setelah normalisasi, sistem melakukan deduplikasi terhadap temuan yang mengacu pada permasalahan keamanan yang sama. Berdasarkan hasil eksperimen, 160 raw findings tersebut dikonsolidasikan menjadi 37 unique findings, sehingga proses deduplikasi mengurangi 123 temuan redundan atau sekitar 76,88% dari keseluruhan temuan awal.

Hasil Konsolidasi dan Prioritisasi Temuan Keamanan

Setelah seluruh keluaran security tools dikonversi ke dalam skema data yang seragam, framework melakukan konsolidasi untuk mengurangi redundansi dan membentuk representasi temuan yang lebih terstruktur. Sebanyak 160 temuan yang telah dinormalisasi

diproses melalui tahapan deduplikasi, penggabungan (merge), pengurutan, pengelompokan berdasarkan berkas, dan seleksi berdasarkan tingkat keparahan. Hasil keseluruhan proses tersebut disajikan pada Tabel 5.

Tabel 5.

Hasil Konsolidasi dan Prioritisasi Temuan Keamanan

Tahap Pemrosesan	Jumlah	Perubahan	Keterangan
Normalized findings	160	-	Seluruh temuan telah dikonversi ke skema data seragam
Unique findings after deduplication	37	-123	Temuan redundan dikonsolidasikan
Findings after merge	37	0	Temuan unik digabungkan dalam satu kumpulan data
Findings after sorting	37	0	Temuan diurutkan berdasarkan mekanisme prioritas
File groups after grouping	11	-	Temuan dikelompokkan berdasarkan berkas terkait

Hasil eksperimen menunjukkan bahwa proses deduplikasi mengurangi jumlah entri dari 160 menjadi 37 temuan unik. Sebanyak 123 entri redundan berhasil dikonsolidasikan, sehingga jumlah data yang diteruskan ke tahap berikutnya berkurang sebesar 76,88%, sedangkan 23,13% dari keseluruhan temuan awal dipertahankan sebagai temuan unik untuk proses analisis lanjutan. Reduksi tersebut tidak menunjukkan penghapusan kerentanan atau pengurangan false positive, melainkan pengurangan representasi temuan redundan berdasarkan mekanisme deduplikasi yang diterapkan dalam framework. Jumlah temuan tetap 37 pada tahap merge dan pengurutan, yang menunjukkan bahwa kedua tahap tersebut tidak digunakan untuk mengurangi jumlah temuan, melainkan untuk mengintegrasikan temuan unik ke dalam satu kumpulan data dan menyusun urutan pemrosesan berdasarkan prioritas. Proses pengelompokan selanjutnya menghasilkan 11 kelompok berkas, sehingga temuan dapat dianalisis berdasarkan konteks lokasi atau komponen kode yang terdampak.

Dari 37 temuan unik, sebanyak 27 temuan dikategorikan sebagai Critical atau Major, yang merepresentasikan 72,97% dari keseluruhan temuan unik. Hasil ini menunjukkan bahwa sebagian besar temuan yang dipertahankan setelah deduplikasi berada pada kategori prioritas tinggi, sehingga seleksi berdasarkan tingkat keparahan memungkinkan framework memfokuskan proses analisis lanjutan pada temuan yang memerlukan perhatian lebih besar tanpa menghilangkan keterlacakan terhadap keseluruhan hasil pemindaian. Berdasarkan

hasil konsolidasi, temuan berprioritas tinggi mencakup beberapa kategori keamanan, antara lain `insecure_hash`, `unsafe_eval`, `sql_injection`, `hardcoded_secret`, `request_without_timeout`, dan `dependency_vulnerability`, sebagaimana disajikan pada Tabel 6.

