Hidup. : Pendekatan Sistem Agenik pada SSDLC Berdasarkan OWASP CLASP

Author: 1st Latief Reswandana
Politeknik Siber dan Sandi Negara
Bandung, Indonesia
mochammad.latief@student.polteksn.ac.id

***Abstract*—Manajemen Secure Software Development Life Cycle (SSDLC) merupakan tugas yang sangat kompleks karena melibatkan koordinasi beberapa peran, aktivitas, dan sumber daya dan juga tetap memastikan tujuan proyek tercapai secara efisien dan memenuhi persyaratan keamanan. Namun, pendekatan SSDLC tradisional seringkali rentan terhadap kesalahan manusia, membutuhkan banyak intervensi manual dan terkadang dokumentasi yang dibuat tidak memenuhi rincian persyaratan keamanan.**

Paper ini memperkenalkan *Hidup.*, sebuah kerangka kerja dengan menggunakan pendekatan sistem kecerdasan buatan agenik pada proses SSDLC dengan mengacu pada kerangka kerja OWASP CLASP untuk mengatasi tantangan tersebut. Dengan memanfaatkan kemampuan Large Language Model (LLM), kerangka kerja ini bertujuan untuk mengotomatisasi dan menyederhanakan tugas-tugas kompleks dalam SSDLC. Agen-agen dalam sistem ini dirancang untuk mempresentasikan peran-peran spesifik dalam setiap tahap SSDLC, sehingga mendukung otomatisasi yang adaptif dan efisien dalam pengembangan perangkat lunak yang aman.

Penelitian yang dilakukan menggunakan pendekatan Design Science Research Methodoloy (DSRM) dalam pengembangan artefak dan melakukan eksperimen terkontrol. Metodoli ini diambil dengan tujuan untuk menunjukan bahwa *Hidup.* dapat memberikan peningkatan efektivitas dan efisiensi dibandingkan menggunakan proses tradisional.

Keywords—SSDLC, Agentic, AI, LLM, Otomatisasi, CLASP, Otonom, Agen, DSRM, Eksperimen

I. INTRODUCTION

Dengan berkembangnya era keamanan teknologi informasi, pengembangan perangkat lunak yang aman menjadi prioritas utama. Cacat pada perangkat lunak, secara langsung dapat menyebabkan munculnya kerentanan keamanan, karena dapat membuka pintu masuk atau bahkan dapat memperluas permukaan serangan keamanan pada perangkat lunak [1]. Secure Software Development Life Cycle (SSDLC) adalah metodologi yang dirancang untuk mengintegrasikan praktik keamanan ke dalam Software Development Life Cycle (SDLC) yang bertujuan untuk mengurangi jalan masuk serangan dan menjaga keamanan perangkat lunak [2].

Namun, pendekatan SSDLC tradisional seringkali rentan terhadap kesalah manusia, membutuhkan intervensi manual, dan terkadang dokumentasi yang dibuat tidak memenuhi rincian persyaratan keamanan [3]. Tahapan pada SSDLC sangat kompleks, karena melibatkan koordinasi antara peran berkepentingan, aktivitas, dan sumber daya. Belum lagi, pendekatan tradisional sering kali tidak mampu menangani masalah keamanan dengan baik karena kurangnya kemampuan dan panduan terstruktur. Selain itu, pendekatan tradisional sering kali menumpuk peran-peran kepada seseorang dikarenakan kekurangan sumber daya manusia. Kekurangan tersebut bisa saja menyebabkan keterlambatan waktu produksi atau dapat meningkatkan kerawanan karena kesalahan manusia.

Untuk mengatasi keterbatasan tersebut, paper ini mengusulkan dan memperkenalkan pendekatan menggunakan sistem kecerdasan agenik. Sistem agenik mempresentasikan agen-agen khusus yang berkoordinasi, berkomunikasi, dan secara dinamis melakukan tindakan untuk menyelesaikan tugas kompleks [4]. Dengan memanfaatkan kemampuan Large Language Model (LLM) yang dapat melakukan penalaran dari konteks dan data yang kompleks, sistem agenik akan dapat mempertahankan dinamika tahapan dan pembaruan pada proses berkelanjutan. Kemampuan ini menawarkan skalabilitas, adaptabilitas, dan efisiensi yang lebih baik dalam pengembangan perangkat lunak yang aman.

