Volume 6, No.1 2025

JISE

Journal of Internet and Software Engineering

ISSN 2797-9016

https://ugm.id/jise

The journal published by
Department of Electrical Engineering and Informatics
Vocational College, Universitas Gadjah Mada

JISE
Journal of Internet and Software Engineering
Volume 6, No.1 2025

EDITORIAL TEAM

Journal of Internet and Software Engineering (JISE)

https://ugm.id/jise

The journal published by
Department of Electrical Engineering and Informatics
Vocational College, Universitas Gadjah Mada
Sekip unit III, Caturtunggal, Terban,
Kec. Gondokusuman, Kab. Sleman, D.I. Yogyakarta 55281

JISE
Journal of Internet and Software Engineering
Volume 6, No.1 2025

https://ugm.id/jise

The journal published by
Department of Electrical Engineering and Informatics
Vocational College, Universitas Gadjah Mada

Analisis Perbandingan Penggunaan Metode *Tunneling Cloud Virtual Private Network* dan WireGuard *Virtual Private Network* pada Implementasi Infrastruktur *Hybrid Cloud*

Rusdi Hermawan[1], Yuris Mulya Saputra[1,*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
rusdi.h@mail.ugm.ac.id
[*]Korespondensi: ym.saputra@ugm.ac.id;

Abstract – The continuous advancement of server infrastructure technology drives organizations to migrate from on-premise to public cloud systems, despite data security and cost challenges. Hybrid cloud infrastructure becomes essential for seamless integration, minimizing downtime, and maintaining operational consistency. This implementation often uses tunneling methods, leveraging VPN technology for secure data transfer between on-premise servers and public cloud services. This research compares two VPN tools: WireGuard (open-source) and Google Cloud Platform's (GCP) Cloud VPN (enterprise solution) in hybrid-cloud contexts. The study measures performance parameters such as latency, throughput, connection stability, ease of configuration, and service costs to evaluate each tool's suitability. Findings show that WireGuard VPN has a throughput advantage of 676.67% and reduces average total latency by 97.66% in initial tests, indicating superior performance compared to* Classic Cloud *VPN.

Keywords – Hybrid Cloud, Tunneling, VPN, WireGuard,* Classic Cloud *VPN

Intisari – Perkembangan teknologi dalam infrastruktur *server* terus maju signifikan setiap tahun. Organisasi dan perusahaan cenderung beralih ke sistem yang lebih kompleks dan terpusat, mendorong migrasi infrastruktur *server* dari *on-premise* ke *public cloud*. Banyak migrasi dilakukan bertahap karena keterbatasan biaya dan keamanan data. Dalam konteks ini, infrastruktur *hybrid cloud* penting untuk integrasi yang mulus, meminimalkan *downtime*, dan menjaga konsistensi operasional. *Hybrid cloud* sering menggunakan *tunneling* untuk menghubungkan *server on-premise* dengan *public cloud*, menggunakan teknologi *Virtual Private Network* (VPN) untuk jalur terenkripsi yang aman. Penelitian ini membandingkan dua alat VPN: WireGuard sebagai solusi *open-source* dan Cloud VPN dari Google Cloud Platform (GCP) sebagai solusi *enterprise*, dalam konteks *hybrid-cloud*. Penelitian fokus pada pengukuran kinerja parameter seperti *latency*, *throughput*, kestabilan koneksi, kemudahan konfigurasi, dan biaya layanan untuk memahami kecocokan masing-masing alat dalam skenario implementasi tertentu. Hasilnya menunjukkan bahwa *throughput* WireGuard VPN unggul 676,67%, dan rata-rata *latency* WireGuard VPN berkurang hingga 97,66% pada uji pertama, menunjukkan kinerja lebih baik dibandingkan Classic Cloud VPN.

Kata kunci – *Hybrid Cloud*, *Tunneling*, VPN, WireGuard, Classic Cloud VPN

I. PENDAHULUAN

Ketatnya persaingan antarperusahaan dan tuntutan akan perkembangan teknologi yang begitu pesat menjadi sebuah aspek yang mendasari pentingnya pengembangan infrastruktur Teknologi Informasi (TI) dalam ranah korporasi. Infrastruktur IT dalam suatu organisasi atau perusahaan mencakup *software*, *hardware,* dan *service* [1]. Aspek yang sangat berpengaruh dalam kelangsungan operasional sebuah perusahaan saat ini sangat bergantung pada layanan yang disediakan, dalam konteks ini adalah layanan digital yang dimiliki perusahaan. Layanan ini biasanya berjalan pada sebuah perangkat *server* yang dimiliki oleh perusahaan. Setiap perusahaan memiliki pendekatan yang berbeda dalam pengelolaan *server* yang mereka miliki.

Perkembangan teknologi *server* yang semakin pesat telah memunculkan paradigma baru dalam pengelolaan sumber daya komputasi yang saat ini dikenal sebagai teknologi *cloud computing*. Hadirnya teknologi *cloud computing* sudah menjadi salah satu paradigma komputasi yang membarui lanskap teknologi isu dalam beberapa tahun terakhir. *Cloud computing* telah menciptakan peluang baru bagi organisasi dan pengguna individu dalam hal skalabilitas, fleksibilitas, dan efisiensi operasional dengan menyediakan akses ke sumber daya komputasi dan penyimpanan data melalui internet [2]. Dengan adanya infrastruktur *cloud* ini, perusahaan menjadi lebih mudah dalam memanajemen layanan digital yang mereka miliki karena layanan *cloud* memberikan fleksibilitas kepada pengelola untuk memanajemen layanan yang berjalan di dalamnya.

