BAB II

TINJAUAN PUSTAKA

2.1 Landasan Teori

2.2.1 *Service-Oriented Architecture*

Service-Oriented Architecture (SOA) merupakan gaya arsitektur yang membangun sistem kompleks melalui integrasi berbagai layanan yang saling terhubung. Struktur layananya mandiri dan dapat berinteraksi dengan sistem lain secara fleksibel. *SOA* telah banyak digunakan di industri dan diakui sebagai solusi efektif untuk mendukung berbagai standar sistem melalui satu atau lebih antarmuka layanan. Sifatnya modular dan fleksibel, *SOA* membuka peluang baru dalam pengembangan proses bisnis dan sistem manufaktur. Setiap komponen perangkat lunak dalam *SOA*, yang disebut layanan (*service*), berfungsi sebagai unit mandiri yang menjalankan tugas tertentu dan dapat digunakan kembali oleh aplikasi atau platform lain tanpa perlu mengetahui detail implementasinya (Giao et al., 2022).

2.2.2 *REST API*

Representational State Transfer (REST) merupakan gaya arsitektur yang umum digunakan dalam pengembangan web *service. RESTful API* memungkinkan sistem berbeda berkomunikasi satu sama lain, dimana *resource* dalam database dapat diakses melalui *endpoint API.* Untuk mendapatkan *resource, RESTful API* menggunakan perintah HTTP *request* (Kaniya Pradnya Paramitha et al., 2022). *REST* adalah arsitektur untuk membangun aplikasi web yang menggunakan komunikasi *stateless* melalui HTTP. *RESTful API* melakukan operasi CRUD (*Create, Read, Update, Delete)* pada sumber daya yang diakses melalui URI,

dengan interpretasi kode status HTTP yang dapat bervariasi tergantung pada *API* (Decrop et al., 2024).

2.2.3 Framework

Framework merupakan kerangka kerja pemrograman yang dirancang untuk digunakan kembali, sehingga programmer tidak perlu memulai dari nol atau menulis skrip yang sama untuk tugas yang serupa. Sebagai contoh, ketika seorang programmer ingin membuat halaman web dengan fitur paginasi, framework sudah menyediakan fungsi tersebut. Programmer cukup memanfaatkan fungsi tersebut saat melakukan pengkodean, dengan mengikuti pedoman yang ditetapkan masing-masing framework (Purnama Sari & Wijanarko, 2020).

Sehingga memudahkan programmer untuk memanggilnya tanpa harus menuliskan *syntax* program yang sama berulang-ulang serta dapat menghemat waktu (Sallaby & Kanedi, 2020). Tabel 2.1 merupakan fitur bawaan keamanan *Laravel* dan *Codeigniter. Laravel* menyediakan modul autentikasi bawaan, perlindungan *XSS* melalui Blade "{{}}", perlindungan *SQL injection* menggunakan *Query Builder* dengan parameter PDO, dan token *CSRF* otomatis untuk sesi pengguna. Sebaliknya, *Codeigniter* tidak memiliki pustaka autentikasi bawaan, tetapi menawarkan filter *XSS* otomatis, beberapa metode pengamanan SQL injection, dan penyisipan otomatis bidang *CSRF* ke formulir web (Adamu et al., 2020).

Tabel 2. 1 Keamanan *default* framework

Parameter	*Laravel*	*Codeigniter*
Modul Otentikasi	Dilengkapi dengan paket autentikasi yang siap pakai dan disederhanakan. Fasilitas autentikasi *Laravel* terdiri dari "penjaga" dan "penyedia".	*Codeigniter* secara default tidak menyertakan pustaka otentikasi.
Skrip Lintas Situs Perlindungan *(XSS)*	Pernyataan *blade "{{}}"* dikirim secara otomatis untuk mencegah serangan *XSS*.	*Codeigniter* memiliki filter *XSS* terintegrasi yang diinisialisasi secara otomatis.
Perlindungan Injeksi *SQL*	Pembuat kueri menggunakan parameter PDO yang memiliki dukungan bawaan untuk pernyataan yang disiapkan.	Menawarkan beberapa metode untuk menangani upaya injeksi berbahaya dalam database.
Pemalsuan Permintaan Lintas Situs *(CSRF)* Perlindungan	*Laravel* secara otomatis menghasilkan token *CSRF* untuk sesi pengguna setiap aplikasi.	Menyisipkan bidang *CSRF* tersembunyi kedalam formulir web secara otomatis.

Tabel 2.2 menyajikan analisis komparatif antara framework *Laravel* dan *Codeigniter* yang diklasifikasikan berdasarkan parameter popularitas, arsitektur sistem, serta kelengkapan fitur keamanan dan fungsionalitas (Muthia Kansha et al., 2023).

