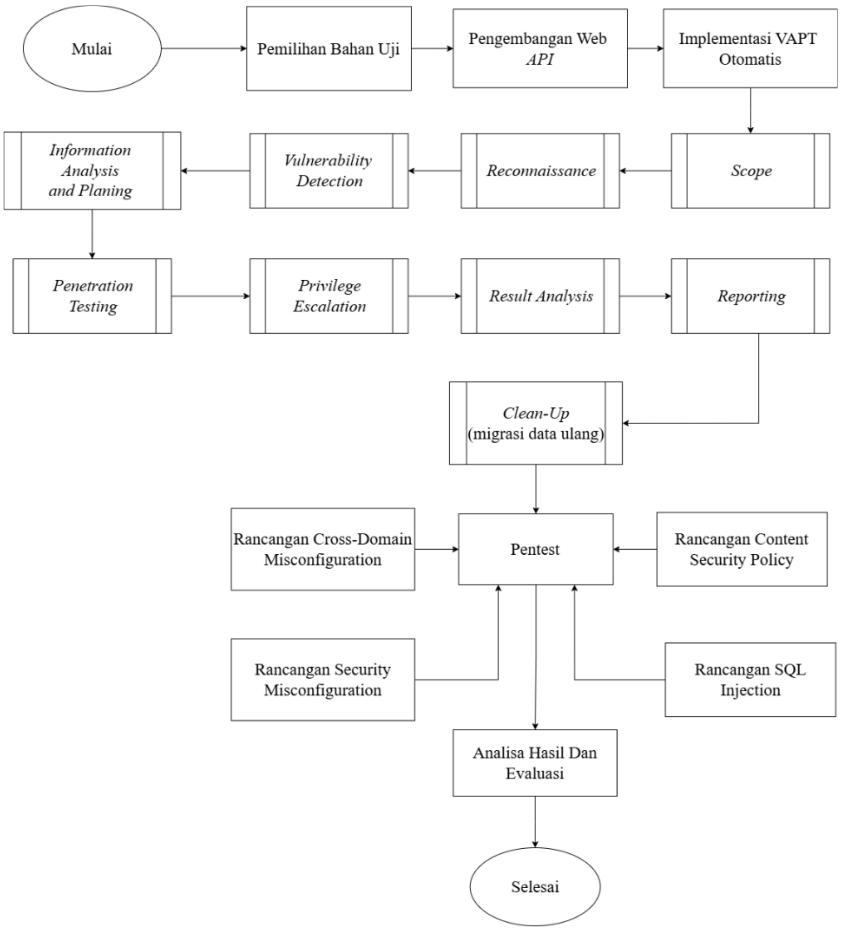
# BAB III

# METODOLOGI PENELITIAN

## 3.1 Metodologi Penelitian

Metodologi penelitian ini menggunakan pendekatan kuantitatif dengan melakukan penilaian kerentanan menggunakan *tools* OWASP, Penilaian tersebut menghasilkan berbagai jenis kerentanan pada sistem. Selanjutnya, dilakukan pengujian penetrasi secara otomatis menggunakan OWASP dan kerentanan tersebut divalidasi melalui pengujian penetrasi manual menggunakan Burp Suite pada kedua framework. Gambar 3.1 merupakan tahapan-tahapan penelitian yang dilakukan.




Gambar 3. 1 Alur tahapan penelitian

Keterangan gambar 3.1 terhadap simbol yang digunakan :

1. *Connector Symbol,* yaitu simbol yang di pakai untuk keluar – masuk atau penyambungan proses dalam lembar / halaman yang sama, simbol tersebut ditandai dengan bentuk bulat.

2. *Predifine Proses,* yaitu simbol untuk pelaksanaan suatu bagian sub-program / *procesure,* simbol tersebut memiliki bentuk persegi panjang mempunyai dua garis vertikal sejajar di sisi kiri dan kanan.

3. *Processing Symbol,* yaitu simbol yang menunjukan pengolahan yang dilakukan oleh komputer, simbol *processing* berbentuk persegi panjang.