Oleh karena itu paper ini memperkenalkan sebuah kerangka kerja *Hidup.,* yang menggunakan sistem kecerdasan buatan agenik pada proses SSDLC. Kerangka kerja ini secara khusus mengacu pada kerangka kerja OWASP CLASP. Dengan memanfaatkan LLM dan sistem agenik, kerangka kerja ini menawarkan kemampuan otomatisasi dan menyederhanakan tugas-tugas kompleks dalam SSDLC. Agen-agen didalam sistem agenik dirancang untuk mempresentasikan peran-peran spesifik dalam setiap tahap pengembangan perangkat lunak. Untuk dapat melakukan eksplorasi dan validasi kontribusi ini, penelitian ini menggunakan pendekatan Design Science Research Methodoloy sebagai landasan pengembangan artefak, dan ditambah eksperimen sebagai bagian dari evaluasi untuk mengukur efektivitas kerangka kerja *Hidup.* dibandingkan pendekatan manual. Hasil dari penelitian diharapkan dapat menyediakan asistensi untuk pengembangan sistem perangkat lunak, dan memberikan keluaran hasil yang lebih terukur, dapat dijelaskan, aman, dan lebih cepat tanpa memberikan pekerjaan tambahan untuk manusia.

II. LITERATURE REVIEW

Pada bab ini akan menjelaskan tentang SSDLC, kerangka kerja OWASP CLASP, kecerdasan buatan agenik dan penelitian terkait secara komprehensif dan mengidentifikasi gap dalam metodologi penelitian yang digunakan.

1. Secure Software Development Life Cycle (SSDLC)

SSDLC adalah metodologi yang dirancang untuk mengintegrasikan praktik keamanan ke dalam setiap fase siklus hidup pengembangan perangkat lunak [2]. Pendekatan ini sejalan dengan prinsip "shifting security left", yaitu menanamkan aspek keamanan sejak awal proses pengembangan untuk meminimalkan permukaan serangan dan meningkatkan ketahanan perangkat lunak secara keseluruhan.

SSDLC mencakup berbagai fase kritis, mulai dari perencanaan, analisis kebutuhan, desain, implementasi, hingga pemeliharaan. Dalam praktiknya, kerangka kerja seperti OWASP CLASP menyediakan panduan

komprehensif dan aktivitas keamanan yang dapat diintegrasikan ke dalam masing-masing fase tersebut [5].

Meskipun demikian, implementasi SSDLC secara tradisional sering kali menghadapi berbagai tantangan. dan rentan terhadap kesalahan manusia. Proses manajemen Kompleksitas proses, ketergantungan pada intervensi manual, dan keterbatasan sumber daya manusia dalam aspek keamanan sering menjadi sumber kesalahan atau kelalaian [3]. Selain itu, kurangnya panduan yang terstruktur serta pendekatan yang menempatkan keamanan sebagai tambahan di akhir proses dapat menyebabkan keterlambatan penanganan kerentanan bahkan setelah sistem diluncurkan. Hambatan lain yang umum meliputi minimnya pemahaman keamanan di antara pengembang, komunikasi lintas peran yang tidak efektif, serta kesulitan mendokumentasikan kebutuhan keamanan secara eksplisit dan dapat ditindaklanjuti [2].

2. OWASP Comprehensive, Lightweight Application Security Process (CLASP) Framework

Salah satu kerangka kerja untuk SSDLC adalah OWASP CLASP, yang dikenal sebagai metodologi yang ringan dan cocok untuk organisasi kecil dengan persyaratan keamanan yang tidak terlalu ketat [5]. CLASP berfokus pada pengintegrasian kekhawatiran keamanan ke tahap awal siklus pengembangan perangkat lunak. Kerangka kerja ini mengambil pendekatan berbasis peran yang memiliki dampak pada keamanan perangkat lunak dan bertanggung jawab atas finalisasi kualitas pada suatu aktivitas.

Berikut merupakan penjelasan umum mengenai peran dan aktivitas utama pada OWASP CLASP [6]:

2.1. Project Manager

Memiliki tujuan utama untuk memastikan dukungan organisasi terhadap praktik keamanan. Berikut aktivitas yang dilakukan oleh project manager:

a. Membuat program kesadaran keamanan untuk memastikan semua anggota proyek memahami pentingnya keamanan melalui pelatihan dan akuntabilitas.
b. Memantau metrik keamanan untuk mengukur postur keamanan proyek dan menegakkan akuntabilitas.
c. Mengelola pengungkapan isu keamanan untuk menjalin komunikasi dengan pihak eksternal dan pelanggan terkait isu keamanan perangkat lunak.

2.2. Requirements Specifier

Bertanggung jawab untuk merinci kebutuhan keamanan yang harus diperhatikan dalam desain dan arsitektur sistem. Aktivitas utama meliputi:

a. Menentukan lingkungan operasional untuk menilai dampaknya terhadap keamanan.
b. Mengidentifikasi kebijakan keamanan global sebagai baseline keamanan produk.
c. Mendokumentasikan persyaratan fungsional dan bisnis yang relevan terhadap keamanan.
d. Menyusun *misuse cases* untuk mengkomunikasikan potensi risiko keamanan kepada stakeholder.