Tabel 2. 2 Kelebihan dan kekurangan framework

No	Parameter	Kelebihan / Kekurangan	*Laravel*	*Codeigniter*
1	Popularitas & Tren	*Laravel* lebih unggul secara signifikan dalam tren dan penggunaan global.	Menduduki posisi teratas dengan persentase penggunaan 67% (data *JetBrains* 2021). Memiliki 71.9 ribu Github *stars*.	Berada di posisi keempat dengan 9%. Memiliki 18.2 ribu Github *stars*.
2	Database *Support*	Keduanya memiliki dukungan database yang baik, namun *Codeigniter* memiliki dukungan *Oracle* dan *CUBRID/ODBC via PDO*.	Mendukung: *MariaDB, MySQL, PostgreSQL, SQLite, dan SQL Server*.	Mendukung: *MySQL, PostgreSQL, SQLite3, MSSQL, Oracle. Via PDO: CUBRID, Interbase/Firebird, ODBC.*

No	Parameter	Kelebihan / Kekurangan	*Laravel*	*Codeigniter*
3	*Programming Paradigm*	Perbedaan arsitektur mendasar (pilihan tergantung preferensi pengembang).	Menggunakan *Component Oriented Programming (COP).*	Menggunakan Object-*Oriented* dengan implementasi *Event Driven.*
4	*Routing*	*Codeigniter* menawarkan fleksibilitas dengan *Auto Routing*, sedangkan *Laravel* lebih terstruktur.	Memiliki 3 kategori: *route parameters, basic routing*, dan *named routes.*	Memiliki 2 jenis: *Defined Route Routing* (manual) dan Auto *Routing* (otomatis berdasarkan konvensi).
5	*Built-in Modules*	Kelebihan *Laravel*: Mendukung modularitas bawaan untuk penggunaan kembali kode.	Mendukung *Built-in Modules* yang memungkinkan reuse modul di berbagai *project.*	Tidak mendukung Built-in *Modules* (perlu *Modular Extension* tambahan).
6	*HTTPS Support*	Kelebihan *Laravel*: Keamanan HTTPS lebih terintegrasi dan otomatis.	Mendukung konfigurasi HTTPS Routes khusus dan menjamin keamanan data dengan	idak sepenuhnya mendukung HTTPS *Routes* (pengembang harus mengelola URL *Helper* secara manual).

No	Parameter	Kelebihan / Kekurangan	*Laravel*	*Codeigniter*
			menambahkan https:// secara otomatis.	
7	*Support RestAPI*	Kelebihan *Laravel*: Memiliki fitur bawaan khusus untuk pembuatan *API*.	Memiliki fitur *RESTful Controllers* untuk membangun *API REST* dengan mudah.	Tidak tersedia fitur khusus; pengembang harus menulis kode manual untuk membuat *API REST*.
8	*Template Engine*	Kelebihan *Laravel*: Blade Engine jauh lebih fleksibel (bisa menggunakan PHP) dibanding Parser CI.	Menggunakan *Blade* yang tidak membatasi penggunaan kode PHP biasa dalam template.	Tidak mengharuskan template engine, namun memiliki bawaan Parser yang terbatas (tidak boleh berisi PHP).
9	*Authentication*	Kelebihan *Laravel*: Fitur autentikasi sudah tersedia secara *built-in*.	Menyediakan kelas autentikasi bawaan untuk implementasi login/otorisasi.	Tidak memiliki fitur autentikasi bawaan (perlu menulis ekstensi khusus).
10	*Learning Curve*	Kelebihan *Codeigniter*: Lebih mudah dipelajari	Cenderung sulit dipahami pemula karena fitur yang banyak, namun	Lebih mudah dipahami pemula karena struktur yang sederhana dan *easy setup*.

No	Parameter	Kelebihan / Kekurangan	*Laravel*	*Codeigniter*
		oleh pemula dibanding *Laravel.*	memiliki clean architecture	
11	*Packages*	*Laravel* memiliki ekosistem paket resmi yang lebih banyak dan lengkap.	Memiliki banyak paket seperti: *Breeze.* Cashier, Passport, Jetstream, Sanctum, dll	Memiliki beberapa paket (khusus versi 4) seperti: Shield, Settings, Cache, DevKit.

2.2.4 Framework *Laravel*

Laravel merupakan sebuah framework web yang berbasis PHP atau juga sering disebut *open-source,* diciptakan oleh Taylor Otwell yang digunakan untuk pengembangan aplikasi web yang menggunakan pola MVC (*Model View Controller).* MVC pada *Laravel* sedikit berbeda pada umumnya, di *Laravel* terdapat *routing* yang menjembatani antara *request* dari *user* dan *controller.* Jadi *controller* tidak langsung menerima *request* tersebut (Bin Tahir et al., 2019).