### 3.1.1 Studi Literatur

Studi literatur merupakan langkah awal penelitian dengan mengumpulkan materi terkait keamanan web *service REST API Laravel* dan *Codeigniter.* Penelitian ini menggunakan metode *Vulnerability Assesment and Penetration Testing*, *tools* yang digunakan seperti OWASP, Burp Suite dan Postman, serta penggunaan OWASP Top 10 *API* 2023 sebagai panduan untuk mengidentifikasi dan memahami kerentanan keamanan paling umum yang dapat terjadi pada *API*. Referensi dikumpulkan dari perpustakaan, situs web dan publikasi ilmiah yang tersedia secara resmi di internet.

### 3.1.2 Pemilihan Bahan Uji

Penelitian dilakukan menggunakan laptop dan bahan uji berupa web *service* yang dibangun menggunakan framework *Laravel* dan *Codeigniter.* Kebutuhan *hardware* dan *software* yang digunakan.

### 3.1.3 Pengembangan Web *API*

Tahap pengembangan web *API,* framework *Laravel* dan *Codeigniter,* digunakan untuk membuat *RESTful API* sederhana dengan fungsi CRUD pada sistem peringkat pemain badminton. Autentikasi menggunakan JWT untuk mengamankan akses, dan setiap fungsi CRUD memiliki *method* dan *endpoint* spesifik untuk mempermudah akses pengujian serta skema jaringan antara *server* dan klien.

### 3.1.4 Implementasi VAPT Otomatis

Proses mengevaluasi keamanan web *RESTful API* CRUD pada sistem peringkat pemain badminton dari kedua framework menggunakan metode *Vulnerability Assesment* dan *Penetration Testing, tools* yang digunakan dalam pengujian yaitu OWASP, Postman dan Burp Suite dipilih karena *tools* tersebut masuk kedalam bagian beberapa *tools* teratas yang sering digunakan untuk pendeteksian keamanan (Vegesna, 2022). Metode yang dilakukan terdapat dua metode yaitu metode otomatis untuk mendeteksi secara keseluruhan dan metode manual digunakan untuk memverifikasi kerentanan yang dihasilkan dari metode otomatis, serta pengujian dilakukan dalam dua kondisi yaitu *enable/disable* keamanan default masing-masing framework dan dua kondisi *enable/disable Modsecurity*. Berikut *tools* yang digunakan dari kedua metode :

1. Metode Otomatis : Pengujian otomatis dilakukan dengan dua tahapan yaitu secara pasive dan aktif menggunakan OWASP sebagai *tools* utama, Postman digunakan untuk mengirim data dari *endpoint* framework melalui proxy.

2. Metode Manual : Pengujian manual dilakukan menggunakan Burp Suite untuk memvalidasi kerentanan yang ditemukan dari pengujian otomatis OWASP dan Postman digunakan untuk mengirim data dari *endpoint* framework melalui proxy.

Proses evaluasi keamanan sistem melalui metode *Vulnerability Assesment* dan *Penetration Testing* dilakukan secara sistematis dengan mengikuti beberapa tahapan penting. Tahapan-tahapan ini mencakup mulai dari perencanaan hingga pelaporan dan pembersihan sistem. Berikut adalah sembilan tahapan yang dilalui dalam proses VAPT untuk memastikan keamanan sistem secara menyeluruh.

a. *Scope,* Penentuan ruang lingkup mencakup pemeriksaan server aktif dan pengecekan port menggunakan *tools Nmap* dan cURL.

b. *Reconnaissance,* Mengumpulkan informasi tentang *endpoint, method* dari kedua framework dan *enable/disable* keamanan dari kedua framework serta.

c. *Vulnerability Detection,* Mengidentifikasi kerentanan pada masing-masing *endpoint* framework dengan melakukan *scanning* otomatis pasif menggunakan *tools* OWASP pada dua kondisi *enable/disable* kedua framework.

d. *Information Analysis and Planing,* Menganalisis kerentanan yang terdeteksi dari *Vulnerability Detection* pengujian otomatis pasif OWASP, kemudian merencanakan validasi pengujian aktif OWASP untuk memverifikasi keaslian kerentanan.

