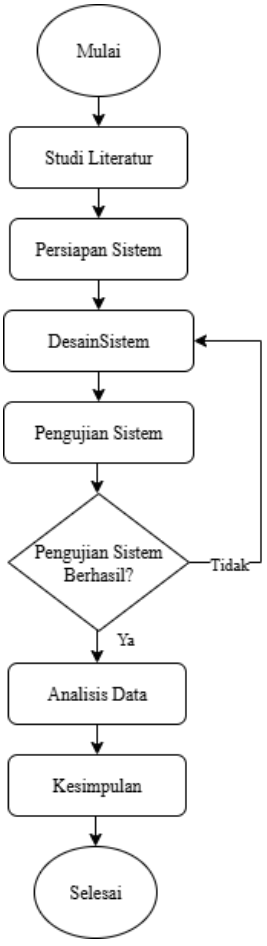
BAB III
METODE PENELITIAN

Pada bagian metode penelitian ini menjelaskan tahapan sistematis yang digunakan dalam proses penelitian. Tahapan sistematis dimulai dari perancangan sistem, pengujian sistem, pengukuran data dan alat dan bahan. Setiap prosedur dijabarkan secara terstruktur untuk mendukung pengembangan sistem *active response* Wazuh yang terintgrasi VirusTotal API dan notifikasi *real-time* Telegram.

3.1 Alur Penelitian

Alur penelitian ini bertujuan untuk mengarahkan penelitian dengan struktur yang sistematis, serta mempermudah dalam memahami proses alur penelitian. Berikut merupakan alur penelitian yang dijelaskan pada Gambar 3.1.



Gambar 3. 1 Alur Penelitian

3.1.1 Studi Literatur

Penelitian ini dimulai dengan mengumpulkan konsep terkait Wazuh, *active reponse*, VirusTotal API, Telegram Bot, dan deteksi *malware* serta mengkaji penelitian terdahulu yang relevan. Tujuan dari tahap ini untuk membangun landasan teoritis yang kuat, mengidentifikasi penelitian terdahulu yang relevan, serta metode yang akan digunakan untuk pengembangan sistem

3.1.2 Persiapan Sistem

Persiapan yang dilakukan pada tahap ini mencakup semua persiapan teknis yang diperlukan sebelum melakukan perancangan dan pengujian. Untuk persiapan yang dilakukan yaitu, menyiapkan *server* untuk Wazuh *Manager* dan melakukan *installasi* pada Wazuh *agent* pada tiga mesin target dengan sistem operasi yang berbeda (Ubuntu, Windows, dan CentOS). Serta menyiapkan *file* EICAR sebagai pengujian *file malware*.

3.1.3 Desain Sistem

Desain sistem bertujuan untuk memberikan informasi penting untuk memberikan informasi perancangan *active response*, integrasi Wazuh dan VirusTotal API, Integrasi Wazuh dan Telegram Bot API, dan skenario penelitian.

3.1.3.1 Desain *Active Response*

active response adalah sebuah mekanisme yang memungkinkan sistem keamanan untuk merespon ancaman secara otomatis berdasarkan aturan yang telah ditetapkan, tanpa memerlukan intervensi manual dari administrator sistem. Tujuan utama dari mekanisme ini adalah untuk mengurangi waktu respons terhadap ancaman serta mencegah serangan lebih lanjut terhadap sistem.

Konfigurasi *active response* pada Wazuh menggunakan skrip Python yang dirancang untuk merespon ancaman yang terdeteksi secara otomatis, seperti menghapus *file* yang teridentifikasi sebagai *malware*. Skrip ini bekerja dengan memeriksa peringatan dalam format JSON yang dikirim oleh Wazuh, yang berisi informasi tentang ancaman, seperti *rule* ID dan *file* yang terlibat. Skrip memverifikasi perintah yang diterima, baik itu untuk menambahkan "*add*"

(menambahkan/memproses *alert* baru) atau menghapus "*delete*" (menghapus tindakan sebelumnya) dan jika perintahnya adalah "*add*", skrip akan mengirim permintaan ke *manager* Wazuh untuk memeriksa apakah tindakan dapat dilanjutkan. Jika diizinkan, skrip melanjutkan untuk menghapus *file* yang terdeteksi sebagai ancaman oleh VirusTotal dan mencatat hasilnya.