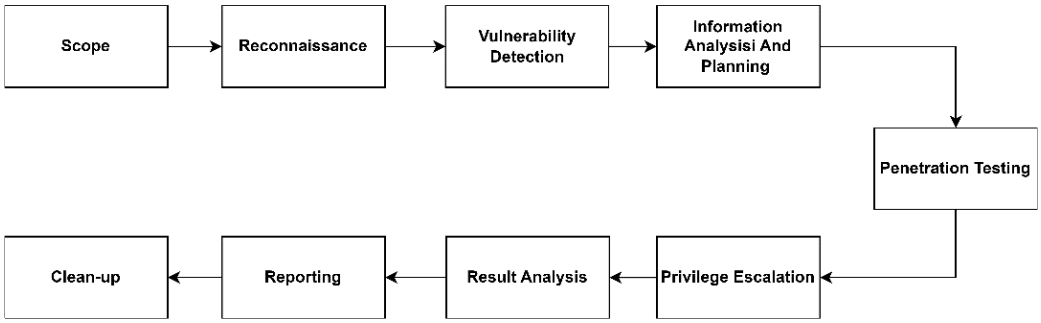
2.2.5 Framework *Codeigniter*

Codeigniter merupakan framework PHP dengan model MVC yang bisa mempercepat kerja *developer* dalam membuat aplikasi web selain ringan dan cepat, *Codeigniter* juga memiliki dokumentasi yang sangat lengkap. Penggunaan metode MVC pada framework *Codeigniter* merupakan bentuk *class* (Widodo Purbo, 2021).

Situs web resmi *Codeigniter (Codeigniter* ,2022) menyatakan framework *Codeigniter* dari bahasa pemrograman PHP tergolong tangguh dengan sedikit bug.

2.2.6 Vulnerability *Assessment and Penetration Testing*

VAPT merupakan sebuah metode untuk menemukan sebuah kerentanan terutama pada sebuah web, penilaian kerentanan berisi berbagai daftar kerentanan keamanan dengan melakukan pengecekan otomatis maupun manual secara menyeluruh tanpa adanya penyerangan yang nyata. Pengujian penetrasi berfokus pada simulasi serangan siber dan mencari tahu cara seorang penyerang sungguhan bisa menembus sistem menggunakan pintu yang mana. *Vulnerability Assesment* mendapatkan beberapa kerentanan dan *Penetration Testing* menguji, menvalidasi dari beberapa kerentanan yang dihasilkan itu valid ataupun tidak (Ravindran & Potukuchi, 2022). Metode VAPT memiliki sembilan tahapan, seperti yang ditunjukan pada gambar 2.1 yaitu *Scope, Reconnaissance, Vulnerability Detection, Information Analysis and Planning, Penetration Testing, Privilege Escalation, Result Analysis, Reporting, Clean-up* (Kour & Kaur, 2022).



Gambar 2. 1 Tahapan VAPT

1. *Scope,* proses menentukan bagian mana dari sistem yang akan diuji, ruang lingkup ini mencakup komponen atau area tertentu dari *infrastruktur,* aplikasi, jaringan yang akan menjadi target pengujian dan mendapatkan informasi tentang port sistem yang akan diuji.

2. *Reconnaissance,* proses mengumpulkan informasi tentang sistem atau target yang akan diuji. Informasi yang dikumpulkan mencakup data seperti IP *Addres, DNS, Url, Method* serta perangkat lunak yang digunakan.

3. *Vulnerability Detection,* proses mendeteksi kelemahan keamanan atau celah-celah pada target. Mencakup pemindaian otomatis maupun manual terhadap target untuk menemukan masalah keamanan seperti konfigurasi yang salah, atau layanan yang rentan terhadap serangan.

4. *Information Analysis And Planning,* proses melakukan analisis mendalam untuk membuat skenario atau merencanakan eksploitasi yang dilakukan pada kerentanan.

5. *Penetration Testing,* proses menjalankan skenario eksploitasi langsung dari kelemahan yang telah ditemukan sebelumnya. Pengujian ini bersifat manual atau otomatis.

6. *Privelege Escalation,* proses dilakukan pengujian pada website yang dikembangkan, pengujian ini dilakukan terhadap token yang tidak aman, pengaturan hak akses yang buruk dan kesalahan konfigurasi pada web tersebut.

7. *Result Analysis,* proses ini melakukan evaluasi hasil pengujian keamanan dan memahami seberapa besar dampak kerentanan *sistem* terhadap risiko yang dihasilkan.

8. *Reporting,* tahapan ini mencakup penyusunan laporan yang merangkum semua temuan, termasuk kerentanan yang diidentifikasi, eksploitasi yang dilakukan serta rekomendasi untuk perbaikan.

9. *Clean-up,* proses mengembalikan semua jejak yang ditinggalkan selama penetrasi, ke kondisi awal. Agar tidak ada kerusakan atau risiko tambahan yang ditimbulkan oleh pengujian.