e. *Penetration Testing,* Dilakukan eksploitasi setiap *endpoint* dari kedua framework secara langsung untuk memvalidasi kerentanan yang ditemukan dari *Vulnerability Detection* pengujian otomatis pasif OWASP, menggunakan pengujian penetrasi *scanning* otomatis aktif OWASP dalam dua kondisi enable/disable keamanan kedua framework.

f. *Privilege Escalation,* Memvalidasi keaslian token yang digunakan pada *endpoint API* dalam sistem peringkat pemain badminton untuk memastikan bahwa kedua framework menerapkan autentikasi token dengan benar dan aman. Pengujian dilakukan menghapus salah satu karakter token. Proses tersebut dilakukan menggunakan *tools* Postman.

g. *Result Analysis,* Analisis hasil kerentanan dari pengujian otomatis aktif OWASP, melakukan klasifikasi terhadap kerentanan yang ditemukan menggunakan OWASP TOP 10 *API* 2023 dan memvalidasi kerentanan tersebut melalui pengujian penetrasi manual dengan melakukan penyisipan queri dari masing-masing kerentanan menggunakan Burp Suite. Melewati proses *clean-up* agar pengujian manual tepat, tidak ada bekas pengujian penetrasi otomatis aktif OWASP di dalam server.

h. *Reporting,* Penyusunan laporan kerentanan yang mendetail mengenai temuan dari pengujian penetrasi otomatis aktif OWASP terhadap kerentanan dari dua framework.

i. *Clean-up,* Dilakukan pemulihan web *API* CRUD peringkat pemain badminton pada kedua framework, melalui migrasi data ulang. Proses ini

dilakukan sebagai persiapan untuk melakukan *Penetration Testing* manual dengan menggunakan Burp Suite.

### 3.1.5 *Penetration Testing*

Pengujian *Penetrasi* dilakukan dua kondisi yaitu enable/disable keamanan kedua framework untuk memvalidasi kerentanan yang dihasilkan dari framework *Laravel* dan framework *Codeigniter.* Validasi ini hanya mengutamakan kerentanan tingkat medium hingga kerentanan tinggi, hasil tersebut akan divalidasi keaslianya yang sebelumnya terdeteksi melalui pengujian otomatis aktif. *Pentest* dilakukan menggunakan Burp Suite, pengujian manual ini dilakukan lebih lanjut untuk mengidentifikasi kerentanan tersebut secara mendalam agar bisa membandingkan framework mana yang lebih baik dalam hal keamanan yang sama diterapkan.

### 3.1.6 Rancangan *Security Misconfiguration*

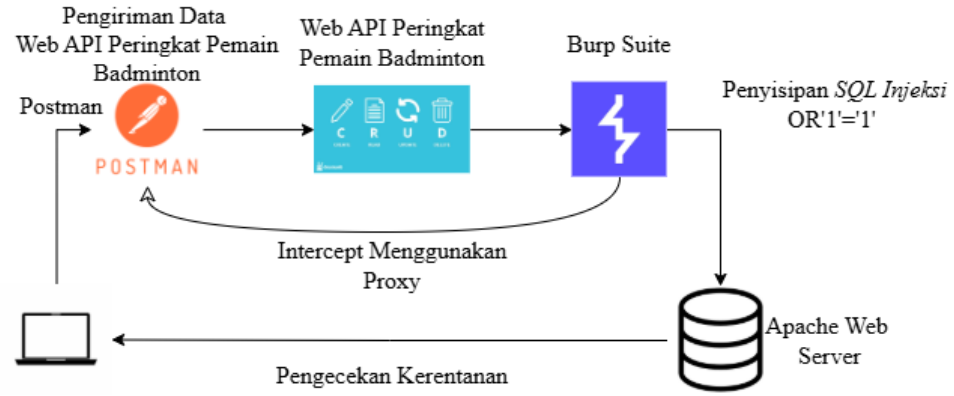
Perancangan skenario Secuirty Misconfiguration dilakukan melalui proses pengklasifikasian berdasarkan OWASP Top 10 API. Beberapa kerentanan yang ditemukan dari setiap method di kedua framework (*Laravel* dan *Codeigniter*) menunjukan bahwa kerentanan tersebut termasuk dalam kategori *Security Misconfiguration*, untuk memvalidasi kerentanan yang dihasilkan melalui pengujian otomatis OWASP, dilakukan pengujian manual menggunakan Burp Suite guna memastikan bahwa kerentanan tersebut benar-benar ada dan dapat dieksploitasi. Kerentanan yang terdeteksi dari kedua framework ini adalah :