Pengaturan di bagian <*command*> dan <*active-response*> mengkonfigurasi perintah *remove-threat*, yang menjalankan *file remove-threat* untuk menghapus *file* berbahaya, dengan pengaturan agar perintah ini dijalankan secara lokal pada sistem saat ada peringatan yang relevan. Konfigurasi ini memastikan bahwa tindakan penghapusan dilakukan secara otomatis begitu ancaman terdeteksi berdasarkan *rule* ID 87105.

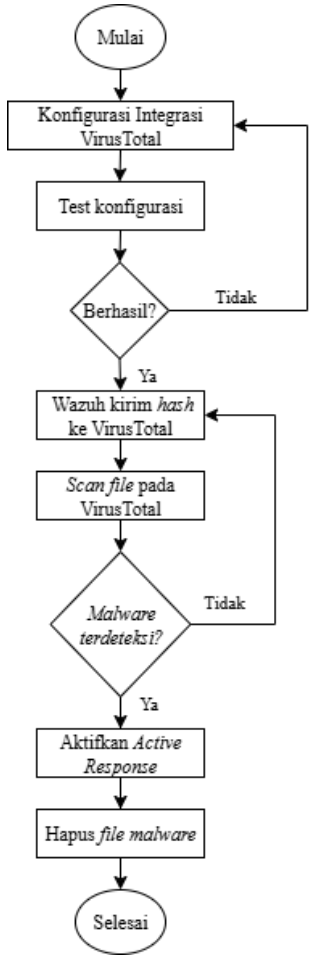
Untuk mempermudah pemantauan hasil tindakan tersebut, ditambahkan aturan dalam konfigurasi <*group*>. Dua aturan dengan ID 100092 dan 100093 digunakan untuk mencatat hasil penghapusan, masing-masing untuk situasi berhasil ("*Successfully removed threat*") dan gagal ("*Error removing threat*"). Keduanya dikaitkan dengan if_sid 657 dan akan ditampilkan di *dashboard* Wazuh, memungkinkan *administrator* untuk memantau hasil otomatis secara *real-time* dan memastikan bahwa ancaman ditangani dengan tepat.

3.1.3.2 Desain Wazuh dan VirusTotal API

Dalam perancangan sistem ini, berdasarkan pada Gambar 3.2 dilakukan integrasi antara Wazuh dan layanan VirusTotal untuk mendeteksi *file* berbahaya secara otomatis. Integrasi ini diatur dalam konfigurasi Wazuh dengan menambahkan bagian <*integration*>, dimana API dari VirusTotal dimasukkan agar Wazuh dapat mengakses layanan tersebut.

Setelah konfigurasi selesai, dilakukan pengujian konfigurasi dengan memasukkan *file malware* sebagai uji coba. Jika pengujian konfigurasi tidak berhasil, maka Wazuh tidak akan menampilkan keluaran. Namun, jika pengujian konfigurasi berhasil, Wazuh akan mengirimkan *hash file* yang terdeteksi ke VirusTotal untuk dilakukan *scan*.

VirusTotal kemudian melakukan *scan file* berdasarkan *hash* yang diterima untuk menentukan apakah *file* tersebut mengandung *malware* atau tidak. Jika hasil *scan* menunjukkan *file* mengandung *malware*, maka sistem akan mengaktifkan fitur *active response* pada Wazuh untuk secara otomatis menghapus *file malware* tersebut dari sistem guna mencegah potensi kerusakan atau penyebaran. Namun, jika hasil *scan* tidak terdeteksi adanya *malware*, maka *file* tersebut dibiarkan dan proses akan kembali ketika Wazuh mengrimkan *hash file*.