2.2.7 Burp Suite

Burp Suite berdiri pada tahun 2003, Burp Suite didirikan oleh Dafydd Stuttard merupakan sekumpulan alat yang digunakan untuk mendukung pengujian penetrasi, mulai dari perencanaan hingga identifikasi dan eksploitasi kerentanan. *Tools* Burp Suite terdapat fitur *proxy* berfungsi sebagai vektor serangan *man-in-the-middle* dengan monitor lalu lintas antara browser dan aplikasi target serta sejumlah tes manual lainya. Burp Suite perlu dikonfigurasi secara manual sebagai *proxy* pada browser (Nagpure & Kurkure, 2017).

2.2.8 Postman

Postman didirikan pada tahun 2012, telah berkembang dari *ekstensi* Chrome sederhana menjadi *platform* pengembangan *API* yang lengkap. Postman menyediakan fitur untuk membuat, menguji dan mendokumentasikan *API*, serta digunakan oleh pengembang individu maupun tim besar dengan fungsionalitas

yang luas dan pengguna yang banyak, Postman kini menjadi standar dalam pengembangan *API* (Wei, 2024).

2.2.9 OWASP

OWASP adalah singkatan dari *Open Web Application Security Project,* sebuah organisasi yang berasal dari Amerika Serikat yang didirikan pada tahun 2004 (Alanda et al., 2020). OWASP merupakan salah satu alat untuk penilaian kerentanan dan tingkat risiko keamanan web, yang dikembangkan dengan tujuan menyediakan sumber informasi agar pengembang aplikasi dapat mempelajari sistem keamanan situs web dan meningkatkan keamananya.

2.2.10 *Modsecurity*

ModSecurity merupakan modul keamanan pada server web Apache yang berfungsi sebagai firewall. Standar dan aturan yang tepat *Modsecurity* ini berfungsi, serta memeriksa komunikasi HTTP guna mendeteksi berbagai serangan, termasuk yang belum diketahui (Jair Barcia Peña, 2023).

2.2.11 Ngrok

NGROK merupakan layanan cloud berbasis *open source* yang menyediakan fasilitas untuk membuat terowongan web. Melalui layanan ini, pengguna dapat mengakses dan berkomunikasi dengan *server* web menggunakan alamat IP publik. *NGROK* beroperasi di berbagai wilayah dan zona ketersediaan global untuk memastikan koneksi terowongan tetap aktif dan dapat diakses kapan saja (Praghash et al., 2021).

2.2.12 CVSS (Common Vulnerability Scoring System)

Common Vulnerability Scoring System (CVSS) merupakan metode *standar* yang digunakan untuk mengukur dan menilai tingkat keparahan kerentanan keamanan pada suatu sistem atau perangkat lunak. CVSS memberikan skor numerik dengan rentang nilai antara 0.0 hingga 10.0, di mana skor yang lebih tinggi menunjukkan tingkat risiko yang lebih serius. Sistem ini membantu organisasi dalam mengevaluasi kerentanan secara objektif dan menentukan prioritas perbaikan berdasarkan tingkat keparahan dan potensi dampaknya. Penilaian CVSS didasarkan pada tiga kelompok metrik utama, yaitu:

1. Metrik Basis *(Base Metrics)*: Mengukur karakteristik dasar kerentanan yang bersifat konstan, seperti *Attack Vector (AV), Attack Complexity (AC), Privileges Required (PR), User Interaction (UI), Scope (S),* serta dampaknya terhadap *Confidentiality (C), Integrity (I), dan Availability (A).*
2. Metrik Temporal *(Temporal Metrics)*: Menilai faktor-faktor yang berubah seiring waktu, seperti ketersediaan eksploitasi dan pembaruan *patch.*
3. Metrik Lingkungan *(Environmental Metrics)*: Mengevaluasi kerentanan berdasarkan konteks spesifik suatu sistem, seperti konfigurasi keamanan dan kebijakan organisasi.

Penggunaan CVSS, pengembang dan tim keamanan dapat mengidentifikasi, memprioritaskan, dan menangani kerentanan secara sistematis guna meminimalkan risiko terhadap sistem dan data.

2.2 Penelitian Terkait (*State Of The Art)*

2.2.1 *Web Service REST API Menggunakan PHP Dan Framework Laravel di Tenta Tour Salatiga*

Penelitian (Gowell & Suprihadi, 2024) menjelaskan bahwa penerapan arsitektur *REST* dan penggunaan framework laravel pengolahan data, informasi yang tersimpan pada database secara responsif tanpa keterbatasan platform berkat fungsi interoperabilitas yang di tawarkan sangat efektif mengingat permasalahan pemesanan tiket yang masih menggunakan metode konvensional di Tenta tour dinilai kurang efisien, penelitian selanjutnya masih terdapat banyak ruang untuk pengembangan dan perbaikan kedepan, fitur promosi dan pemesanan dapat dilengkapi lagi dengan sistem pembayaran baik secara manual maupun dengan payment gateway. Batasan penelitian penggunaan Arsitektur *REST* dengan framework *Laravel*.