Dalam konteks penelitian ini, jenis *cloud computing* dibagi menjadi tiga, yaitu *public cloud*, *private cloud*, dan *hybrid cloud*. *Public cloud* mengacu pada sebuah layanan *cloud* yang diberikan oleh *provider* besar seperti Google Cloud Platform, AWS, dan Microsoft Azure. Kemudian, aksesibilitas *public cloud* dapat terhubung secara *public* melalui internet, sehingga *user* dapat mengakses melalui internet yang bersifat *public*. Di sisi lain, *private cloud* merupakan layanan *cloud* yang memiliki aksesibilitas terbatas, sehingga tidak semua *user* internet dapat mengaksesnya dengan jaringan internet *public*. Hanya *user* yang terhubung dengan jaringan perusahaan atau organisasi saja yang dapat mengaksesnya. Namun, dalam beberapa skenario, terdapat infrastruktur *cloud* yang menggabungkan penggunaan *public cloud* dan *private cloud*, yang dikenal

sebagai *hybrid cloud*. *Hybrid cloud* memungkinkan *user* untuk memanfaatkan masing-masing kelebihan dari *public* dan *private cloud* seperti keamanan dan kontrol dari *private cloud* serta skalabilitas dan elastisitas dari *public cloud* [3].

Pada umumnya, dalam membangun sebuah infrastruktur *hybrid cloud* memerlukan sebuah interkoneksi untuk menghubungkan antara *public cloud* dengan *private cloud* secara aman. Secara umum untuk membuat sebuah interkoneksi digunakan *tools* VPN atau menggunakan protokol *Internet Protocol Security* (IPsec) agar komunikasi jaringan antarinfrastruktur *cloud* berjalan secara aman (*secure*). Namun, untuk membuat sebuah interkoneksi pada sebuah infrastruktur *hybrid cloud* juga perlu memperhatikan aspek kinerja koneksi dari jalannya komunikasi antara *public cloud* dengan *private cloud* melalui VPN. Kinerja ini sangat memengaruhi skalabilitas layanan yang digunakan, sehingga pemilihan *tools* yang sesuai untuk menghubungkan *public cloud* dan *private cloud* melalui interkoneksi menjadi sangat vital.

Berdasarkan latar belakang tersebut, penelitian ini mengangkat dan menganalisis perbandingan *tools* VPN yang bersifat berbayar (*enterprise*) dan gratis (*open source*), sebagai pertimbangan seorang *cloud architect* dalam mengimplementasikan infrastruktur *hybrid cloud* dengan menyesuaikan kondisi infrastruktur *cloud* perusahaan yang sudah ada. Perbandingan ini difokuskan pada pengukuran kinerja parameter uji seperti *latency*, *throughput*, kestabilan koneksi, kemudahan konfigurasi, dan biaya layanan, untuk memberikan pemahaman yang mendalam tentang kecocokan masing-masing *tools* untuk skenario implementasi tertentu.

II. DASAR TEORI

Penelitian ini mengambil beberapa referensi yang memiliki kesamaan dalam pembahasan infrastruktur *hybrid cloud* dan analisis penggunaan VPN. Penulis mengambil referensi pertama dari penelitian dengan judul "*A Performance Comparison of* WireGuard *and* OpenVPN". Penelitian tersebut berisi pembahasan mengenai perbandingan kinerja antara *tools* VPN dengan menggunakan OpenVPN dengan WireGuard VPN. Dalam melakukan perancangan *deployment* tiap *tools* VPN, menggunakan bantuan *tools* otomasi menggunakan Ansible. Penggunaan Ansible pada penelitian ini digunakan untuk melakukan efisiensi waktu saat melakukan *deployment* ulang pada infrastruktur *cloud* yang berbeda. Dalam penelitian ini, infrastruktur yang digunakan mencakup pengguanaan *public cloud* dari AWS dengan menggunakan empat *Virtual Machine* (VM) dengan empat *region* yang berbeda, serta diimplementasikan pada VM lokal dengan OS Ubuntu dan Centos. Keseluruhan penelitian ini membahas secara jelas analisis perbandingan kinerja *tools* antara Open VPN dan WireGuard VPN dengan parameter uji mencakup *throughput* dan sumber daya komputasi yang berfokus pada parameter CPU *usage*. Konklusi dari penelitian ini menjelaskan bahwa hasil perbandingannya memberikan analisis bahwa *tools* WireGuard VPN lebih unggul dibandingkan dengan OpenVPN [4].

Penelitian selanjutnya terdapat bahasan yang hampir sama seperti dengan penelitian sebelumnya yang membahas perbandingan terkait analaisis kinerja *tools* VPN. Pada yang berjudul "*Performance Analysis of VPN Gateways*" yang lebih difokuskan pada analisis kinerja dari *gateway* VPN dengan membandingkan perangkat lunak dari OpenVPN, Linux IPsec, dan WireGuard VPN. Penelitian ini dilakukan untuk mengevaluasi arsitektur perangkat lunak VPN yang paling efektif dalam mencapai kinerja yang optimal. Hasil penelitian ini menunjukkan bahwa WireGuard merupakan implementasi perangkat lunak VPN yang paling efektif dari segi arsitektur dan memiliki hasil pengujian *throughput* tertinggi dari ketiga implementasi VPN yang uji kinerjanya. Penelitian ini juga mengidentifikasi bahwa *bottleneck* utama dalam penskalaan perangkat lunak VPN terletak pada struktur data dan sinkronisasi *multi-core*, tetapi masalah tersebut dapat diatasi dengan mengadopsi arsitektur berbasis *pipelining* dan *message passing* [5].