Gambar 3. 2 Desain Wazuh dan VirusTotal API

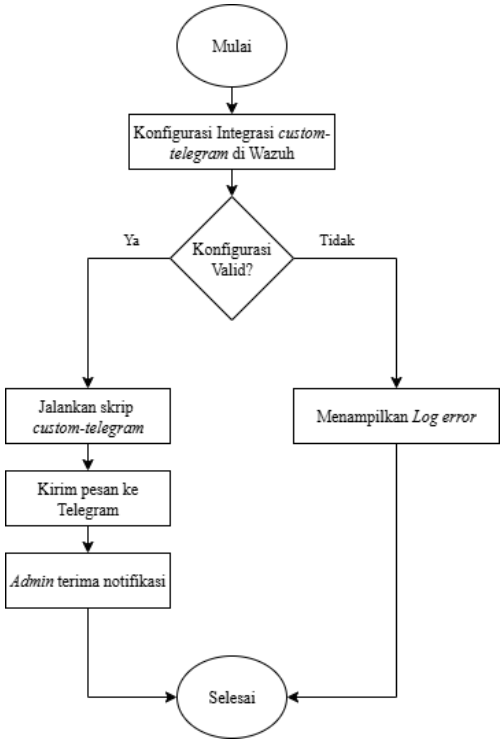
3.1.3.3 Desain Wazuh dan Telegram Bot

Berdasarkan Gambar 3.3, Wazuh diintegrasikan dengan aplikasi Telegram menggunakan skrip kustom bernama *custom-telegram*. Integrasi ini terdiri dari dua

komponen utama, yaitu skrip *shell custom-telegram* dan skrip Python *custom-telegram.py*. Skrip shell bertugas menentukan jalur eksekusi yang sesuai berdasarkan lokasi *file* dalam struktur direktori Wazuh, sehingga memudahkan pemanggilan skrip Python secara otomatis tanpa harus menyesuaikan konfigurasi secara manual. Sementara itu, skrip Python berfungsi untuk membaca data peringatan dari Wazuh yang dikirim dalam format JSON.

Proses dimulai dengan konfigurasi integrasi yang diatur dalam Wazuh, di mana *administrator* harus memasukkan token bot dan *chat* ID Telegram yang valid. Setelah konfigurasi dilakukan, sistem memeriksa apakah pengaturan tersebut valid atau tidak sebelum melanjutkan ke proses berikutnya.

Jika konfigurasi valid, Wazuh akan menjalankan skrip *custom-telegram* yang terdiri dari skrip *shell* untuk menentukan jalur eksekusi dan skrip Python yang membaca data peringatan dalam format JSON. Skrip ini memproses informasi penting seperti tingkat keparahan ancaman, deskripsi, dan nama *agent* yang terlibat, lalu mengirimkan pesan otomatis ke Telegram menggunakan API Telegram. Namun, jika konfigurasi tidak valid, sistem akan menampilkan *log error* sebagai tanda kegagalan pengaturan integrasi.



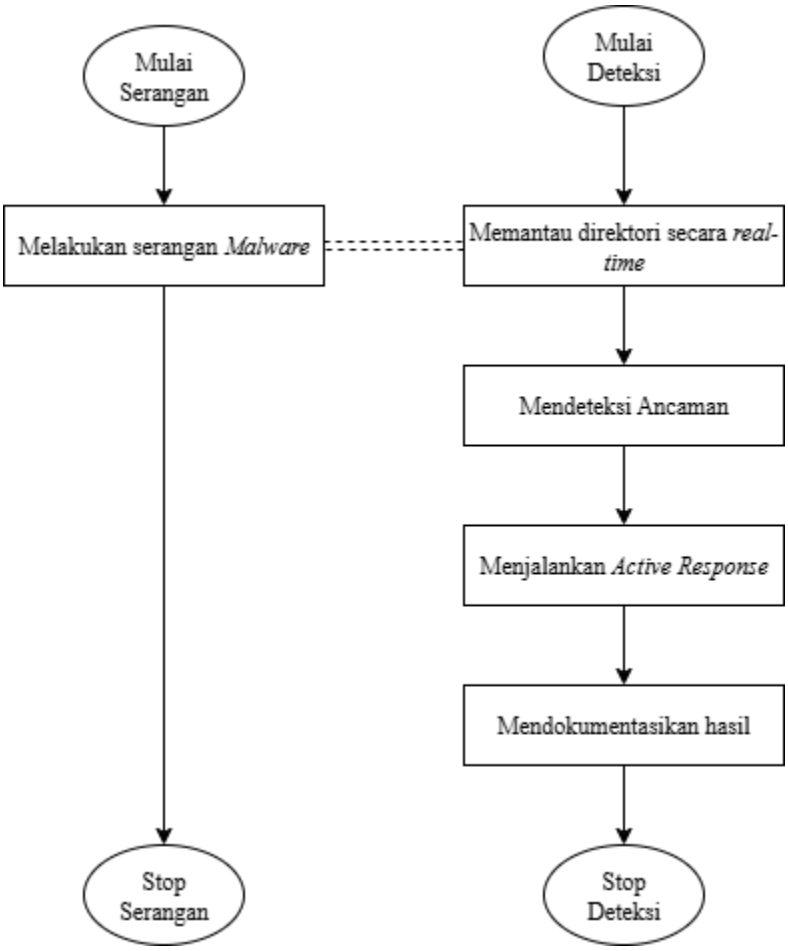
Gambar 3. 3 Desain Wazuh dan Telegram Bot