2.2.2 Implementasi Cross Site Scripting Vulnerability Assessment Tools Berdasarkan OWASP Code Review

Penelitian (Hany et al., 2021) menjelaskan dengan menerapkan metode *Vulnerability Assessment* dan menggunakan kaidah OWASP *Code Review* dapat membantu penemuan kerentanan dengan waktu yang lebih efisien mengingat kerentanan *Cross site Scripting* yang paling sering ditemukan dalam aplikasi web tidak semuanya dari tim *security* fasih terhadap kerentanan web secara menyeluruh.

Penelitian ini menghasilkan, dapat mendeteksi kerentanan dua kali lebih banyak dan variatif dibandingkan dengan aplikasi analisis statis sumber terbuka lainya. Batasan penelitian ini menggunakan metode *Vulnerability Assesment*, menggunakan kaidah OWASP *Code Review* dan menggunakan kerangka Django.

2.2.3 Implementation of Service Oriented Architecture Using Web API dan SOMA in E-commerce Web Application

Penelitian (Samuel & Girsang, 2020) penggunaan arsitektur *SOA* berbasis *API* memiliki waktu akses yang lebih singkat untuk setiap layanan/halaman dibandingkan dengan layanan dengan integrasi n-ke-n (arsitektur monolitik).. Pemilihan tersebut dikarenakan Ditoko.com ada kendala terhadap pembeli dari luar jabodetabek pengiriman hanya bisa terpusat disatu tempat wilayah. Menerapkan sistem pemantauan pada SOMA *Modeling* dan *API Gateway* bertujuan untuk mendeteksi kegagalan pada setiap layanan yang diakses. Hal ini penting karena SOMA *Modeling* dan *API Gateway* digunakan dalam aplikasi berskala semi-enterprise, sehingga mampu mendukung kebutuhan analisis data dari setiap layanan yang dijalankan. Batasan penelitian ini perpaduan penerapam Arsitektur SOA dan *Microservice.* Metode yang digunakan berupa SOMA dan *API gateway* sebagai integrasi aplikasi dan layanan, serta pengujian keamanan menggunakan *white box* dan *black box testing*.

2.2.4 Rest API Pada Toko Kelontong Untuk Transaksi Penjualan Menggunakan Framework Laravel

Penelitian (Herdiyatmoko & Pratama, 2024) menjelaskan penerapan arsitektur *Restful API* dan keamanan token akses*, Rest API* dikembangkan dengan teknologi *VUE.JS* dan *Laravel*. Penelitian ini berkontribusi dalam pengembangan model akses data berbasis *Rest API*, yang memiliki pendekatan berbeda dibandingkan dengan metode pemrograman. Hasil pengujian menunjukkan bahwa penerapan *VUE.JS* dan *Laravel* memberikan kinerja yang optimal dengan penggunaan *JSON Web Token (JWT)* sebagai token akses. Selain itu, integrasi *Laravel* sebagai *REST Server* dan *VUE.JS* sebagai *REST Client* menunjukkan tingkat interoperabilitas yang baik. Batasan penelitian penggunaan arsitektur *Restful API* dengan metode *waterfall* penggunaan framework yaitu *Laravel* dan *VUE.JS*.

2.2.5 A Review On Web Application Vulnerability Assessment and Penetration Testing

Penelitian (Ravindran & Potukuchi, 2022) menjelaskan pentingnya pengujian kerentanan sejak dini, penggunaan metode VAPT sangat disarankan untuk mengetahui celah kerentanan apa yang ada dalam sebuah web untuk dilakukanya perbaikan guna menghindari serangan dari pihak yang tidak bertanggung jawab serta menjelaskan *tools* teratas dalam pengujian. Karena seiring bertambahnya pengguna internet serangan *cyber* pun akan semakin meningkat dan menjadi tantangan bagi sebuah organisasi untuk memastikan web yang dimiliki Perusahaan itu terbilang aman. Hasil dari penelitian ini menjelaskan tentang mendeteksi kerentanan itu ada 2 cara yakni pengujian secara manual dan pengujian secara otomatis serta tools yang relevan terhadap pengujian otomatis seperti Nmap,

Acunetix, Nessus, Owasp dan untuk pengujian manual burp suite profesional. Batasan penelitian ini menjelaskan sebuah metode VAPT beserta tools dalam pengujianya.