Dalam penelitian yang masih membahas di lingkungan VPN, terdapat penelitian dengan judul "*Performance Evaluation of Software Routers with VPN Features*". Penelitian tersebut membahas terkait analisis perangkat lunak *router* dengan fitur VPN berbasis *open source* menggunakan Quagga dan StrongSwan. Penelitian ini bertujuan memvalidasi fungsionalitas dalam lingkungan *real case* dan mengukur kinerja algoritma enkripsi dan *hash* yang didukung oleh StrongSwan untuk memberikan konfigurasi VPN optimal. Hasil dari penelitian ini menunjukkan bahwa integrasi Quagga dan StrongSwan dapat meningkatkan ketahanan terhadap kegagalan koneksi dengan AES GCM sebagai pilihan terbaik untuk kinerja enkripsi yang optimal [6].

Selanjutnya terdapat referensi yang memberikan perspektif langsung pada sebuah implementasi VPN dalam infrastruktur *hybrid cloud*. Penelitian ini berjudul "*CloudJoin: Experimenting at scale with Hybrid Cloud Computing*". Penelitian ini membahas tentang CloudJoin, yaitu sebuah pendekatan transformatif untuk memperluas infrastruktur penelitian komputasi dengan menggabungkan CloudLab dan Google Cloud Platform. CloudJoin memungkinkan eksperimen berskala besar tanpa perlu melakukan perubahan pada infrastruktur yang ada. Penelitian ini menunjukkan bagaimana mengintegrasikan kedua infrastruktur tersebut melalui VPN dan *cloud monitoring tools* untuk mendukung skala eksperimen yang lebih besar. Hasil analisis menunjukkan bahwa CloudJoin memberikan akses mudah ke perangkat keras khusus dan layanan *cloud*, serta memungkinkan eksperimen berjalan dengan lancar tanpa perlu infrastruktur tambahan. Dengan demikian, dapat disimpulkan bahwa CloudJoin adalah pendekatan yang efektif untuk mendukung eksperimen berskala besar dalam penelitian sistem komputer [7].

Referensi berikutnya mengambil penelitian dengan judul "Integrasi *Server On-Premise* dengan *Server Cloud* Menggunakan Cloud VPN dan Mikrotik IPSEC untuk Peningkatan Keamanan Koneksi". Pada penelitian ini dilakukan konfigurasi dengan menggunakan metode MikroTik IPsec di *server on-premise* dan pengaturan VPN Tunnel IKEv2 pada layanan Google Cloud, penelitian ini berhasil mengamankan koneksi antara kedua infrastruktur. Uji coba praktis menunjukkan efektivitas solusi keamanan yang diimplementasikan dalam lingkungan infrastruktur *hybrid*, dengan mengurangi potensi risiko seperti *sniffing*, serangan *port scanning*, *brute force*, DDOS, dan ancaman siber lainnya. Hasil penelitian ini memberikan bukti konkret bahwa penggunaan MikroTik IPsec dan VPN Tunnel IKEv2 pada layanan Google Cloud dapat menjadi langkah yang efektif dalam mengamankan komunikasi antara *server on-premise* dan *server cloud* [8].

Pada referensi berikutnya ditinjau penelitian dengan judul "Desain dan Implementasi *Hybrid Cloud Computing* Sebagai Infrastruktur untuk Analisis *Big Data* Menggunakan *Analytic Hierarchy Process* (AHP)". Penelitian ini bertujuan untuk mengimplementasikan *hybrid cloud computing* sebagai infrastruktur untuk analisis *big data* dengan menggunakan metode *Analytic Hierarchy Process* (AHP) untuk minimalisasi biaya dalam pemilihan *public cloud*. Tahapan dalam penelitian ini meliputi pengumpulan data biaya dari berbagai *public cloud provider*, praproses data untuk konsistensi, pengembangan sistem dengan fitur aplikasi AHP, desain arsitektur sistem, dan implementasi AHP dalam bahasa pemrograman Java. Hasil analisis AHP menunjukkan rekomendasi *public cloud* terbaik untuk digunakan dalam infrastruktur *hybrid cloud*, dengan Digital Ocean sebagai pilihan terpilih [9].

Referensi berikutnya mengambil jurnal penelitian dengan judul "*A Comparative Research on VPN Technologies on Operating System for Routers*". Pada penelitian ini mengulas tentang perbandingan dari tiga *tools* VPN meliputi WireGuard VPN, IPsec, dan SSL-VPN, dengan parameter yang digunakan adalah kompleksitas rancangan infrastruktur VPN dan *throughput*. Penelitian ini dilakukan dengan merancang *router* khusus dan menguji *throughput* masing-masing teknologi VPN menggunakan iPerf3 dalam model jaringan Client-to-Site. Hasil penelitian menunjukkan bahwa WireGuard memiliki *throughput* tertinggi dan paling efisien dalam penggunaan sumber daya CPU dibandingkan dengan SSL-VPN yang memiliki kinerja paling rendah, dan IPsec yang meskipun lebih baik dari SSL-VPN, tetapi masih lebih kompleks dan kurang efisien dibandingkan WireGuard [10].

Referensi selanjutnya mengambil dari jurnal berjudul "*IPSec: Performance Analysis in IPv4 and IPv6*". Penelitian tersebut bertujuan untuk menganalisis kinerja *throughput* protokol IPSec pada jaringan IPv4 dan IPv6 menggunakan berbagai algoritma kriptografi yang direkomendasikan. Penelitian dilakukan dengan mengukur kinerja *throughput* untuk algoritma enkripsi terautentikasi seperti AES-GCM dan AES-CCM, algoritma enkripsi seperti AES-CBC, AES-CTR, dan 3DES, serta algoritma autentikasi seperti SHA1, SHA2, dan XCBC. Hasil penelitian menunjukkan bahwa algoritma AES-GCM memberikan kinerja *throughput* yang lebih baik dibandingkan dengan algoritma kriptografi lainnya yang digunakan dalam implementasi protokol IPSec pada jaringan IPv4 dan IPv6 [11].