Tabel 6.
Contoh Kategori Temuan Berprioritas Tinggi

Kategori Temuan	Lokasi	Raw Issues	Severity
<code>dependency_vulnerability</code>	Dockerfile:1	100	Critical
<code>dependency_vulnerability</code>	requirements.txt:1	15	Critical
<code>insecure_hash</code>	app/crypto.py:5	2	Critical
<code>unsafe_eval</code>	app/payment.py:6	2	Critical
<code>sql_injection</code>	app/db_data.py:28	1	Critical
<code>sql_injection</code>	app/auth.py:6	1	Critical
<code>hardcoded_secret</code>	app/vuln_examples.py:10	1	Critical
<code>hardcoded_secret</code>	app/db_data.py:45	1	Critical
<code>request_without_timeout</code>	app/request.py:39	1	Critical
<code>unused_variable</code>	app/payment.py:3	2	Major

Hasil tersebut menunjukkan bahwa jumlah raw findings tidak dapat secara langsung diinterpretasikan sebagai jumlah masalah keamanan yang sepenuhnya berbeda. Sebagai contoh, `dependency_vulnerability` pada Dockerfile merepresentasikan 100 raw issues, sedangkan kategori yang sama pada requirements.txt merepresentasikan 15 raw issues, sehingga kedua lokasi tersebut tetap dicatat sebagai temuan unik yang terpisah meskipun berasal dari kategori yang sama. Kondisi ini menjelaskan perbedaan signifikan antara jumlah temuan sebelum dan setelah proses deduplikasi, sekaligus menegaskan bahwa framework mengurangi redundansi penyajian tanpa menghilangkan informasi mengenai jumlah kejadian yang direpresentasikan oleh setiap temuan. Proses konsolidasi ini juga berperan menyiapkan data untuk tahap AI-Driven Security Analytics, karena penggunaan 37 temuan unik yang telah diurutkan dan dikelompokkan menghasilkan konteks keamanan yang lebih terstruktur dibandingkan meneruskan seluruh 160 temuan secara langsung.

Hasil AI-Driven Security Analytics dan Security Advisory

Setelah proses konsolidasi dan prioritasasi, framework menghasilkan 37 temuan unik yang terdiri atas 27 temuan berkategori Critical atau Major sebagai temuan berprioritas tinggi. Sebelum diteruskan ke komponen AI-Driven Security Analytics, framework menerapkan mekanisme pembatasan jumlah temuan melalui parameter `max_top_findings`. Pada

eksperimen ini, nilai parameter ditetapkan sebesar 15, sehingga hanya 15 temuan dengan prioritas tertinggi yang digunakan sebagai konteks utama untuk analisis berbasis LLM. Pembatasan tersebut diterapkan untuk mengendalikan ukuran konteks masukan dan penggunaan token selama proses inferensi, dan dilakukan setelah temuan melalui tahapan normalisasi, deduplikasi, penggabungan, pengurutan, dan prioritasasi, sehingga seleksi top findings tidak dilakukan secara acak melainkan tetap mempertahankan temuan dengan tingkat kepentingan tertinggi sebagai fokus analisis. Ringkasan transformasi temuan menuju tahap AI-Driven Security Analytics disajikan pada Tabel 7.

Tabel 7.

Transformasi Temuan Menuju AI-Driven Security Analytics

Tahap	Jumlah	Keterangan	Tahap
Raw findings	160	Total keluaran dari seluruh security tools	Raw findings
Normalized findings	160	Temuan setelah standardisasi format	Normalized findings
Unique findings	37	Temuan setelah deduplikasi	Unique findings
Critical/Major findings	27	Temuan berprioritas tinggi	Critical/Major findings
max_top_findings	15	Batas maksimum temuan yang dikirim untuk analisis AI	max_top_findings
Top findings analyzed	15	Temuan prioritas tertinggi yang menjadi konteks LLM	Top findings analyzed

Hasil analisis terhadap 15 temuan prioritas tersebut selanjutnya digunakan untuk membentuk Security Advisory Report. Pada eksperimen terhadap payment-service, laporan menghasilkan penilaian Overall Risk: Critical, Exploitation Risk: Critical, dan rekomendasi rilis Hold, serta mengidentifikasi 14 dari 15 temuan tersebut sebagai Release Blockers dan menghasilkan rekomendasi tindakan perbaikan segera berdasarkan konteks keamanan yang dianalisis. Hasil ini menunjukkan bahwa framework mampu mengubah temuan keamanan yang telah diprioritaskan menjadi laporan yang menyajikan ringkasan risiko, rekomendasi rilis, temuan yang berpotensi menghambat proses rilis, dan tindakan remediasi. Kondisi keamanan aplikasi yang diklasifikasikan pada tingkat risiko Critical menghasilkan rekomendasi Hold, yang menunjukkan bahwa proses rilis sebaiknya ditunda hingga temuan yang dikategorikan sebagai release blocker memperoleh penanganan yang sesuai.