1. *.htaccess Information Leak*
2. *Buffer Overflow*

3. *Cross Domain Misconfiguration*

### 3.1.7 Rancangan *SQL Injection*

Pengujian *SQL Injection* dilakukan dengan menerapkan rancangan *SQL Injection* pada method framework *Laravel* yang terdeteksi rentan. Meskipun pada framework *Codeigniter* tidak ditemukan kerentanan *SQL Injection*, pengujian manual tetap dilakukan menggunakan Burp Suite untuk memvalidasi hasil pengujian otomatis. Validasi dilakukan dengan menyisipkan payload *SQL Injection* pada header di method yang terdeteksi rentan dan untuk framework *Codeigniter* pengujian dilakukan pada *method* tertentu kemudian ketika rentan maka akan diuji pada *method* yang lain seperti halnya pengujian terhadap framework *Laravel*.

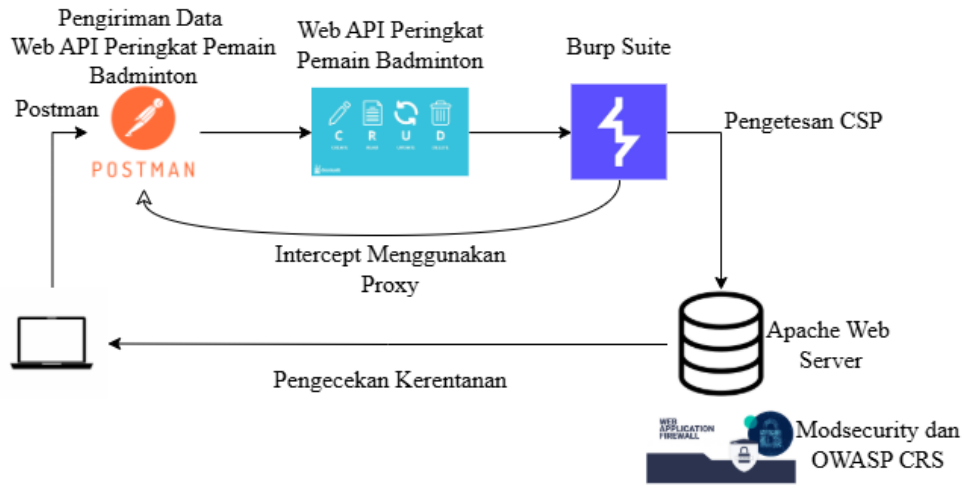



Gambar 3. 2 Kerentanan *SQL Injection*

Gambar 3.2 dilakukan melalui proses penyisipan *Query SQL Injeksi* yaitu OR’1’=’1’ terhadap *method* yang terdeteksi rentan pada framework *Laravel* dan dilakukan juga pengujian pada framework *Codeigniter* untuk memastikan framework *Codeigniter* aman.

### 3.1.8 Rancangan *Content Security Policy (CSP) Header Not Set*

Proses merancang *Pentest Content Security Policy (CSP) Header Not Set,* ini dilakukan untuk mengetahui lebih lanjut terhadap *method* yang terindikasi dari kedua framework yaitu *method PUT* dan *DELETE.* Kerentanan yang muncul yaitu *Content Security Policy (CSP) Header Not Set* divalidasi lebih lanjut karena tingkat kerentananya tergolong medium. Kerentanan ini bisa terjadi apabila *server* tidak mengonfigurasi *header CSP* dengan benar. Kondisi ini menunjukkan bahwa aplikasi rentan terhadap serangan seperti *cross-site scripting (XSS)* karena tidak adanya kebijakan keamanan konten yang membatasi sumber daya eksternal. Akan tetapi, mekanisme *ModSecurity* dapat berperan sebagai lapisan pertahanan tambahan *(defense layer)* untuk meminimalkan risiko tersebut dengan melakukan penyaringan dan pemblokiran terhadap permintaan berbahaya yang tidak diizinkan (Scano., 2025). Sehingga pengujian manual dilancarkan dengan dua cara yaitu, pengujian *Modsecurity* dalam kondisi On/Off.