Referensi selanjutnya mengambil tinjauan dari jurnal dengan judul "*Impact of IPSec on Real Time Applications in IPv6 and 6to4 Tunneled Migration Network*". Penelitian ini bertujuan untuk menyelidiki dampak implementasi IPSec pada aplikasi *real-time* dalam jaringan IPv6 dan jaringan migrasi *tunneling* 6to4. Penelitian ini menggunakan metode simulasi dengan OPNET Simulator versi 14.5, melalui tiga skenario berbeda yaitu jaringan IPv6 tanpa IPSec, IPv6 dengan IPSec, dan migrasi *tunneling* 6to4 dengan IPSec. Hasilnya menunjukkan bahwa meskipun IPSec berhasil memberikan keamanan yang diperlukan, terdapat peningkatan *delay*, *jitter*, dan *packet drop rate* yang signifikan, yang mempengaruhi kinerja aplikasi *real-time* [12].

Referensi berikutnya penulis meninjau jurnal judul penelitian "*Security vs Bandwidth: Performance Analysis Between IPsec and OpenVPN in Smart Grid*", memaparkan penelitian terkait analisis pengaruh enkripsi terhadap *bandwidth* antara IPsec dengan OpenVPN. Skenario pengujian dilakukan dengan mengatur koneksi IPsec dan OpenVPN melalui jaringan *public* LTE menggunakan router industri dan menganalisis *overhead* yang ditambahkan oleh masing-masing metode enkripsi terhadap paket data. Hasil penelitian menunjukkan bahwa IPsec menambahkan rata-rata 64 *byte* per paket dan OpenVPN menambahkan rata-rata 42 *byte* per paket, dengan IPsec membutuhkan sekitar 23% lebih banyak *bandwidth* dibandingkan OpenVPN. Penelitian ini memberikan wawasan penting tentang kebutuhan *bandwidth* dalam implementasi metode enkripsi pada jaringan komunikasi *smart grid* [13].

Referensi selanjutnya merujuk pada penelitian berjudul *"CloudJoin: Experimenting at Scale with Hybrid Cloud Computing"*. Penelitian ini menekankan pentingnya pengembangan *testbed* eksperimen yang terintegrasi dan skalabel antara CloudLab dan Google Cloud Platform (GCP), tanpa memerlukan perubahan mendasar pada infrastruktur yang sudah ada. Penelitian ini menghasilkan pendekatan integrasi infrastruktur eksperimen melalui mekanisme *peer-to-peer Virtual Private Network* (VPN) menggunakan StrongSwan, serta pemanfaatan alat *observability* (alat *monitoring*) seperti Google Stackdriver yang diintegrasikan dengan agen BindPlane dan Collectd. Tujuan utama dari penelitian ini adalah untuk menawarkan solusi *hybrid cloud* yang mampu menjawab tantangan skalabilitas dan keterbatasan sumber daya dalam eksperimen sistem komputasi skala besar. Manfaat yang dihadirkan mencakup penyediaan pendekatan riset komputasi yang lebih inklusif, adaptif, serta berorientasi masa depan (*future-oriented*),

terutama dalam menjawab dinamika kebutuhan sumber daya yang terus meningkat di lingkungan penelitian akademik [14].

Pada referensi terakhir yang ditinjau penulis, yaitu penelitian berjudul *"Comparative Analysis of Optimizing Hybrid Cloud Environments Using AWS, Azure, and GCP"*, dipaparkan bahwa pemilihan penyedia layanan *public cloud* yang tepat merupakan aspek krusial dalam merancang infrastruktur *hybrid cloud* yang optimal. Permasalahan utama yang diangkat dalam penelitian ini berkaitan dengan meningkatnya kebutuhan organisasi untuk mengadopsi arsitektur *hybrid cloud* demi mencapai fleksibilitas, efisiensi biaya, dan peningkatan skalabilitas. Namun, proses adopsi ini tidak dapat dilakukan secara langsung tanpa adanya kajian komparatif terhadap masing-masing *platform cloud*, guna memastikan kesesuaian dengan karakteristik dan kebutuhan spesifik setiap organisasi. Hasil penelitian ini memberikan solusi aplikatif bagi seorang *solution architect* dalam menyelaraskan strategi penerapan layanan cloud dengan kebutuhan organisasi. Dengan demikian, studi ini menjadi kontribusi signifikan bagi akademisi di bidang teknologi informasi maupun praktisi industri, khususnya *solution architect*, dalam memahami serta mengoptimalkan penerapan infrastruktur *hybrid cloud* sebagai fondasi transformasi digital yang berkelanjutan [15].

