BAB I PENDAHULUAN

1.1. Latar Belakang

Security telah menjadi aspek penting dalam perkembangan perangkat lunak [1], karena pesatnya perkembangan perangkat lunak baru, menyebabkan masalah *security* menjadi semakin kompleks [2]. Sebuah perangkat lunak perlu memastikan keamanan terhadap perangkat lunaknya, dan kegagalan dalam mengatur keamanan untuk perangkat lunak dapat mengakibatkan kerugian finansial, kerusakan lingkungan, hingga membahayakan manusia[3] [1]. Mempertimbangkan keamanan pada tahap awal *software development life cycle* (SDLC) dapat meningkatkan kesadaran terhadap potensi ancaman pada perangkat lunak, dan berpotensi mengurangi kerentanan keamanan pada post development [4] [5] [6]. *Security requirements engineering* (SRE) adalah sebuah proses melakukan elisitasi, analisis, dan dokumentasi security requirement pada sebuah sistem perangkat lunak, SRE bertujuan untuk memastikan *confidentiality*, *availability*, dan *integrity* perangkat lunak pada tahap awal pengembangan [5] [2]. Melakukan SRE dapat melibatkan berbagai teknik *threat modeling* untuk melakukan *threat analysis* [7].

Threat modeling telah digunakan secara luas untuk menganalisis potensi serangan atau ancaman terhadap aset berharga sistem [7]. Threat modeling memiliki berbagai metode yang dapat diadopsi, seperti *attack tree* dan *fault tree* [7]. Security Threat Oriented Requirement Engineering (STORE) adalah metodologi security requirement engineering (SRE) yang menggunakan *threat modeling* sebagai bagian dari prosesnya [8]. Dalam STORE, Security Attack Analysis (SAA) adalah kegiatan di mana engineer harus melakukan *threat modeling* untuk mengungkap potensi sumber serangan [8]. Ketika melakukan pemodelan ancaman, terutama pada tahap awal pengembangan, terdapat kemungkinan risiko kesalahan manusia dalam melakukan analisis, yang dapat mengakibatkan hasil analisis yang cacat. Kemungkinan, hal tersebut dapat terjadi karena kurangnya pemahaman *engineer* terhadap definisi

kebutuhan, dan kurangnya pengetahuan atau pengalaman di bidang keamanan sistem [5]. Oleh karena itu, diperlukan solusi untuk mendukung para *engineer* dalam melakukan SAA.

Kemunculan LLM telah mengubah arus pembentukan proses di banyak bidang. LLM telah berkontribusi secara signifikan di berbagai domain, membantu memecahkan masalah spesifik domain. Misalnya, Islam dkk. mengintegrasikan LLM ke dalam proses yang diusulkan untuk mengatasi kerentanan keamanan kode dalam konteks *code repair* [9], dan Ning dkk. Mengintegrasikan LLM ke dalam *attack framework* baru mereka untuk menyerang *recommendation system* berbasis LLM dalam pengaturan black box untuk mempelajari kerentanan keamanannya [10]. Dengan potensi LLM yang luar biasa, menggabungkan LLM untuk mendukung para *engineer* dalam melakukan SAA akan menjadi solusi yang menjanjikan. Oleh karena itu, Security Attack Analysis Support (SAAS) diusulkan untuk mendukung para *engineer* dalam melakukan SAA, dengan tujuan dari proses nya yaitu melakukan SAA berdasarkan suatu ruang lingkup proyek, dan SAAS mengintegrasikan LLM untuk menjadi otak dalam menjalankan SAA. Dalam penelitian ini, kami menggunakan pendekatan LLM berbasis *multi-agent* untuk proses SAAS, karena mengandalkan *single-agent* LLM untuk menangani seluruh proses SAA dan tugas tambahan lainnya dapat memberatkan LLM, karena memerlukan sebuah kemampuan analisa yang mendalam terkait kemungkinan serangan terhadap sebuah proyek [8]. Oleh karena itu, kami menggunakan pendekatan *multi-agent* untuk mengintegrasikan LLM ke dalam proses SAAS dengan lebih baik. Namun, terlepas dari pendekatan kami, diketahui bahwa mengintegrasikan LLM ke dalam proses tertentu akan menimbulkan tantangan yang harus diatasi.