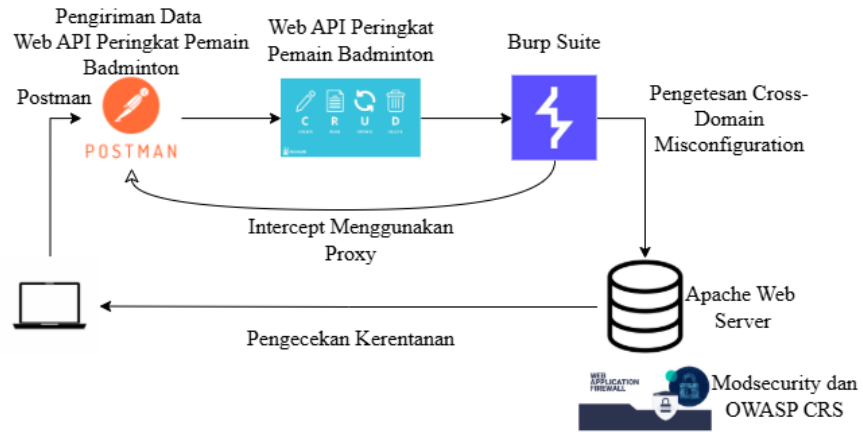



Gambar 3. 3 Skenario *Content Security Policy*

Gambar 3.3 menunjukkan alur pengujian *Content Security Policy (CSP) Header Not Set* yang dibagi menjadi dua kondisi, yaitu saat *ModSecurity* dalam keadaan aktif (*On*) dan tidak aktif (*Off*). Pada pengujian ini, *Postman* digunakan sebagai alat untuk mengirimkan data, sedangkan *Burp Suite* berfungsi untuk mengintercept data melalui *proxy* sebelum data dikirimkan ke server melalui *Burp Suite*.

### 3.1.9 Rancangan *Security Misconfiguration*

Rancangan *Security Misconfiguration* sebelumnya dilakukan pengklasifikasian berdasarkan OWASP Top 10 *API* 2023. Beberapa *method* framework *Laravel* terdeteksi rentan yaitu *method GET All, POST* dan *GET By Data*, pengujian manual dijalankan guna memastikan hasil dari *pentest* aktif OWASP bahwa kerentanan tersebut benar-benar valid dan dapat tereksploitasi. Kerentanan ini terjadi akibat konfigurasi domain tidak membatasi asal permintaan (*origin*) secara tepat, sehingga memungkinkan akses lintas domain tanpa otorisasi yang valid dan berpotensi dimanfaatkan oleh penyerang untuk melakukan pencurian data atau manipulasi permintaan. Skenario pengujian untuk *Security Misconfuguration* pada Gambar 3.4




Gambar 3. 4 Skenario *Security Misconfiguration*

Gambar 3.4 pengujian manual menggunakan *tools* Burp Suite, langkah tersebut menggunakan bantuan *tools* Postman untuk mengirimkan data dari *API* web peringkat pemain badminton melalui *proxy* kemudian di intercept menggunakan Burp Suite. Apabila hasil respon saat pengiriman data terdapat *Acces-Control-Allow-Origin* pada *header* respon, maka *method* tersebut tervalidasi rentan.

### 3.1.10 Analisa Hasil dan Evaluasi

Proses analisa kerentanan dan evaluasi diambil dari hasil *generate report* OWASP, seperti yang menjelaskan penyebab dan dampak dari setiap kerentanan. Langkah ini dilakukan untuk melakukan perbandingan keamanan terhadap dua framework ketika keamanan default dari masing-masing diaktifkan dan dinonaktifkan. Mitigasi risiko keamanan dan perhitungan skor kerentanan menggunakan CVSS diperlukan untuk membandingkan tingkat keamanan yang lebih baik pada kedua framework dalam konteks konfigurasi bawaan (*default*).

### 3.1.11 Kesimpulan dan Saran

Tahapan ini merupakan tahapan akhir yang menyimpulkan hasil penelitian dan saran untuk penelitian selanjutnya