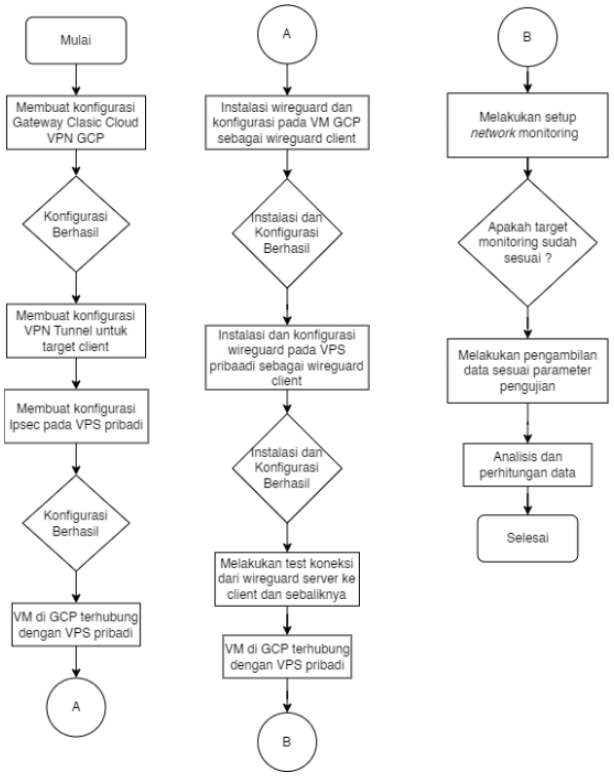
Dari sepuluh referensi yang diambil, penelitian ini memiliki keterbaruan dalam konteks jenis *tools* yang dibandingkan, di mana dalam penelitian ini penulis membandingkan Classic Cloud VPN dari produk GCP yang belum pernah dibandingkan dengan WireGuard VPN. Selanjutnya, terdapat perbedaan dalam implementasi yang diterapkan. Dalam konteks penelitian ini, penulis menerapkan implementasi pada infrastruktur *hybrid cloud*. Terakhir, terdapat keterbaruan dalam skenario pengujian yang dilakukan. Dalam penelitian ini, penulis melakukan skenario dengan menggunakan metode *rsync* untuk menguji performa interkoneksi gateway VPN dari kedua *tools* VPN yang diimplementasikan pada infrastruktur *hybrid cloud*. Detail skenario yang dilakukan adalah perintah *rsync* digunakan untuk mengirim *file dataset* dengan berbagai ukuran, mulai dari kecil hingga besar. Pengiriman dilakukan dengan skenario pengiriman dari *private cloud* ke *public cloud*, *public cloud* ke *private cloud*, dan terakhir adalah pengiriman *file* antar lingkungan *public cloud*.

III. METODOLOGI

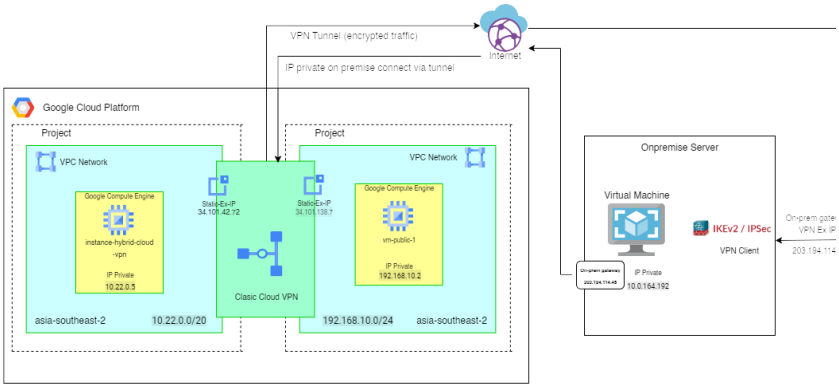
Dalam melaksanakan penelitian ini terdapat beberapa tahapan penelitian yang dilakukan oleh penulis. Detail tahapan penelitian dapat dilihat dalam diagram alir pada Gambar 1.

Gambar 1. Diagram alir penelitian

Setelah menentukan tahapan penelitian yang dikerjakan, penulis melanjutkan untuk melakukan tahapan perancangan sistem. Detail tahapan perancangan sistem yang dilakukan dapat dilihat pada Gambar 2.



Gambar 2. Diagram alir perancangan sistem



Gambar 3. Topologi *Hybrid Cloud Classic Cloud* VPN

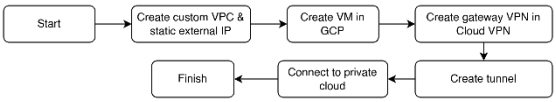
A. Desain Infrastruktur *Hybrid Cloud*

Penelitian ini memiliki dua rancangan infrastruktur *hybrid cloud* dengan perbedaan dalam metode *tunneling* yang digunakan. Pada infrastruktur yang pertama diimplementasikan metode *tunneling* menggunakan Classic Cloud VPN. Classic Cloud VPN merupakan fitur VPN yang memang diberikan Google Cloud untuk kebutuhan interkoneksi yang salah satu tujuanya dibuat untuk diimplementasikan pada arsitektur *hybrid cloud*. Protokol yang dipakai pada Classic Cloud VPN ini menggunakan protokol IPSec/IkeV2. Berikut adalah detail desain topologi pada rancangan infrastruktur *hybrid cloud* menggunakan Classic Cloud VPN tertera pada Gambar 3.

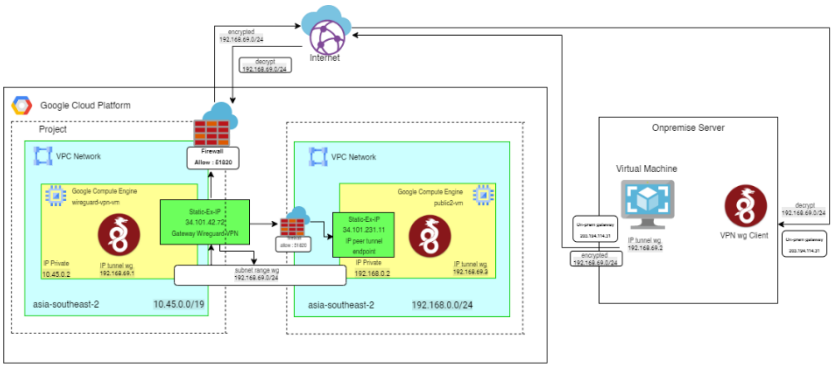
Berikutnya pada infrastruktur kedua diimplementasikan metode *tunneling* menggunakan WireGuard VPN. WireGuard VPN memiliki protokol independen yang juga dinamai sebagai protokol WireGuard. Protokol ini memiliki keunggulan dalam hal kecepatan koneksi, lalu menurut beberapa sumber protokol ini juga ringan jika diimplementasikan di dalam sebuah *server*. Berikut merupakan detail desain arsitektur *hybrid cloud* menggunakan WireGuard VPN tertera pada Gambar 4.