2.3. Architect

Bertanggung jawab untuk memastikan sistem tidak memiliki celah keamanan pada tahap perancangan. Aktivitas utama:

a. Mengidentifikasi sumber daya dan batas kepercayaan sebagai dasar struktur keamanan.
b. Menentukan peran pengguna dan kapabilitas sumber daya yang mereka akses.
c. Menganotasi desain kelas dengan properti keamanan yang relevan.

2.4. Designer

Menjamin bahwa desain sistem meminimalkan risiko keamanan. Aktivitas utama:

a. Mengidentifikasi permukaan serangan (attack surface) untuk analisis lebih lanjut.
b. Menerapkan prinsip desain keamanan dalam rancangan aplikasi.
c. Menilai postur keamanan teknologi atau komponen pihak ketiga.
d. Menyelesaikan isu keamanan yang dilaporkan dalam tahap implementasi.

2.5. Implementor

Berperan dalam membangun perangkat lunak dengan mengikuti spesifikasi keamanan yang telah ditentukan. Aktivitas utama:

a. Mengimplementasikan kontrak antarmuka, termasuk validasi dan penanganan kesalahan.
b. Mengelaborasi kebijakan keamanan dan mengintegrasikan teknologi keamanan dalam kode.

2.6. Analyst

Bertugas mengidentifikasi kerentanan keamanan melalui pengujian perangkat lunak. Aktivitas utama:

a. Merancang dan melakukan pengujian keamanan berdasarkan kebutuhan sistem.
b. Memverifikasi atribut keamanan dari sumber daya sesuai kebijakan yang ditetapkan.

2.7. Security Auditor

Mengidentifikasi pelanggaran keamanan dan merekomendasikan langkah remediasi. Aktivitas utama:

a. Melakukan pemodelan ancaman untuk menganalisis risiko dari persyaratan dan desain.
b. Melakukan tinjauan keamanan pada tingkat kode sumber untuk menemukan kerentanan.

CLASP bertujuan untuk menempatkan keamanan dilaksanakan sejak dini dalam proses pengembangan [5], [6]. Selain itu dengan mengimplementasikan kerangka kerja ini, kita dapa mendefinisikan peran yang dapat memengaruhi tingkat keamanan perangkat lunak. CLASP juga menekan kan pemantauan metrik keamanan yang berkelanjutan di seluruh siklus hidup pengembangan untuk mengukur kemajuan keamanan dan progres pengembangan. Metrik ini dapat membantu mengidentifkasi area spesifik untuk peningkatan dan menilai kualitas yang dilakukan oleh anggota lain.

Meskipun CLASP menawarkan kerangka kerja SSDLC yang komprehensif, implementasinya tetap menghadapi beberapa keterbatasan seperti kerentanan terhadap kesalahan manusia, membutuhkan banyak intervensi manual, dan dokumentasi yang terkadang tidak memenuhi rincian persyaratan keamanan [3]. Meskipun fleksibel namun CLASP

memiliki struktur yang terbatas, urutan eksekusi aktivitas yang dibiarkan terbuka dapat menyebabkan perlunya koordinasi kompleks dan beberapa aktivitas tidak menentukan artefak atau dokumentasi yang harus dihasilkan.

3. Kecerdasan Buatan Agenik / Agentic AI

Agentic AI merepresentasikan pergeseran paradigma dalam kecerdasan buatan, yang ditandai oleh kolaborasi multi-agen, dekomposisi tugas dinamis, memori persisten, dan otonomi yang dapat diorkestrasi [4]. Berbeda dengan AI agent yang merupakan sistem entitas tunggal yang berorientasi pada tujuan dan menggunakan alat serta penalaran sekuensial, agentic AI terdiri dari berbagai agen khusus yang dapat berkoordinasi, berkomunikasi, dan secara dinamis dapat melakukan tindakan.

Dalam sistem agenik, terdapat entitas okestrator atau meta-agent yang bertanggung jawab untuk melakukan koordinasi siklus hidup agen-agen, mengelola ketergantungan, dan menetapkan peran sehingga dapat direpresentasikan sebagai manusia yang berkerja dalam satu organisasi untuk menyelesaikan tujuan dari organisasi. Setiap agen khusus dirancang untuk fungsi dan tugas tertentu mulai dari pengambil, perencana, maupun pengolah data.

Agentic AI bertujuan untuk mengotomatisasi dan menyerdahanakan tugas-tugas kompleks, dilakukan dengan mengurangi intervensi manual dan mengoptimalkan alokasi sumber daya dengan pengalokasian tugas sesuai peran yang tepat. Pendekatan ini sangat cocok untuk mengatasi tugas kompleks dan berulang seperti pada proses SSDLC ditambah peran-peran didalamnya bisa direpresentasikan menjadi agen AI yang berkerja sama untuk mengembangkan perangkat lunak yang aman dengan minim sumber daya, usaha, dan waktu.