Evaluasi Security Quality Gate dan Rekomendasi Rilis

Hasil analisis keamanan selanjutnya digunakan untuk mengevaluasi kondisi keamanan aplikasi dan mendukung proses pengambilan keputusan rilis. Dalam framework yang

dikembangkan, keputusan keamanan tidak hanya didasarkan pada jumlah temuan, tetapi juga mempertimbangkan tingkat keparahan, hasil prioritisasi, potensi eksploitasi, dan dampak temuan terhadap proses rilis. Security Quality Gate tetap berfungsi sebagai kontrol keamanan yang berbasis pada hasil pemeriksaan dan kriteria yang telah ditentukan, sedangkan hasil AI-Driven Security Analytics memberikan interpretasi kontekstual dan rekomendasi terhadap kondisi keamanan yang ditemukan. Pemisahan fungsi ini memastikan bahwa Large Language Model tidak menjadi satu-satunya dasar dalam menentukan kelayakan rilis, tetapi berperan sebagai mekanisme pendukung analisis untuk membantu pengembang atau tim keamanan memahami risiko dan menentukan tindakan yang diperlukan. Ringkasan hasil evaluasi keamanan dan rekomendasi rilis disajikan pada Tabel 8.

Tabel 8.
Hasil Evaluasi Keamanan dan Rekomendasi Rilis

Indikator Evaluasi	Hasil
Temuan unik setelah deduplikasi	37
Temuan Critical/Major	27
Top findings yang dianalisis AI	15
Overall Risk	Critical
Exploitation Risk	Critical
Release Blockers	14
Release Recommendation	Hold

Berdasarkan hasil eksperimen, kombinasi antara hasil pemindaian multi-security tools, mekanisme konsolidasi dan prioritisasi temuan, Security Quality Gate, serta AI-Driven Security Analytics menghasilkan alur evaluasi keamanan yang terintegrasi. Pendekatan ini memungkinkan temuan keamanan diproses secara bertahap, dari data teknis hasil pemindaian menjadi informasi risiko dan rekomendasi rilis yang lebih mudah digunakan dalam proses pengambilan keputusan pada pipeline CI/CD.

Evaluasi Efektivitas Framework

Evaluasi efektivitas framework dilakukan berdasarkan kemampuan sistem dalam mengintegrasikan proses analisis keamanan, mengelola temuan dari berbagai security tools, memprioritaskan informasi keamanan, serta menghasilkan security advisory yang mendukung proses remediasi dan pengambilan keputusan rilis. Hasil eksperimen menunjukkan bahwa framework berhasil mengintegrasikan Ruff, Semgrep, SonarQube, dan Trivy dalam satu alur Agent-Based Jenkins Pipeline. Keempat alat tersebut menghasilkan

total 160 raw findings dengan karakteristik dan cakupan analisis yang berbeda, dan seluruh temuan berhasil dinormalisasi ke dalam skema data yang seragam sehingga dapat diproses melalui mekanisme konsolidasi yang sama. Proses deduplikasi mengonsolidasikan 160 normalized findings menjadi 37 unique findings, mengurangi 123 entri redundan atau sebesar 76,88%, dan temuan unik tersebut selanjutnya dikelompokkan menjadi 11 kelompok berkas, dengan 27 temuan berkategori Critical atau Major diprioritaskan untuk analisis lanjutan.