B. *Setup* Infrastruktur *Hybrid Cloud* VPN GCP

Tahap awal dalam membangun infrastruktur *hybrid* menggunakan Cloud VPN di Google Cloud Platform (GCP) dimulai dengan pembuatan *Virtual Private Cloud* (VPC) sebagai dasar pemisahan jaringan untuk layanan Google Compute Engine. Pada tahap ini juga dilakukan pembuatan *static external public IP address* yang akan digunakan sebagai *endpoint* atau *gateway* untuk koneksi Cloud VPN. Setelah konfigurasi *basic networking* selesai, proses dilanjutkan dengan pembuatan *virtual machine* (VM) yang dikaitkan dengan subnet sesuai dengan topologi *custom* VPC yang telah dirancang. Selanjutnya, dilakukan konfigurasi Cloud VPN Gateway menggunakan alamat *static IP* yang telah disiapkan sebelumnya. Tahapan ini diikuti dengan pembuatan VPN *tunnel* yang menghubungkan *gateway* di sisi GCP dengan jaringan di sisi *private cloud*. Konfigurasi terakhir mencakup pengaturan klien VPN pada sisi *private cloud* agar dapat menjalin koneksi terenkripsi secara stabil dengan jaringan di GCP. Detail arsitektur *setup* dapat dilihat pada Gambar 5.



Gambar 5. Diagram alir setup infrastruktur *hybrid cloud* VPN GCP

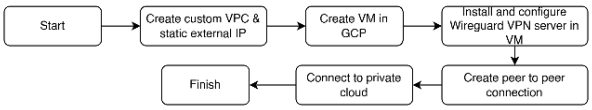


Gambar 4. Topologi *Hybrid Cloud* WireGuard VPN

C. *Setup* Infrastruktur *Hybrid* WireGuard VPN

Perancangan infrastruktur *hybrid cloud* menggunakan metode *tunneling* WireGuard VPN pada tahap awal memiliki alur yang serupa dengan penerapan Classic Cloud VPN, khususnya dalam hal penyiapan *project* di Google Cloud Platform (GCP). Langkah-langkah awal yang dilakukan meliputi aktivasi *billing account*, pembuatan VPC, serta penyusunan *virtual machine* (VM) di lingkungan *public cloud* GCP. Namun, perbedaan utama terletak pada pendekatan implementasi VPN-nya. Pada arsitektur WireGuard, VM di GCP berfungsi langsung sebagai *server* VPN. Ini berbeda dengan Classic Cloud VPN, di mana layanan VPN disediakan secara terpisah dari VM melalui layanan terkelola milik GCP. Dalam skema WireGuard, komunikasi antar jaringan dilakukan melalui alokasi *private* IP *range* yang didefinisikan secara manual, dengan dukungan fitur IP *forwarding*. Dengan demikian, layanan WireGuard berjalan langsung di dalam VM, dan menggunakan *external IP address* dari VM tersebut sebagai *gateway* komunikasi VPN.

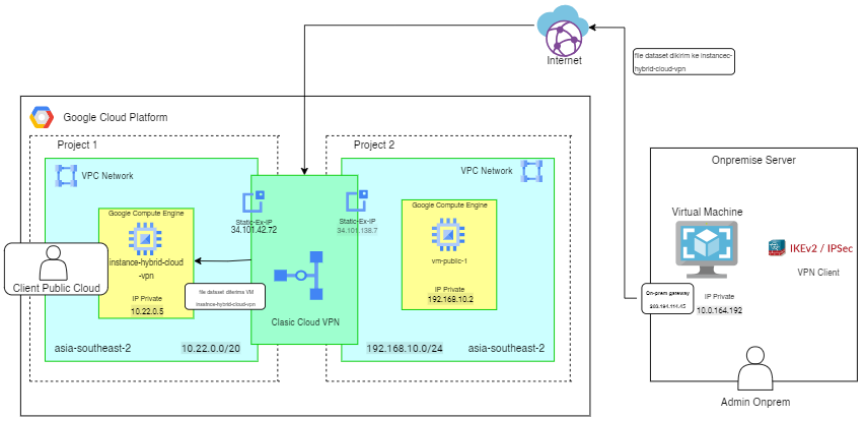
Setelah konfigurasi dasar WireGuard *server* pada VM GCP selesai dilakukan, langkah selanjutnya adalah melakukan konfigurasi pada sisi klien, yaitu di lingkungan *private cloud*. Konfigurasi pada sisi client secara umum mengikuti struktur yang sama seperti pada *server*. Pada kedua sisi, dilakukan pertukaran informasi melalui *peer configuration*, yang memerlukan *public key* dari masing-masing VM serta informasi subnet *private IP range* yang telah ditentukan sebelumnya di sisi *server*. Ilustrasi lengkap mengenai setup infrastruktur *hybrid* dengan WireGuard VPN dapat dilihat pada Gambar 6.