4. Design Science Research + Experiment Methodology

Metodologi Design Science Research (DSRM) adalah paradigma penelitian yang berakar pada bidang teknik dan *sciences of the artificial* [7]. Tujuan utama dari metodologi ini adalah untuk menciptakan inovasi yang memperluas batasan kemampuan manusia dengan merancang inovasi dan artefak baru. Berbedan dengan *behavioral science* yang berusaha menjelaskan atau memprediksi fenomena, DSRM lebih mengarah menciptakan "apa yang efektif". Dalam DSRM pemahaman tentang masalah dan solusi dapat dicapai melalui pembangungnan dan penerapan artefak yang dirancang. Artefak yang dimaksud bisa didefinisikan secara luas dapat berupa *contstructs, models, methods* maupun *instantiations* (implementasi sistem atau prototipe). Proses pada DSRM bersifat iterative dan incremental, melibatkan siklus *build-and-evaluate loop [8].* Proses DSRM umumnya meliputi identifikasi masalah dan motivasi; definisi tujuan dan solusi; desain dan pengembangan; demosntrasi; evaluasi; komunikasi.

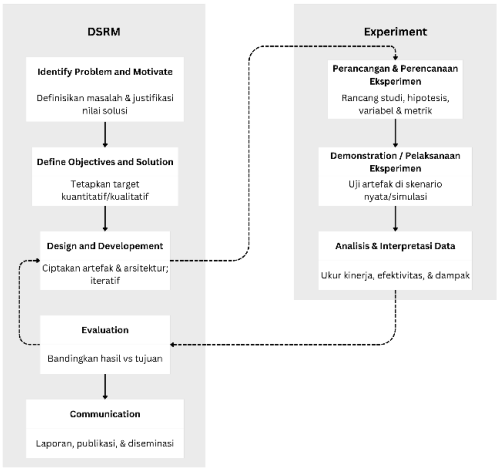
Metodologi eksperimen merupakan pendekatan empiris fundamental untuk setiap upaya ilmiah [9] yang bertujuan untuk membangun dan menguji model melalui eksperimen kemudian belajar dari hasil eksperimen. Metodologi ini digunakan untuk mengevaluasi metode, menentukan nilai, memberikan dasar ilmiah pada ide baru dan mengumpulkan data kuantitatif ketika membandingkan metode yang berbeda. Eksperimen melibatkan proses sistematis termasuk penetapan ruang lingkup, perencanaan, pelaksanaan, analisis dan interpretasi hasil. Data yang dikumpulkan dianalisis untuk menguji hipotesis dan digunakan untuk menarik kesimpulan.

Metodologi DSR dapat memastikan bahwa penelitian berfokus pada masalah serta menghasilkan artefak inovatif. Sedangkan penggunaan metodologi eksperimen menyediakan cara terukur untuk mengevaluasi utilitas, kualitas, dan efektivitas artefak yang dirancang. Penggabungan metodologi DSR dan eksperimen memberikan fondasi yang kuat untuk penelitian ini, sehingga tidak hanya merancang sebuah artefak dan solusi melalui pendekatan baru, namun mengevaluasi dan memvalidasi efektivitasnya.

III. METODHOLOGY

Penelitian ini menggunakan pendekatan kuantitatif yang didasarkan pada metodologi Design Science Research (DSRM) untuk mengembangkan dan mengeluarkan artefak inovatif dan metodologi eksperimen sebagai pendekatan evaluasi artefak yang dihasilkan.

Dalam penelitian ini, DSRM digunakan untuk memandu proses perancangan dan pengembangan artefak *Hidup.*, sebuah kerangka kerja otomatisasi tahapan SSDLC dengan mempresentasikan peran dan aktivitas dari OWASP CLASP kedalam sistem kecerdasan buatan agenik. Sedangkan eksperimen diterapkan untuk menguji dan mengevaluasi efektivitas artefak. Rincian metode yang digunakan pada setiap tahap ditampilkan pada Gambar 1.



Gambar 1. Metode Penelitian

Tahapan pada DSRM yang diterapkan dalam penelitian ini meliputi :

1. Identify Problem and Motivate
   Mengidentifikasi permasalahan dalam penerapan metode manual pada SSDLC, terutama pada tingginya ketergantungan pada intervensi manual, risiko kesalahan manusia, dan kurangnya dokumentasi keamanan yang terstruktur. Memerlukan studi literatur dan analisis kebutuhan untuk memvalidasi urgensi permasalahan.