Untuk mengendalikan ukuran konteks dan penggunaan token pada proses analisis berbasis LLM, framework menerapkan mekanisme pembatasan melalui parameter `max_top_findings` dengan nilai 15, sehingga 15 temuan dengan prioritas tertinggi dipilih sebagai konteks utama untuk AI-Driven Security Analytics. Hasil AI-Driven Security Analytics pada payment-service menghasilkan Overall Risk dan Exploitation Risk pada kategori Critical, mengidentifikasi 14 Release Blockers, serta menghasilkan Release Recommendation berupa Hold, disertai rekomendasi tindakan remediasi yang disajikan melalui Security Advisory Report. Hasil tersebut menunjukkan bahwa efektivitas framework terletak pada integrasi end-to-end antara proses deteksi, normalisasi, konsolidasi, prioritisasi, analisis berbasis AI, dan penyajian security advisory. Framework tidak menggantikan fungsi security tools sebagai sumber utama deteksi kerentanan, tetapi menambahkan lapisan pengolahan dan analisis untuk mengurangi fragmentasi hasil serta meningkatkan keterbacaan informasi keamanan. Ringkasan evaluasi efektivitas framework disajikan pada Tabel 9.

Tabel 9.

Ringkasan Evaluasi Efektivitas Framework

Aspek Evaluasi	Hasil Eksperimen	Indikasi Efektivitas
Integrasi multi-security tools	4 alat terintegrasi dalam satu pipeline	Analisis keamanan dapat dijalankan dalam satu alur otomatis
Agregasi temuan	160 raw findings	Hasil dari beberapa alat berhasil dikumpulkan
Normalisasi	160 normalized findings	Keluaran heterogen dapat diproses dalam skema seragam
Deduplication	160 menjadi 37 unique findings	123 entri redundan dikonsolidasikan
Reduksi representasi redundan	76,88%	Mengurangi volume data repetitif sebelum analisis lanjutan
Pengelompokan	11 file groups	Temuan dapat disusun berdasarkan konteks berkas
Prioritisasi	27 Critical/Major findings	Analisis difokuskan pada temuan berprioritas tinggi

Context-budget control	Top 15 findings	Membatasi konteks LLM melalui max top findings
AI security analytics	Overall Risk: Critical	Menghasilkan interpretasi risiko
Evaluasi dampak rilis	14 Release Blockers	Mengidentifikasi temuan yang berdampak terhadap rilis
Rekomendasi rilis	Hold	Menghasilkan dukungan keputusan rilis

Pembahasan dan Implikasi Penelitian

Hasil penelitian menunjukkan bahwa Intelligent DevSecOps Automation Framework mampu mengintegrasikan keluaran multi-security tools ke dalam satu alur keamanan yang terstruktur. Proses normalisasi dan deduplikasi mengurangi redundansi representasi temuan, sedangkan mekanisme pengelompokan dan prioritisasi memungkinkan analisis difokuskan pada temuan dengan tingkat risiko yang lebih tinggi. Temuan ini sejalan dengan penelitian Zhao et al. (2024) yang menegaskan bahwa integrasi beberapa security tools tidak cukup hanya dengan menjalankannya dalam satu pipeline, tetapi juga memerlukan mekanisme pengolahan hasil agar informasi keamanan dapat digunakan secara lebih efektif.

Integrasi AI-Driven Security Analytics memberikan nilai tambah pada tahap interpretasi dan penyusunan rekomendasi keamanan. Dalam framework ini, LLM tidak menggantikan fungsi security tools sebagai sumber deteksi, tetapi digunakan untuk menganalisis temuan yang telah diprioritaskan dan menghasilkan security advisory yang lebih kontekstual, sejalan dengan temuan Bedoya et al. (2024) mengenai peran LLM dalam memperkaya praktik DevSecOps. Penerapan max_top_findings juga memungkinkan jumlah temuan yang dikirim ke LLM dibatasi berdasarkan prioritas, sehingga ukuran konteks dan penggunaan token dapat dikendalikan, sekaligus menjawab tantangan efisiensi komputasi yang disoroti oleh Camacho (2024) dalam pemanfaatan AI/ML pada DevSecOps.