2.2.6 *A* Systematic *Analysis: Website Development Using Codeigniter And Laravel Framework*

Penelitian (Widodo Purbo, 2021) menjelaskan pentingya pemilihan framework untuk mempermudah proses kerja, dalam penelitian ini membandingkan framework dari bahasa pemrograman PHP antara framework *Codeigniter* dan *Laravel* dalam hal performa waktu, kecepatan dan keamanan. Website kini menjadi sarana komunikasi penting dalam berbagai aktivitas manusia karena biayanya yang terjangkau. Selain berfungsi sebagai media penyimpanan data dan informasi dengan topik tertentu, website juga merupakan kumpulan halaman web yang saling terhubung dalam jaringan. Hasil dari penelitian ini menunjukan *Codeigniter* lebih unggul dalam rata-rata waktu dan kecepatan, codeigniter memiliki waktu yang lebih besar durasinya lebih lama daripada *Laravel* dengan selisih waktu 4,2 ms dan pada kecepatan 40,08 Kbit/s dari laravel. Batasan penelitian ini framework *Laravel* dan *Codeigniter* serta alat ujinya menggunakan *Load Test*.

2.2.7 Analisis Perbandingan Kinerja Framework *Codeigniter* Dengan Expres*.Js* Pada *Server RESTfull API*

Penelitian (Mulana et al., 2022) membandingkan performance framework *Codeigniter* dan *Expres.Js* dalam hal performace *Expres.Js* lebih unggul daripada

Codeigniter dengan rata waktu respons 420,72 ms sedangkan pada framework *Codeigniter* 555,90 ms, tetapi di sisi lain *Expres.Js* mengonsumsi resource lebih besar daipada *Codeigniter* dengan nilai penggunaan CPU 1,95% dan penggunaan memori 2,74% sedangkan pada *Codeigniter* CPU 1,34% penggunaan memori 2,31%. Karena pentingnya pemilihan teknologi dalam pembuatan *RESTful API* dapat mempengaruhi kinerja *server*. Hasil dari ponelitian ini *Express.Js* cocok untuk sistem yang diakses oleh banyak pengguna dan ditempatkan pada server dengan spesifikasi tinggi. Sementara framework *Codeigniter* cocok diimplementasikan untuk sistem dengan akses pengguna yang lebih sedikit. Penelitian selanjutnya dalam penelitian ini mencoba framework backend lainya yang bisa diterapkan pada *RESTful API*. Batasan penelitian ini framework *Codeigniter* dan *Expres.Js* metode yang digunakan pengujian kinerja.

2.2.8 *Utilising VAPT Technologies (Vulnerability Assessment & Penetration Testing) As A Method For Actively Preventing* Cyberattacks

Penelitian (Kour & Kaur, 2022) membahas tentang metode VAPT sebagai teknologi pencegahan serangan siber, yang memberikan perlindungan proaktif terhadap ancaman. Kompleksitas sistem yang seiring berkembangnya jaman akan semakin banyaknya kerentanan dalam sistem. Hasil penelitian ini dijelaskan proses lengkap penggunaan Penilaian Kerentanan dan Uji Penetrasi sebagai teknologi pertahanan siber yang efektif beserta *tools* teratas yang di rekomendasikan untuk pengujian VAPT. Batasan penelitian penggunaan metode VAPT serta rekomendasi 15 *tools* dalam pengujian VAPT.

2.2.9 *Comparative Analysis Of Codeigniter, Laravel, And* KTUPAD Framework: *Case Study Online Exam Applications*

Penelitian (Sismadi et al., 2022) membandingkan framework *Laravel, Codeigniter* dan KTUPAD pada ujian Online di SMP Negeri 242 Jakarta dalam hal performa dan keamanan sebagai acuan bagi para pengembang web agar dapat memilih dengan tepat perihal kerangka web. Dilakukan penelitian ini menilai akses informasi digital memiliki peran yang sangat penting. Hasil dari penelitian ini framework dari KTUPAD dalam performa dan kemanan unggul dari kedua framework PHP dengan nilai performa 99 dengan kemanan rendah untuk penelitian selanjutnya penggunaan alat forensik digital analisis (SDLC) untuk membandingkan ketiga kerangka web tersebut. Batasan penelitian penggunaan *tools* mengukur performa yaitu *Apache Jmeter* dan *tools* keamanan yaitu OWASP.

2.2.10 Nautilus: Deteksi Kerentanan *RESTful API* Otomatis

Penelitian (Deng et al., 2023) membandingkan beberapa alat pengujian untuk menguji kerentanan pada Rest Api, NAUTILUS berhasil mendeteksi lebih baik dengan nilai 141% mendapatkan lebih banyak kerentanan. Kesulitan dalam mendeteksi kerentanan di sebuah *Rest API* memanglah sebuah langkah yang tidak mudah dilakukan karena tidak dapat secara efektif memperoleh hubungan antara kedua operasi *API*, begitu juga dengan tim keamanan kurangnya akan kesadaran dan kebenaran urutan operasi *API* selama pengujian. Hasil dari penelitian ini menghasilkan penggunaan tools NAUTILUS mampu mendeteksi kerentanan pada

Rest API mengungguli dari 2 tools. Batasan penelitian ini penggunaan metode *Pentest* dan penerapan tools NAUTILUS.

