BAB I

PENDAHULUAN

1.1 Latar Belakang

Secara umum aplikasi web dibangun menggunakan arsitektur monolitik, dimana seluruh komponen sistem seperti antarmuka pengguna, dan basis data digabungkan dalam satu kesatuan aplikasi (Elgheriani et al., 2022). Perkembangan pesat teknologi informasi telah mendorong perubahan menuju arsitektur layanan berbasis web yang lebih modular dan terdistribusi. Pengembangan web berbasis layanan framework PHP seperti *Laravel* dan *Codeigniter* banyak digunakan karena strukturnya yang modular dan mudah diintegrasikan (Fauzan Praseyto Eka Putra et al., 2025). Fitur *Laravel* yang lengkap serta dukungan alat optimasi tingkat lanjut menjadikannya framework yang ideal untuk pengembangan aplikasi berskala besar dan kompleks, sedangkan kesederhanaan serta efisiensi yang dimiliki *Codeigniter* membuatnya unggul sebagai solusi dengan kinerja tinggi untuk aplikasi yang ringan (Ahmed et al., 2024).

Kedua framework yang dibuat dengan fungsi *CRUD* tersebut mendukung penerapan *Service-Oriented Architecture,* implementasi *CRUD endpoint* dalam *RESTful API* dapat dipandang sebagai bentuk konkret dari penerapan prinsip SOA yang mendukung modularitas, skalabilitas, dan reusabilitas layanan (Parab et al., 2022). Pendekatan SOA semakin populer karena mampu meningkatkan fleksibilitas dan efisiensi pengembangan sistem. Salah satu implementasi utamanya yaitu web *service* berbasis *Restful API,* yang memiliki waktu akses lebih singkat dan proses integrasi yang lebih sederhana dibandingkan arsitektur monolitik

(Samuel & Girsang, 2020). *Restful API* sendiri merupakan gaya arsitektur layanan web yang memanfaatkan protokol HTTP serta format data seperti *JSON* atau *XML* untuk pertukaran data (Widodo Purbo, 2021). *JSON* menjadi standar utama karena mendukung struktur data berbentuk objek dan *array,* sehingga memudahkan proses pengolahan data. *Restful API* dapat diintegrasikan dengan berbagai framework karena standart komunikasi *Restful API* yaitu HTTP, tanpa penyesuaian yang kompleks (Muharom et al., 2025). Penerapan *Restful API* juga menghadirkan tantangan baru dalam aspek keamanan. Setiap web yang dibangun dengan banyak layanan *service* memiliki potensi kerentanan yang berbeda, menjadikan *API* sebagai salah satu sasaran utama bagi penyerang di berbagai platform. Berbagai insiden keamanan telah terjadi dalam beberapa tahun terakhir, seperti kerentanan pada *API* Facebook yang mengekspos jutaan data pribadi pengguna (Du et al., 2024).

Pentingnya untuk mengidentifikasi kerentanan web service sejak dini guna melindungi dari serangan siber yang terus berkembang seiring kemajuan teknologi (Altulaihan et al., 2023). Salah satu langkah pengamanan yang banyak diterapkan untuk melindungi aplikasi web digunakanya *Web Application Firewall*. *WAF* yang bersifat open-source dan popular yaitu *ModSecurity,* yang berfungsi memantau, menganalisis, dan memfilter lalu lintas HTTP (Sobola, 2020). Kerentanan umum yang sering terjadi pada web *Restful API* meliputi *Broken Object Level Authorization, Broken Authentication, Broken Object Property Level Authorization, Unrestricted Resource Consumption, Broken Function Level Authorization, Unrestricted Access to Sensitive Business Flows, Server Side Request Forgery,*

Cross-Site Request Forgery, Security Misconfiguration, Improper Inventory Management dan *Unsafe Consumption of APIs* (OWASP *API* Top 10, 2023) maka dari itu, menganalisis keamanan framework berbasis *Restful API* seperti *Laravel* dan *Codeigniter*, digunakan metode *Vulnerability Assessment and Penetration Testing (VAPT)* serta pengujian penerapan *Modsecurity* sebagai lapisan keamanan tambahan. Melalui metode *VAPT,* penguji mensimulasikan serangan guna mengidentifikasi dan mengeksploitasi kelemahan sistem sehingga pengembang dapat melakukan perbaikan sebelum aplikasi dipublikasikan (Mohan et al., 2021)