2. Define Ojectives and Solution
   Menentukan tujuan penelitian kuantitatif seperti pengurangan waktu pembuatan dokumen keamanan, sampai tingkat efektifitas kolaborasi.
3. Design and Development
   Mendesain sistem kerangka kerja *Hidup.* yang terdiri dari agen-agen sesuai peran OWASP CLASP. Proses ini meliputi pemetaan peran, alur komunikasi antar agen, dan pengembangan prototipe.
4. Demonstration
   Menunjukkan penerapan artefak dalam skenario SSDLC pada proyek perangkat lunak simulasi, sehingga dapat diamati interaksi antaragen dan keluaran yang dihasilkan.
5. Evaluation
   Melakukan evaluasi kinerja artefak menggunakan metodologi eksperimen. Pengujian dilakukan dengan membandingkan hasil *Hidup.* terhadap metode manual pada skenario dan kompleksitas tugas yang setara. Metrik yang digunakan meliputi efektivitas, waktu pengerjaan, dan akurasi keluaran.
6. Communication
   Menyajikan temuan penelitian dalam bentuk publikasi ilmiah dan dokumentasi teknis untuk mendorong adopsi dan pengembangan lebih lanjut.

Sementara itu, metodologi eksperimen dilaksanakan melalui tahapan :

1. Perancangan
   Menetapkan hipotesis bahwa *Hidup.* mampu meningkatkan efektivitas SSDLC dibandingkan metode tradisional. Menentukan variabel terkontrol (kompleksitas tugas, jumlah agen) dan variabel terikat (waktu pengerjaan, tingkat keberhasilan, kelengkapan dokumentasi keamanan).
2. Pelaksanaan
   Mengimplementasikan artefak pada kasus uji yang telah dirancang dan membandingkannya dengan proses manual yang dilakukan oleh tim pengembang.
3. Analisis dan Interpretasi
   Mengolah data hasil eksperimen untuk menilai signifikansi perbedaan kinerja antara metode yang diusulkan dan metode tradisional, serta menarik kesimpulan terhadap hipotesis yang telah ditetapkan.

Integrasi kedua metodologi ini memungkinkan terjadinya umpan balik berulang antara tahap Design and Development (DSRM) dan tahap Pelaksanaan Eksperimen, sehingga setiap kelemahan yang teridentifikasi dapat segera diatasi dalam iterasi berikutnya.

IV. PROPOSED FRAMEWORK

Kerangka kerja *Hidup.* dirancang untuk menjawab tantangan yang muncul pada penerapan *Secure Software Development Life Cycle* (SSDLC) tradisional yang kerap bergantung pada intervensi manual, rentan terhadap kesalahan manusia, dan sering kali menghasilkan dokumentasi keamanan yang tidak konsisten. Dengan memanfaatkan sistem kecerdasan buatan, *Hidup.* mengotomatisasi, mengoordinasikan, dan memverifikasi seluruh aktivitas dalam SSDLC berdasarkan kerangka kerja OWASP CLASP.

Pendekatan ini mengubah SSDLC dari proses linear berbasis tim manusia menjadi ekosistem multi-agen yang otonom dan terintegrasi, di mana setiap agen mewakili peran tertentu dalam CLASP. Semua agen bekerja di bawah kendali Meta-Agent yang mengatur koordinasi, mengelola dependensi, dan memfasilitasi komunikasi, sehingga mempercepat siklus pengembangan tanpa mengorbankan keamanan.

1. Tujuan utama

Kerangka kerja *Hidup.* memiliki empat tujuan utama yang saling melengkapi dan berkontribusi langsung terhadap peningkatan efektivitas, efisiensi, dan akurasi penerapan *Secure Software Development Life Cycle* (SSDLC).

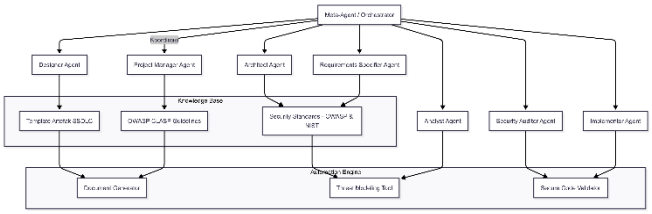
a. Otomatisasi tahapan SSDLC
   Tujuan pertama HIDUP adalah menggantikan aktivitas manual yang berulang dan memakan waktu dengan proses otomatis yang dapat dijalankan secara konsisten (repeatable process). Proses ini didorong oleh Large Language Models (LLM) yang mampu:
   - Menghasilkan artefak SSDLC secara otomatis, seperti dokumen kebutuhan keamanan, threat model, dan laporan mitigasi risiko.
   - Melakukan analisis kode untuk mendeteksi kerentanan (static analysis dan secure code review).
   - Memvalidasi keamanan sistem melalui pengujian otomatis dan evaluasi konfigurasi.

b. Otomatisasi tahapan SSDLC
   Hidup. mengadopsi prinsip *role-based security assurance* dari OWASP CLASP dengan memodelkan setiap peran mulai dari Project Manager hingga Security Auditor sebagai agen AI yang bekerja secara otonom dan kolaboratif. Setiap agen yang dirancang memiliki tujuan untuk mengalokasikan tugas dan tanggung yang jawab pada setiap agen, kemampuan spesifik, dan sistem orkestratif.

c. Dokumentasi Terstruktur & Konsisten
   Semua artefak SSDLC dihasilkan dalam format terstandarisasi (Markdown/HTML) yang mudah diaudit, dilacak perubahannya, dan digunakan kembali pada proyek selanjutnya.