Dibandingkan dengan penelitian Kokku (2024) dan Kiran Kumar Reddy Yelkoti (2025) yang masing-masing berfokus pada otomasi pengujian keamanan dan arsitektur security as code secara terpisah, kontribusi utama penelitian ini terletak pada integrasi end-to-end antara multi-security scanning, konsolidasi temuan, prioritisasi berbasis risiko, pengendalian konteks LLM, dan AI-assisted security advisory dalam satu Agent-Based Jenkins Pipeline. Secara praktis, framework membantu mengubah keluaran keamanan yang tersebar menjadi informasi yang lebih terstruktur untuk mendukung proses remediasi dan evaluasi rilis. Meskipun demikian, keluaran LLM tetap diposisikan sebagai pendukung keputusan, karena kualitas hasilnya bergantung pada konteks masukan, konfigurasi prompt, dan model yang

digunakan, sehingga keterlibatan tim keamanan dalam memverifikasi rekomendasi AI tetap diperlukan sebelum keputusan rilis final ditetapkan.

KESIMPULAN DAN REKOMENDASI

Penelitian ini berhasil mengembangkan Intelligent DevSecOps Automation Framework berbasis Agent-Based Jenkins Pipeline yang mengintegrasikan Ruff, Semgrep, SonarQube, dan Trivy dalam satu alur analisis keamanan otomatis. Framework mampu menormalisasi dan mengonsolidasikan 160 raw findings menjadi 37 unique findings, dengan 27 temuan berkategori Critical atau Major. Melalui mekanisme prioritisasi dan pembatasan konteks, 15 temuan dengan prioritas tertinggi dianalisis menggunakan LangChain dan Large Language Model untuk menghasilkan security advisory dan rekomendasi mitigasi secara otomatis. Hasil eksperimen pada payment-service menghasilkan Overall Risk dan Exploitation Risk pada kategori Critical, 14 Release Blockers, serta Release Recommendation berupa Hold. Hasil tersebut menunjukkan bahwa framework mampu mengintegrasikan proses deteksi, konsolidasi, prioritisasi, dan analisis keamanan berbasis AI dalam satu alur CI/CD untuk mendukung remediiasi dan pengambilan keputusan keamanan.

Penelitian selanjutnya disarankan mengevaluasi framework pada aplikasi dan lingkungan yang lebih beragam, mengukur akurasi dan konsistensi keluaran LLM secara lebih sistematis, serta mempertimbangkan pengujian pada berbagai bahasa pemrograman dan skala repositori yang lebih besar untuk memperkuat generalisasi hasil penelitian.

REFERENSI

1. Abiola, O., Abiola, O. B., & Olufemi, O. G. (2023). An enhanced CI/CD pipeline: A DevSecOps approach. *International Journal of Computer Applications*, 184(48). <https://www.researchgate.net/publication/373950058>
2. Aljohani, M. A., & Alqahtani, S. S. (2023). A unified framework for automating software security analysis in DevSecOps. In *Proceedings of the International Conference on Smart Computing Applications (ICSCA)*. <https://doi.org/10.1109/ICSCA57840.2023.10087568>