Penelitian terkait yang telah dilakukan pada penelitian sebelumnya menjelaskan hasil dari penerapan *Service-Oriented Architecture (SOA)* mampu meningkatkan efisiensi komunikasi antar layanan dan mengurangi waktu akses dibandingkan arsitektur monolitik (Elgheriani et al., 2022) Framework PHP seperti *Laravel* dan *Codeigniter* yang menerapkan prinsip *Service-Oriented Architecture (SOA)* melalui layanan berbasis *endpoint*. Implementasi *CRUD endpoint* dalam *RESTful API* menjadi bentuk konkret penerapan SOA (Parab et al., 2022). *Laravel* dengan fitur lengkap dan alat optimasinya cocok untuk aplikasi kompleks dan skalabel, sedangkan *Codeigniter* yang sederhana dan efisien unggul untuk aplikasi ringan berkinerja tinggi (Ahmed et al., 2024). *RESTful API* adalah gaya arsitektur layanan web yang menggunakan protokol HTTP dan format data seperti *JSON* atau *XML* sebagai media untuk melakukan pertukaran informasi (Widodo Purbo, 2021). Menyoroti meningkatnya ancaman keamanan terhadap *Restful API,* karena setiap web yang dibangun dengan banyak layanan *service* memiliki potensi kerentanan yang berbeda, di mana banyak serangan *seperti SQL Injection, Cross-Site Scripting*

(XSS), dan *Cross-Site Request Forgery (CSRF)* terjadi akibat kurangnya pengujian keamanan pada *API* public (Du et al., 2024) (Altulaihan et al., 2023). Penerapan *Modsecurity* terhadap keamanan web, mampu mengamankan dari serangan siber (Sobola, 2020).

Meskipun web sudah diterapkan WAF yang mampu memblok otomatis serangan siber, pentingnya melakukan penilaian kerentanan serta simulasi serangan sebelum layanan berbasis web dipublikasikan, guna mengurangi risiko eksploitasi celah keamanan (Ravindran & Potukuchi, 2022). Disimpulkan metode *Vulnerability Assessment and Penetration Testing (VAPT)* merupakan teknik modern yang efektif untuk mengidentifikasi kelemahan sistem serta pengujian penerapan *Modsecurity* sebagai lapisan keamanan secara komprehensif melalui simulasi serangan (Mohan et al., 2021).

Berdasarkan permasalahan dan penelitian yang pernah dilakukan sebelumnya. Penelitian ini mengusulkan perbandingan keamanan antara framework *Laravel* dan *Codeigniter* yang dilapisi keamanan *Modsecurity* untuk mengidentifikasi kerentanan dalam pengembangan *REST API* sehingga para pengembang *Restful API* dapat mengetahui skema apa saja yang perlu dipersiapkan dalam pemilihan framework supaya terhindar dari serangan siber.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam penelitian ini yaitu :

1. Bagaimana tingkat keamanan framework *Laravel* dan *Codeigniter* pada *Rest API* jika diuji menggunakan metode *Vulnerability Assessment and Penetration Testing?*
2. Bagaimana pengaruh penerapan keamanan *default* dan *Modsecurity* terhadap peningkatan keamanan framework *Laravel* dan *Codeigniter* berbasis *Rest API* berdasarkan hasil pengujian menggunakan metode *Vulnerability Assesment* dan *Penetration Testing ?*

1.3 Tujuan Penelitian

Berdasarkan latar belakang masalah dan rumusan masalah, maka tujuan dari penelitian ini adalah :

1. Penelitian ini bertujuan untuk menganalisis tingkat kerentanan framework *Laravel* dan *Codeigniter* berbasis *Rest API* dengan menggunakan metode *Vulnerability Assesment (VA)* dan *Penetration Testing (PT).*
2. Penelitian ini bertujuan untuk menganalisis pengaruh penerapan keamanan *default* dan *Modsecurity* terhadap peningkatan keamanan framework *Laravel* dan *Codeigniter* berbasis *Rest API* berdasarkan hasil pengujian *VAPT.*

1.4 Batasan Penelitian

Adapun batasan penelitian yang dilakukan adalah sebagai berikut :

1. Framework yang digunakan *Laravel* dan *Codeigniter.*
2. Keamanan WAF yang digunakan *Modsecurity* dengan versi branch v2 (2.9.12)
3. Penggunaan metode *Vulnerability Assesment* dan *Penetration Testing.*

4. Hasil pengujian kerentanan menggunakan *tools* Burp Suite, OWASP dan Post man.

5. Ruang lingkup penelitian berfokus pada komparasi mekanisme otentikasi serta fitur perlindungan terhadap *XSS, SQL Injection,* dan *CSRF* pada kedua framework.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat bermanfaat bagi seluruh pihak yang terkait, diantaranya :

1. Memahami berbagai masalah dan meningkatkan kesadaran pengguna digital.

2. Membantu praktisi keamanan dalam memilih framework berbasis PHP, untuk membuat strategi keamanan dalam pembuatan web *service.*

3. Menjadi referensi ilmiah bagi penelitian selanjutnya dalam bidang keamanan aplikasi berbasis *Rest API*, khususnya yang berfokus pada analisis dan perbandingan tingkat keamanan antar framework

4. Sebagai referensi bagi penelitian lain yang membahas keamanan dalam perbandingan framework.

5. Memberikan referensi bagi penelitian selanjutnya bahwa framework dengan pengaturan default masih tergolong kurang efektif dalam aspek keamanan.