2.2.11 Design *And Implementation Of RESTful API-Based Model For Vulnerability Detection And Mitigation*

Penelitian (Modi et al., 2022) menjelaskan berbagai kerentanan web dan implementasi serta deteksinya, penerapan autentikasi firewall aplikasi web dalam bentuk *REST API* dapat membantu mengurangi beban pada server klien karena data diproses pada server cermin tempat *API* disimpan. Pesatnya pertumbuhan aplikasi di dunia maka masalah keamanan juga akan semakin beragam, dan berbagai macam ancaman web. Hasil dari penelitian ini setelah berbagai macam pengujian kerentanan menghasilkan penerapan firewal pada rest api membuat sistem aman dan lebih efisien. Batasan penelitian pendeteksian berbagai macam kerentanan menggunakan *tools* waf dan teknik serangan secara manual.

2.2.12 *A* Black *Box Tool for Robustness Testing of Rest API*

Penelitian (Laranjeiro et al., 2021) melakukan penerapan tools bBOXRT untuk menguji katahanan *Rest API*, bBOXRT ini sebuah alat black-box sederhana berbasis aturan yang di berfokus dalam evaluasi ketahanan dan menunjukkan efektivitasnya dalam mengungkapkan permasalahan ketahanan, bahkan dalam layanan bisnis penting yang telah teruji, di mana ketahanan merupakan perhatian utama. Layanan *REST* masih kekurangan pendekatan praktis dalam keamanan, sehingga bisa menyebabkan bug residual dan menyebabkan operasi layanan gagal. Adapun untuk penelitian ini selanjutnya akan menambahkan dukungan untuk

payload tambahan untuk menyempurnakan pengetesan menggunakan bBOXRT. Batasan masalah penggunaan *tools* bBOXRT dan penggunaan metode Pentest.

2.2.13 Analysis *Of Vulnerability Assessment With Penetration Testing*

Penelitian (Mohan et al., 2021) menjelaskan metode VAPT sebagai metode yang paling modern untuk mendeteksi kerentanan dengan berbagai macam tools untuk mendeteksinya, karena dalam penelitian ini pendeteksian kerentanan itu hal wajib dilakukan agar terhindar dari serangan dan eksploitasi yang tidak bertanggung jawab. Melihat kejahatan siber yang seiring waktu semakin berkembang, yang bisa memamfatkan kerentanan untuk mencuri informasi rahasia yang bisa dieksploitasi, Hasil dari penelitian ini dalam penggunaan metode VAPT berhasil menyediakan daftar lengkap kelemahan dan hasil disebabkan oleh kerentanan dalam web tersebut, untuk penelitian selanjutnya berencana menganalisis sistem target dengan kompleksitas yang lebih tinggi. Batasan penelitian penggunaan metode VAPT dan penggunaan beberapa tools untuk menjalankan VAPT.

2.2.14 API Security Testing: The Challenges Of Security Testing For Restful API

Penelitian (Alharbi & Moulahi, 2023) menawarkan langkah-langkah dalam pembuatan *RESTful API* terjaga dari serangan cyber, serta menjelaskan kerentanan umum yang biasa terjadi di *RESTful API* dan pengujianya. Aplikasi web banyak beralih menggunakan *RESTful API,* hal ini membuat lebih rentan terhadap ancaman

karena aplikasi web bekerja secara *independent*. Hasil dari penelitian ini pentingnya pengujian menyeluruh dan mitigasi untuk mengamankan *RESTful API*, untuk penelitian selanjutnya penerapan algoritma pembelajaran mesin untuk mendeteksi kerentanan pada *RESTful API* dan pembuatan alat pengujian otomatis. Batasan penelitian langkah-langkah pengujian *REST API* dan berbagai macam cara pengujian *REST API*.

2.2.15 *Vulnerability Assessment* Pada Situs Web KPPM FRI Dengan Burp Suite dan *Intruder*

Penelitian (Indera et al., 2023) menjelaskan metode *Vulnerability Assesment* dan penggunaan *tools* Burp Suite dan Intruder sebagai aplikasi untuk pengujian kerentanan, hasil dari penelitian ini Burp Suite sebanyak 9 kerentanan yang seluruhnya valid, sehingga tingkat akurasi scanning pada software Burp Suite sebesar 100%, sementara kerentanan yang ditemukan oleh software Intruder sebanyak 6 kerentanan dengan kerentanan yang valid hanya sebanyak 3 kerentanan, sehingga tingkat akurasi scanning pada software Intruder sebesar 50%. Semakin Berkembanganya jaman, kejahatan siber akan semakin meningkat melihat situs web KPPM yang digunakan untuk mengelola kegiatan kerja praktik dan pengabdian masyarakat, belum pernah dilakukan deteksi kerentanan. Hasil dari penelitian ini bahwa aplikasi Burp Suite efektif dalam *scanning* untuk menemukan kerentanan pada sebuah web. Batasan penelitian ini penggunaan metode *Vulnerability Assesment* penggunaan tools burp suite dan intruder.