3. Anjaria, D., & Kulkarni, M. (2022). Effective DevSecOps implementation: A systematic literature review. <https://doi.org/10.18137/cardiom>
4. Bedoya, M., Palacios, S., Díaz-López, D., Laverde, E., & Nespoli, P. (2024). Enhancing DevSecOps practice with large language models and security chaos engineering. *International Journal of Information Security*, 23(6), 3765–3788. <https://doi.org/10.1007/s10207-024-00909-w>
5. Binbeshr, F., & Imam, M. (2025). Comparative analysis of AI-driven security approaches in DevSecOps: Challenges, solutions, and future directions. *arXiv*. <https://arxiv.org/abs/2504.19154>
6. Camacho, N. G. (2024). Unlocking the potential of AI/ML in DevSecOps: Effective strategies and optimal practices. *Journal of Artificial Intelligence General Science*, 2(1), 79–89. <https://doi.org/10.60087/jaigs.v2i1.p89>
7. Chandra Thota, R. (2024). Cloud-native DevSecOps: Integrating security automation into CI/CD pipelines. *International Journal of Innovative Research and Creative Technology*, 1. <https://doi.org/10.5281/zenodo.15036934>
8. Cheenepalli, J., Hastings, J. D., Ahmed, K. M., & Fenner, C. (2025). Advancing DevSecOps in SMEs: Challenges and best practices for secure CI/CD pipelines. <https://doi.org/10.1109/ISDFS65363.2025.11011960>
9. Chittala, S. (2024). Securing DevOps pipelines: Automating security in DevSecOps frameworks. *Journal of Recent Trends in Computer Science and Engineering*, 12(5), 31–44. <https://doi.org/10.70589/jrtcse.2024.5.5>
10. Fu, M., Pasuksmit, J., & Tantithamthavorn, C. (2025). AI for DevSecOps: A landscape and future opportunities. *ACM Transactions on Software Engineering and Methodology*, 34(4), Article 115. <https://doi.org/10.1145/3712190>
11. Haverinen, H., Janhunen, T., Päivärinta, T., Lempinen, S., Kaartinen, S., & Merilä, S. (2024). Automating cybersecurity compliance in DevSecOps with open information model for security as code. In *ACM International Conference Proceeding Series* (pp. 93–102). <https://doi.org/10.1145/3685651.3686700>

12. Kağızmandere, Ö., & Arslan, H. (2024). Vulnerability analysis based on SBOMs: A model proposal for automated vulnerability scanning for CI/CD pipelines. *International Journal of Information Security Science*, 13(2), 33–42. <https://doi.org/10.55859/ijiss.1455039>
13. Kiran Kumar Reddy Yelkoti, N. (2025). Security as code: An architectural framework for automated risk mitigation in DevSecOps pipelines. *Journal of Computer Science and Technology Studies*, 7(6), 235–244. <https://doi.org/10.32996/jcsts.2025.7.6.26>
14. Kokku, R. (2024). Revolutionizing DevOps security: AI and ML-enabled automated testing approaches. *International Journal of Current Science Research and Review*, 7(11). <https://doi.org/10.47191/ijcsrr/V7-i11-01>
15. Malik, G., & Sachs, G. (2025). Integrating threat intelligence with DevSecOps: Automating risk mitigation before code hits production. <https://www.researchgate.net/publication/395022579>
16. Nocera, S., Fucci, D., & Scanniello, G. (2025). Dealing with SonarQube Cloud: Initial results from a mining software repository study. *arXiv*. <https://arxiv.org/abs/2508.18816>
17. Prates, L., & Pereira, R. (2025). DevSecOps practices and tools. *International Journal of Information Security*, 24(1). <https://doi.org/10.1007/s10207-024-00914-z>
18. Ramaj, X., Sánchez-Gordón, M., Gkioulos, V., & Colomo-Palacios, R. (2024). On DevSecOps and risk management in critical infrastructures: Practitioners' insights on needs and goals. In *Proceedings of EnCyCriS/SVM 2024* (pp. 45–52). <https://doi.org/10.1145/3643662.3643954>
19. Sarkar, S., & Castro, H. (2024). Policy-as-code: Enforcing governance with Open Policy Agent (OPA). <https://www.researchgate.net/publication/391016222>
20. Vakhula, O., & Opirskyy, I. (2025). AI-driven security as code for software development using multi-agent systems.
21. Zhao, X., Clear, T., & Lal, R. (2024). Identifying the primary dimensions of DevSecOps: A multi-vocal literature review. *The Journal of Systems and Software*, 214, 112063. <https://doi.org/10.1016/j.jss.2024.112063>

-
22. Zhou, X., Mao, R., Zhang, H., Dai, Q., Huang, H., Shen, H., Li, J., & Rong, G. (2023). Revisit security in the era of DevOps: An evidence-based inquiry into DevSecOps industry. *IET Software*, 17(4), 435–454. <https://doi.org/10.1049/sfw2.12132>