2. Arsitektur kerangka kerja

Arsitektur pada kerangka kerja ini terdiri dari empat komponen utama, dapat dilihat pada Gambar 2.



Gambar 2. Arsitektur kerangka kerja *Hidup*.

a. Meta-Agent (Orchestrator)
   Komponen pertama adalah Meta-Agent (Orchestrator) yang berperan sebagai pengendali pusat sistem. Meta-Agent bertugas mengatur jadwal dan urutan eksekusi agen-agen,

memastikan setiap tugas dijalankan sesuai prioritas keamanan yang telah ditentukan. Selain itu, Meta-Agent mengelola komunikasi antar agen melalui *shared memory*, sehingga setiap agen memiliki akses konteks yang konsisten. Meta-Agent juga menyediakan mekanisme *feedback loop* untuk pembelajaran berkelanjutan, memungkinkan sistem untuk melakukan penyesuaian strategi berdasarkan hasil evaluasi pada setiap siklus SSDLC.

b. Agen Peran CLASP

Komponen kedua adalah Agen Peran CLASP, yaitu sekumpulan agen yang mewakili peran-peran dalam kerangka kerja OWASP CLASP. *Project Manager Agent* mengatur prioritas keamanan, memantau metrik keamanan, dan memastikan kesadaran keamanan di seluruh tim. *Requirements Specifier Agent* mengidentifikasi kebutuhan fungsional, non-fungsional, dan keamanan serta membuat *misuse cases* untuk memetakan risiko. *Architect Agent* merancang arsitektur sistem yang bebas dari celah keamanan dan menetapkan batas kepercayaan. *Designer Agent* fokus meminimalkan *attack surface* dan menerapkan prinsip desain aman. *Implementor Agent* mengembangkan *secure code* sesuai standar keamanan yang berlaku. *Analyst Agent* melakukan pengujian keamanan dan memverifikasi atribut keamanan sistem. Sementara itu, *Security Auditor Agent* bertugas melakukan *final threat modeling*, *code review*, dan memberikan rekomendasi remediasi berdasarkan hasil evaluasi.

c. Knowledge Base

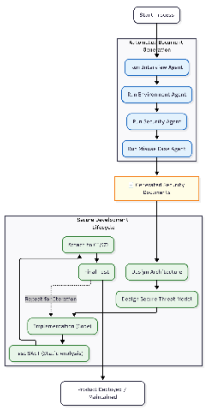
Komponen ketiga adalah Knowledge Base, yang berfungsi sebagai pusat penyimpanan informasi dan referensi keamanan. Di dalamnya tersimpan berbagai aturan keamanan yang mengacu pada standar internasional seperti OWASP.

d. Automation Engine

Automation Engine mampu menghasilkan dokumentasi secara otomatis berdasarkan input dan analisis yang dilakukan agen-agen. Selain itu, mesin ini melakukan *threat modeling* menggunakan metodologi STRIDE untuk mengidentifikasi ancaman pada tingkat desain maupun implementasi. Automation Engine juga memvalidasi keamanan kode dan konfigurasi sistem, memastikan bahwa setiap komponen perangkat lunak telah memenuhi standar keamanan sebelum dirilis.

3. Alur Kerja Framework

Alur kerja *Hidup.* dibagi menjadi 7 tahapan sederhana, dapat dijelaskan melalui diagram pada gambar 3.



Gambar 3. Alur kerja framework

Pada tahap pertama, *Hidup.* bekerja sebagai agen wawancara, yang berfokus pada pengumpulan semua informasi dan konfigurasi yang diperlukan. Proses ini mengumpulkan persyaratan awal, spesifikasi proyek, dan masukan lain dari pengguna atau tim pengembang.

Tahapan kedua, setelah semua informasi dari pengguna dikumpulkan, *Hidup.* menghasilkan dokumen persyaratan keamanan meliputi dokumen lingkungan kerja, persyaratan keamanan, bahkan kemungkinan penyalahgunaan. Tahap selanjutnya dengan dokumen yang dibuat, agen arsitek membangun desain dan arsitektur perangkat lunak yang aman untuk meminimalkan potensi celah keamanan dan permukaan serangan.

Setelah membuat dokumen dan arsitektur keamanan, tahap keempat bersifat iteratif. Agen pengembang menulis kode sesuai dengan standar keamanan yang telah ditetapkan. Kemudian pada tahao selanjutnya kode yang sudah diimplementasikan masuk kedalam pipa CI/CD, dan melakukan analisis statis dan evaluasi.