3.1.4 Pengujian Sistem

Skenario pengujian dalam penelitian ini, seperti yang digambarkan pada gambar 3.4 pengujian sistem, menggambarkan dua proses yang berjalan secara bersamaan, proses serangan dan proses deteksi dan respon. Proses diawali dengan memasukkan *file malware* EICAR ke direktori yang dipantau pada *agent*, yang secara langsung memicu alur deteksi dan respon. Sistem Wazuh kemudian akan mendeteksi *file* tersebut melalui FIM, menganalisisnya dengan VirusTotal, dan jika terkonfirmasi sebagai ancaman, sistem akan secara otomatis menjalankan *active response* untuk menghapus *file* tersebut. Seluruh rangkaian kejadian dari deteksi hingga mitigasi ini dicatat sebagai satu siklus pengujian untuk didokumentasikan.

Untuk memastikan validitas dan ketepatan data, setiap pengujian tersebut diulang sebanyak 15 kali untuk masing-masing *agent* (Ubuntu, Windows, dan CentOS). Pengulangan ini bertujuan untuk mendapatkan data yang representatif dengan meminimalkan dampak pengaruh dari gangguan sementara pada sistem.

Dengan memiliki 15 data per *agent*, ini bertujuan untuk menghitung nilai rata-rata waktu respon yang akurat serta menganalisis konsistensi dan stabilitas performa sistem.



Gambar 3. 4 Pengujian Sistem

3.1.5 Analisis Data

Dalam proses analisis data, membutuhkan pengukuran data dalam membandingkan hasil nilai parameter yang didapat. Pada penelitian ini, peniliti akan melakukan repetisi sebanyak 15 kali. Hal ini dilakukan demi mendapatkan data hasil yang lebih kuat untuk analisis dan interpretasi. Formula yang digunakan untuk pengukuran data untuk menghitung nilai rata-rata sebagai berikut.

$$\bar{x} = \frac{x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + .. + xn}{n} \quad (3.1)$$

Berdasarkan formula pengukuran diatas, dibawah ini merupakan penjelasan mengenai persamaan diatas, berikut keteranganya:

$\bar{x}$ = Nilai Rata-rata

$x(n)$ = Data Repitisi ke-n

n = Banyak data

Selain nilai rata-rata, penelitian ini juga menggunakan standar deviasi untuk mengukur sebaran atau variasi dari kumpulan data. Standar deviasi digunakan untuk mengukur konsistensi dan keandalan kinerja sistem, nilai yang kecil membuktikan kinerja sistem stabil dan dapat diprediksi, sementara nilai yang besar menunjukkan adanya variasi atau inkonsistensi. Ini memungkinkan analisis yang lebih mendalam mengenai seberapa andal sebuah sistem. Formula yang digunakan adalah rumus standar deviasi sampel:

$$s = \sqrt{\frac{\sum_{i=1}^{\eta}(x_i - \bar{x})^2}{n-1}} \quad (3.2)$$

s = Standar Deviasi Sampel

$\sum$ = Simbol penjumlahan (menjumlahkan semua hasil)

x_i = Setiap nilai data individual

$\bar{x}$ = Nilai rata-rata (mean) dari data

n = Jumlah total data

3.2 Spesifikasi Perangkat Penelitian

Dalam proses penelitian, diperlukan alat dan bahan yang dapat mendukung keberhasilan proses penelitian. Tabel 3.1 menampilkan tentang perangkat keras

yang digunakan serta spesifikasi perangkat, Tabel 3.2 berisi tentang perangkat lunak yang digunakan.

Tabel 3. 1 Informasi *Hardware*

No.	*Hardware*	Spesifikasi
1.	Wazuh *Server*	4 CPU(s), RAM 8 GB, Storage 50GB
2.	*Agent* #1 (Ubuntu)	2 CPU(s), RAM 6 GB, Storage 50GB
3.	*Agent* #2 (Windows)	4 CPU(s), RAM 8 GB, Storage 100GB
4.	*Agent* #3 (CentOS)	2 CPU(s), RAM 8 GB, Storage 300GB

Perbedaan spesifikasi perangkat keras pada setiap *agent* merupakan desain yang disengaja untuk memperkuat validitas penelitian dengan mensimulasikan lingkungan TI perusahaan Wika Beton yang realistis. Di dunia nyata, *server* memiliki peran dan beban kerja yang berbeda, sehingga memerlukan spesifikasi yang beragam. Dalam penelitian ini *agent* 2 (Windows) diberi spesifikasi lebih tinggi (4 CPU, 8 GB RAM) untuk mengakomodasi *overhead* sistem operasi yang lebih besar dan merepresentasikan peran *server* krusial seperti *Domain Controller* atau *server* aplikasi bisnis.

Sebaliknya, *agent* 1 (Ubuntu) dengan spesifikasi (2 CPU, 6 GB RAM) dapat disimulasikan sebagai *server* aplikasi standar. Sementara itu, *agent* 3 (CentOS) memiliki RAM besar (8 GB) dan penyimpanan yang sangat luas (300 GB) untuk mencerminkan perannya sebagai *server database* atau *server* penyimpanan *file*, yang tidak selalu butuh banyak CPU tetapi sangat bergantung pada memori dan ruang penyimpanan.

Selain alasan fungsional, penentuan spesifikasi ini juga didasarkan pada perbedaan arsitektur fundamental dalam pembuatan proses. Sistem operasi Linux (Ubuntu dan CentOS) menggunakan mekanisme *fork()* yang sangat ringan untuk membuat salinan proses, sehingga dapat beroperasi secara efektif hanya dengan 2 CPU (Zhao dkk., 2021). Sebaliknya, Windows menggunakan *CreateProcess*() (*spawn*) yang lebih berat karena harus membangun proses baru dari awal, yang

menciptakan *overhead* signifikan. Oleh karena itu, *agent* Windows dibekali spesifikasi lebih tinggi (4 CPU, 8 GB RAM) untuk mengimbangi beban kerja tersebut dan memastikan perbandingan performa tetap relevan.

Tabel 3. 2 Informasi *Software*

No.	*Software*	Spesifikasi
1.	*Operating System* Wazuh Manager	Amazon Linux 2023
2.	*Operating System Agent*#1 (Ubuntu)	Ubuntu 20.04.6 LTS
3.	*Operating System Agent*#2 (Windows)	Microsoft Windows *Server* 2022
4.	*Operating System Agent*#3 (CentOS)	CentOS Linux 7.9
5.	*Malware*	*Version* EICAR 1.0
6.	Wazuh	*Version* 4.11.2
7.	VirusTotal API	*Version* VirusTotal API v3
8.	Telegram Bot	*Version* 11.9.0

Lingkungan simulasi penelitian terdiri dari sejumlah empat VM, tiga komponen utama sistem keamanan Wazuh, VirusTotal API, dan Telegram Bot sebagai sistem keamanan *Active Response* dan *agent* sebagai pihak yang diserang oleh *attacker*.