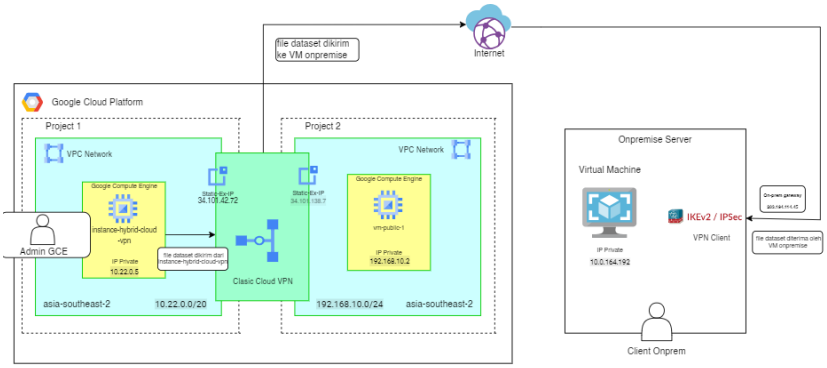
Gambar 6. Diagram alir *setup* infrastruktur *hybrid* WireGuard VPN

D. Skenario Pengujian

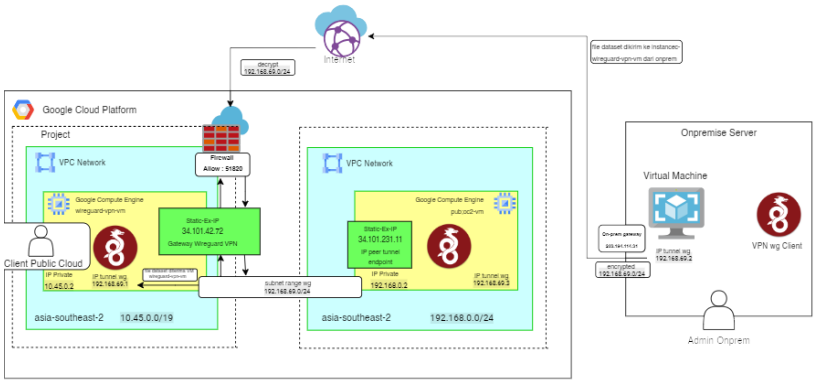
Pada pengerjaan penelitian dilakukan pengujian dengan berfokus pada pengukuran parameter QoS dengan target setiap *gateway* VPN dari kedua infrastruktur *hybrid cloud*. Pengujian dilakukan untuk melakukan pengambilan data metrics terkait data *bytes*, data *packet*, dan detail waktu dari pengiriman kedua data tersebut. Pengujian pertama dilakukan dengan melakukan skenario pengiriman *file dataset* dari lingkungan *private cloud* ke *public cloud*. Lalu selanjutnya pengiriman dilakukan dari lingkungan *public cloud* ke *private*



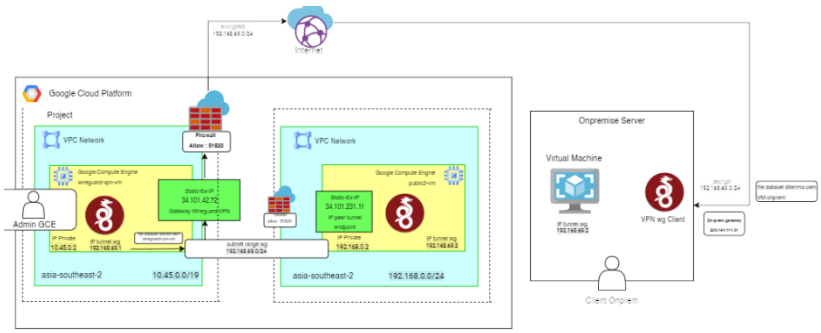
Gambar 7. Skenario pengujian 1 Cloud VPN



Gambar 8. Skenario pengujian 2 Cloud VPN



Gambar 9. Skenario pengujian 1 WireGuard VPN



Gambar 10. Skenario pengujian 2 WireGuard VPN

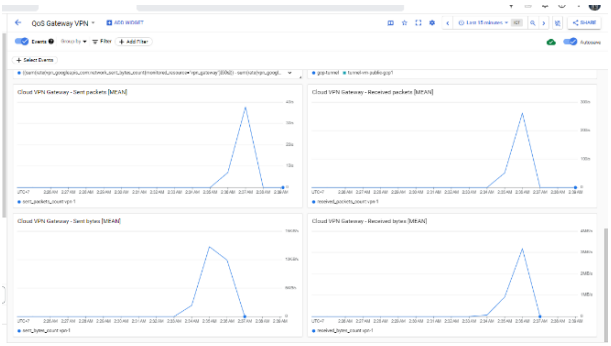
cloud. Secara detail alur skenario pengujian dapat dilihat pada Gambar 7 dan 8.

Pengujian pada lingkungan infrastruktur *hybrid cloud* yang menggunakan WireGuard VPN sebagai alat interkoneksi untuk menghubungkan *private cloud* dengan *public cloud* secara keseluruhan memiliki skenario yang sama dengan pengujian yang dilakukan pada infrastruktur *hybrid cloud* yang menggunakan Classic Cloud VPN dari GCP. Detail skenario pengujian dapat dilihat dari Gambar 9 dan 10.