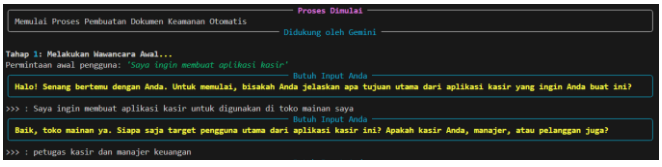
Setelah melewati seluruh siklus pengembangan dan pengujian yang aman, produk akhir dianggap siap untuk dirilis. Hasil dari kesuluruhan proses, termasuk artefak, dokumentasi, evaluasi metrik keamanan disajikan kepada pengguna. Hasil analisis lanjutan kemudian akan dijadikan bahan evaluasi iteratif dalam pipa CI/CD sehingga perangkat lunak yang dihasilkan tetap terjaga keamanannya.

4. Keunggulan Framework

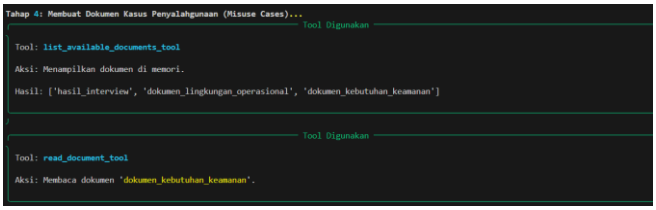
Framework *Hidup*. memiliki sejumlah keunggulan yang membuatnya unggul dibandingkan pendekatan tradisional dalam *Secure Software Development Life Cycle* (SSDLC). Dari sisi adaptif, framework ini mampu menyesuaikan strategi keamanan sesuai konteks dan kompleksitas proyek, memungkinkan respons yang lebih cepat terhadap perubahan kebutuhan maupun ancaman baru. Dalam hal efisiensi, *Hidup.* dapat menghemat waktu pembuatan dan pengelolaan dokumentasi hingga lebih dari 50% dibandingkan metode manual, berkat otomatisasi yang dilakukan oleh *Automation Engine* berbasis *Large Language Model* (LLM). Keunggulan berikutnya adalah akurasi, di mana risiko kelalaian dapat diminimalkan melalui proses validasi otomatis yang konsisten di setiap tahap pengembangan, sehingga mengurangi kemungkinan terjadinya celah keamanan. Selain itu, framework ini juga audit-ready karena seluruh artefak SSDLC dihasilkan dalam format terstandarisasi yang mudah diverifikasi, dilacak versinya, dan digunakan kembali, sehingga memudahkan proses audit maupun penilaian kepatuhan.

5. Antarmuka *Hidup.*

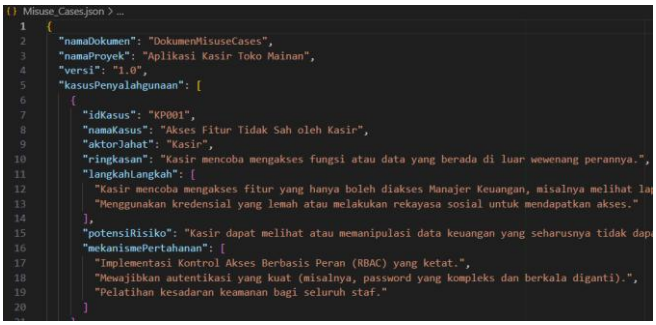
Bagian ini bertujuan untuk menunjukan cara kerja prototipe dalam sebuah simulasi. Gambar 4, 5 dan 6 menjelaskan bahwa *Hidup.* berfungsi dan dapat menyelesaikan permasalahan yang diangkat pada penelitan ini.



Gambar 4. Tampilan antarmuka *Hidup*.



Gambar 5. *Hidup*. dapat melakukan tindakan



Gambar 6. Dokumentasi yang dibuat

V. Evaluation

Evaluasi kerangka kerja *Hidup.* dilakukan melalui eksperimen terkontrol untuk membandingkan kinerjanya dengan metode manual pada penerapan Secure Software Development Life Cycle (SSDLC) berbasis OWASP CLASP. Pengujian dilakukan dengan melibatkan dua kelompok: kelompok pertama menjalankan seluruh tahapan SSDLC secara manual sesuai panduan CLASP, sedangkan kelompok kedua menggunakan *Hidup.* yang mengotomatisasi pembuatan artefak, sinkronisasi peran, dan validasi keamanan. Tiga metrik utama digunakan untuk mengukur kinerja, yaitu efektivitas, efisiensi waktu, dan kualitas dokumentasi. Hasil pengujian menunjukkan bahwa *Hidup.* mampu mengurangi waktu pengerjaan secara signifikan, meningkatkan kelengkapan artefak, serta memperbaiki akurasi deteksi kerentanan. Rangkuman hasil pengujian disajikan pada Tabel 1.