1.2. Rumusan Masalah

Mengintegrasikan LLM ke dalam proses suatu bidang yang spesifik, memerlukan pertimbangan terhadap permasalahan yang dimiliki oleh LLM itu sendiri. Hal ini penting untuk dipertimbangkan karena mengatasi masalah

LLM seperti halusinasi, merupakan masalah penting yang perlu dihadapi agar LLM tidak menghasilkan informasi yang salah. Contohnya, Nouri et al. Yang mengalami halusinasi pada *LLM-based* HARA yang mereka kembangkan, yang menyebabkan inferensi dari LLM tersebut menghasilkan informasi yang salah [11]. Selain itu, Wang et al. menyatakan bahwa LLM dapat menghasilkan halusinasi, sehingga dengan mengurangi halusinasi pada LLM dapat meningkatkan kualitas keluaran LLM [12]. Dalam penelitian ini, LLM pada SAAS dapat dikatakan memiliki tingkat halusinasi yang tinggi, jika keluaran SAAS menghasilkan hasil SAA yang tidak sesuai dengan ruang lingkup proyeknya. Oleh karena itu, agar LLM dapat diintegrasikan secara efektif dengan SAAS, sangat penting untuk mengatasi masalah halusinasi LLM terlebih dahulu.

Untuk mengintegrasikan LLM menggunakan pendekatan *multi-agent* ke dalam proses yang spesifik untuk domain tertentu, perlu dilakukan penyusunan strategi dalam dekomposisi tugas, penentuan *agent* LLM yang akan terlibat, dan membuat *agent* LLM dapat berperilaku sesuai dengan yang telah diperintahkan, sehingga *agent* dapat melaksanakan tugas seefektif mungkin. Selain itu, perlu dipastikan juga bahwa proses SAAS yang telah terbentuk, dapat melakukan SAA secara efektif yaitu dengan melakukan analisis sesuai dengan cakupan proyek. Oleh karena itu, kami mengajukan pertanyaan penelitian untuk memandu penelitian kami dalam mengidentifikasi proses yang efektif untuk membentuk proses SAAS:

PP1: Bentuk proses apa yang efektif untuk menggabungkan LLM dengan SAA?

PP2: Seberapa efektif proses SAAS yang diusulkan?

Pertanyaan penelitian pertama akan berfokus pada mencari tahu proses SAAS seperti apa yang dapat secara efektif menggabungkan proses SAA dengan LLM. Pertanyaan penelitian kedua berfokus pada evaluasi efektivitas proses SAAS, untuk mencari tahu kemampuan SAAS dalam melakukan SAA yang

sesuai dengan konteks ruang lingkup suatu proyek, serta mencari kekuatan dan kelemahannya proses.

1.3. Tujuan dan Manfaat

Penelitian ini bertujuan untuk mengidentifikasi sebuah proses SAAS yang efektif untuk menggabungkan LLM dengan proses SAA agar dapat membentuk sebuah proses SAAS yang dapat melakukan SAA yang sesuai dengan cakupan suatu proyek. Penelitian ini bermanfaat untuk membantu para engineer dalam melakukan SAA.

1.4. Batasan Masalah

Dalam penelitian ini terdapat beberapa batasan masalah yang telah ditentukan, hal itu dibuat guna memperjelas ruang lingkup masalah yang terdapat dalam penelitian ini. Berikut batasan-batasan masalah dalam penelitian ini:

1. Dalam penelitian ini, konteks pengukuran efektivitas proses SAAS, akan dilakukan terhadap sisi kefektifan SAAS dalam melakukan SAA, bukan mengukur sisi usability dari proses SAAS.

1.5. Jadwal Pelaksanaan

Kegiatan pengerjaan tugas akhir ini akan dilakukan dalam rentang waktu maksimal 7 bulan, dengan tahapan pengerjaan yang dilakukan secara sistematis. Tahapan pengerjaan TA dimulai dari merancang solusi berupa proses SAAS, Evaluasi proses SAAS dengan para ahli, dan menyusun laporan TA.

Tabel 1: Jadwal Pelaksanaan

No.	Deskripsi Tahapan	Bulan 1	Bulan 2	Bulan 3	Bulan 4	Bulan 5	Bulan 6
1	Merancang Solusi						
2	Evaluasi Proses Dengan Para Ahli						
3	Menyusun Laporan TA						