2.3 Matriks Penelitian

Tabel 2.3 merupakan matriks penelitian yang berisi penelitian yang berpokus untuk menyelesaikan masalah keamanan Arsitektur *Service-Oriented Architecture,* selain itu matriks ini dapat memberikan informasi tentang pembedaan dan kesamaan penelitian yang akan dilakukan dari penelitian terdahulu.

Tabel 2. 3 Matriks Penelitian

No	Judul	Ruang Lingkup									
		Arsitektur		**Framework**		**Metode**		***Tools***			
		Service Oriented Architecture	***Restful API***	***Laravel***	***Codeigniter***	***Vulnerability Assessment***	***Penetration Testing***	***Burp Suite***	***OWASP***	***Postman***	***Nmap***
1	Perancangan Web Service REST API Menggunakan PHP dan Framework Laravel di Tenta Tour Salatiga (Gowell & Suprihadi, 2024)		✓	✓							
2	Implementasi Cross Site Scripting Vulnerability Assessment Tools berdasarkan OWASP Code Review (Hany et al., 2021)					✓			✓		
3	Implementation of Service Oriented Architecture Using Web API dan SOMA in E-commerce Web Application (Samuel & Girsang, 2020)	✓	✓								
4	Rest API Pada Toko Kelontong Untuk Transaksi Penjualan Menggunakan Framework Laravel (Herdiyatmoko & Pratama, 2024)	✓	✓	✓							

No	Judul	Ruang Lingkup									
		Arsitektur		Framework		Metode		*Tools*			
		Service Oriented Architecture	*Restful API*	*Laravel*	*Codeigniter*	*Vulnerability Assessment*	*Penetration Testing*	*Burp Suite*	*OWASP*	*Postman*	*Nmap*
5	A Review on Web Application Vulnerability Assessment and Penetration Testing (Ravindran & Potukuchi, 2022)					✓	✓				
6	A Systematic Analysis: Website Development using Codeigniter and Laravel Framework (Widodo Purbo, 2021)			✓	✓						
7	Analisis Perbandingan Kinerja Framework Codeigniter Dengan Express.Js Pada Server RESTful Api (Mulana et al., 2022)		✓		✓						
8	Utilizing VAPT Technologies (Vulnerability Assessment & Penetration Testing) as a Method for Actively Preventing Cyberattacks (Vegesna, 2022)					✓	✓	✓			
9	Comparative Analysis Of Codeigniter, Laravel And KTUPAD Frameworks: Case Study Online Exam Applications (Sismadi et al., 2022)			✓	✓				✓		
10	Nautilus: Automated RESTful API Vulnerability Detection (Deng et al., 2023)		✓				✓				
11	Design and implementation of RESTFUL API based model for vulnerability detection and mitigation (Modi et al., 2022)		✓				✓				
12	A Black Box Tool for Robustness Testing of REST Services (Laranjeiro et al., 2021)		✓				✓				

No	Judul	Ruang Lingkup									
		Arsitektur		Framework		Metode		*Tools*			
		Service Oriented Architecture	***Restful API***	***Laravel***	***Codeigniter***	***Vulnerability Assessment***	***Penetration Testing***	***Burp Suite***	***OWASP***	***Postman***	***Nmap***
13	Analysis Of Vulnerability Assessment With Penetration Testing (Mohan et al., 2021)					✓	✓	✓			✓
14	API Security Testing: The Challenges of Security Testing for Restful APIs (Alharbi & Moulahi, 2023)		✓			✓	✓				
15	Vulnerability Assessment Pada Situs Web KPPM FRI Dengan Burp Suite dan Intruder (Indera et al., 2023)					✓		✓			
16	Analisis Keamanan Framework Laravel dan Codeigniter Dalam Service Oriented Architecture Menggunakan VAPT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Berdasarkan relevansi matriks penelitian yang terdapat pada Tabel 2.3 penelitian ini mengusulkan deteksi keamanan pada *web service* menggunakan Arsitektur *Service-Oriented Architecture (SOA)* berbasis *REST API,* yang memanfaatkan metode *Vulnerability Assessment and Penetration Testing (VAPT). Tools* yang digunakan seperti Postman, OWASP, dan Burp Suite. Penelitian ini membandingkan tingkat keamanan antara framework PHP, yaitu *Laravel* dan *Codeigniter* melalui kombinasi pengujian otomatis dan manual.