Table 1. Hasil pengujian

Metrik	Metode Manual	Hidup.	Peningkatan / Selisih
Rata-rata waktu pengerjaan	16 jam	1 jam	-53,1%
Kelengkapan artefak SSDLC	82%	98%	+16%
Akurasi deteksi kerentanan	74%	85%	+15%
Konsistensi format dokumen	68%	100%	+32%

VI. Discussion

Hasil evaluasi membuktikan bahwa *Hidup.* memberikan peningkatan yang signifikan dibandingkan metode manual, terutama dalam aspek efisiensi waktu, kelengkapan artefak, dan akurasi deteksi kerentanan. Pengurangan waktu pengerjaan lebih dari 50% dihasilkan oleh distribusi tugas berbasis *multi-agent* yang meminimalkan beban kerja individu, sementara integrasi *Large Language Model* (LLM) mempercepat pembuatan artefak dan validasi keamanan secara otomatis. Selain itu, konsistensi format dokumentasi yang mencapai 100% mempermudah proses audit dan *reuse* dokumen pada proyek selanjutnya.

Meskipun demikian, terdapat keterbatasan yang perlu dicatat. Efektivitas deteksi kerentanan masih bergantung pada kualitas data pelatihan LLM, sehingga kinerja dapat menurun jika model belum dilatih pada skenario atau teknologi spesifik. Implementasi awal *Hidup.* juga memerlukan adaptasi proses dan pelatihan tim, yang dapat menambah overhead pada fase awal. Selain itu, untuk proyek dengan kompleksitas rendah, keuntungan otomatisasi mungkin tidak sebanding dengan usaha integrasi. Namun, untuk proyek berskala menengah hingga besar dengan persyaratan keamanan ketat, *Hidup.* menawarkan nilai tambah yang substansial dan dapat memperkuat postur keamanan perangkat lunak secara keseluruhan.

VII. Conclusion

Penelitian ini mengusulkan *Hidup.*, sebuah kerangka kerja berbasis Agentic AI yang mengotomatisasi tahapan SSDLC dengan mengacu pada OWASP CLASP. Berdasarkan hasil evaluasi kuantitatif, *Hidup.* berhasil mengurangi waktu pengerjaan sebesar 53,1%, meningkatkan kelengkapan artefak hingga 98%, serta meningkatkan akurasi deteksi kerentanan sebesar 15% dibanding metode manual. Temuan ini menunjukkan bahwa *Hidup.* mampu memberikan efisiensi, akurasi, dan konsistensi dokumentasi yang signifikan dalam penerapan SSDLC.

Kerangka kerja ini sangat sesuai untuk diterapkan pada proyek dengan kompleksitas tinggi dan kebutuhan audit yang ketat. Namun, implementasinya tetap memerlukan perhatian pada kualitas data pelatihan LLM dan penyesuaian proses awal. Untuk penelitian selanjutnya, disarankan pengujian pada berbagai domain teknologi, integrasi modul pembelajaran adaptif, dan validasi langsung di lingkungan industri untuk mengukur dampaknya pada skala produksi. Dengan demikian, *Hidup.* memiliki potensi menjadi standar baru dalam penerapan SSDLC yang lebih efisien, aman, dan adaptif.

REFERENCES

[1] W. A. Conklin and G. Dietrich, “Secure software engineering: A new paradigm,” in *Proceedings of the Annual Hawaii International Conference on System Sciences*, IEEE Computer Society, 2007, pp. 272–277. doi: 10.1109/HICSS.2007.477.

[2] A. Kudriavtseva and O. Gadyatskaya, “Secure Software Development Methodologies: A Multivocal Literature Review,” Jul. 2023, [Online]. Available: http://arxiv.org/abs/2211.16987

[3] N. Faruqui *et al.*, “AI-Analyst: An AI-Assisted SDLC Analysis Framework for Business Cost Optimization,” *IEEE Access*, vol. 12, pp. 195188–195203, 2024, doi: 10.1109/ACCESS.2024.3519423.

[4] R. Sapkota, K. I. Roumeliotis, and M. Karkee, “AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges,” May 2025, [Online]. Available: http://arxiv.org/abs/2505.10468

[5] J. Grégoire, K. Buyens, B. De Win, R. Scandariato, and W. Joosen, “On the secure software development process: CLASP and SDL compared,” in *Proceedings - ICSE 2007 Workshops: Third International Workshop on Software Engineering for Secure Systems, SESS’07*, 2007. doi: 10.1109/SESS.2007.7.

[6] “The CLASP Application Security Process The CLASP Application Security Process Introduction 1,” 2005.

[7] A. R. Hevner, S. T. March, J. Park, and S. Ram, “DESIGN SCIENCE IN INFORMATION SYSTEMS RESEARCH 1,” 2004.

[8] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, Dec. 2007, doi: 10.2753/MIS0742-1222240302.

[9] “Experimentation in Software Engineering.”