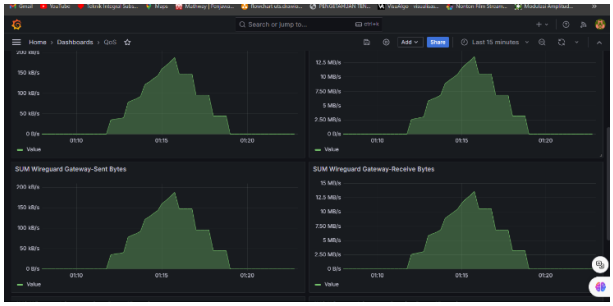
Skenario pengujian yang dilakukan berfokus pada pengukuran *traffic* yang masuk melalui *gateway* Cloud VPN. Tujuan dari pengukuran *traffic* adalah untuk mengetahui seberapa cepat kinerja interkoneksi pada infrastruktur *hybrid cloud*. Selain itu, pengujian ini juga bertujuan untuk mengetahui seberapa besar beban *traffic* yang mampu di tangani oleh VPN, melalui kecepatan data yang masuk pada *gateway* VPN. Skenario pengujian dilakukan dengan mengirim *file* secara berulang menggunakan variasi ukuran *file dataset* yang berbeda, mulai dari ukuran kecil, sedang, hingga ukuran besar. Pengujian juga dilakukan dengan menjalankan skenario pengiriman *file dataset* dari lingkungan cloud yang berbeda. Detail pengujian dapat dilihat pada Gambar 7 hingga 10, yang memberikan informasi terkait alur pengiriman *file dataset*. Pengujian dimulai dengan mengirim *file dataset* dari *private cloud* ke *public cloud*, kemudian dilanjutkan dengan mengirim dari lingkungan *public* ke *private*. Pengiriman *file* yang dilakukan menggunakan metode *remote sync* atau disingkat *rsync*, dengan metode ini memungkinkan *cloud* melakukan pengirirman *file* secara *remote*.

E. Perhitungan *Quality of Service* (QoS)

Setelah mengirim *file* menggunakan metode *rsync*, tahapan terakhir adalah pengambilan data *traffic* interkoneksi VPN pada *dashboard monitoring*. Fungsi dari monitoring ini adalah mencatat data metrik terkait data dan paket yang dikirim dan diterima melalui *gateway* VPN, yang nantinya digunakan untuk analisis hasil perbandingan. Detail gambar *dashboard* visualisasi data dapat dilihat pada Gambar 11 dan 12.



Gambar 11. *Dashboard cloud* monitoring GCP



Gambar 12. *Dashboard* WireGuard VPN

Selanjutnya, proses pengambilan data dilakukan dengan masuk ke *dashboard* monitoring dan mengunduh data CSV dari salah satu visualisasi data yang ditampilkan di dalam *dashboard*. Contoh pada salah satu data mentah dari pengambilan nilai *bytes* dan *packet* saat dilakukan pengujian tertera pada Tabel 1 dan 2.

Tabel 1. *Receive bytes* Cloud VPN

Waktu	*Receive Bytes*
6/24/2024 06:12:00	34,225
6/24/2024 06:13:00	6490566,817
6/24/2024 06:14:00	11712539,69
6/24/2024 06:15:00	149019,4
Total	18352160,13
Rata-Rata	4588040,033

Tabel 2. *Receive bytes* WireGuard VPN

Waktu	*Receive Bytes*
6/24/2024 5:48	7833385,917
6/24/2024 5:49	8307402,961
6/24/2024 5:49	8781420,01
6/24/2024 5:49	9255437,05
6/24/2024 5:49	7833385,92
6/24/2024 5:50	8307402,96
6/24/2024 5:50	8781420,01
6/24/2024 5:50	9255437,05
6/24/2024 5:50	7833385,92
6/24/2024 5:51	8307402,96
6/24/2024 5:51	8781420,01
6/24/2024 5:51	9255437,05
6/24/2024 5:51	7833385,92
6/24/2024 5:52	8307402,96
6/24/2024 5:52	8781420,01
6/24/2024 5:52	9255437,05
Total	4334773,2
Rata-Rata	1444924,4

IV. HASIL DAN PEMBAHASAN

A. Hasil Pengukuran *Throughput*

Pengujian untuk mengukur *throughput* dilakukan untuk mengetahui seberapa cepat data dikirimkan dan diterima oleh *client* yang terhubung melalui jaringan VPN. Parameter uji ini sangat penting untuk kebutuhan operasional perusahaan atau organisasi yang menerapkan infrastruktur *hybrid cloud*. Dari hasil pengujian ini, diambil sembilan nilai *throughput* dari masing-masing skenario pengujian pada penerapan VPN sebagai interkoneksi *hybrid cloud*.

Dari hasil pengujian *throughput* yang dilakukan pada skenario 1 (pengiriman *file dataset* dari *private* ke *public*) yang dapat dilihat pada Tabel 4, perbandingan rata-rata *throughput* menunjukkan bahwa nilai *throughput* WireGuard VPN lebih tinggi dibandingkan dengan Classic Cloud VPN. Namun, jika mengacu pada Tabel 3 hasil uji pengujian pada setiap pengiriman *file dataset*, terlihat bahwa pengiriman *file dataset* yang berukuran kecil dan sedang, Classic Cloud VPN memiliki nilai *throughput* yang sedikit lebih tinggi dibandingkan dengan WireGuard VPN. Pada hasil uji pada data Tabel 3 juga terlihat bahwa waktu yang diperlukan dalam melakukan pengiriman *file* dengan ukuran 500 MB dan 1GB Classic Cloud VPN cenderung lebih cepat dibandingkan dengan WireGuard VPN.

Tabel 3. Hasil uji *throughput* skenario 1

Ukuran *File*	Jenis VPN	Waktu Pengiriman (s)	Total *Throughput (kbps)*
500 MB	Classic Cloud VPN	180	192,66
	WireGuard VPN	300	955,38
1 GB	Classic Cloud VPN	240	295,87
	WireGuard VPN	360	1629,2
2 GB	Classic Cloud VPN	240	611,74
	WireGuard VPN	240	4557,02

Setelah dilakukan percobaan dengan *file dataset* berukuran besar, nilai *throughput* WireGuard VPN mengalami peningkatan signifikan, mencapai dua kali lipat dibandingkan dengan Classic Cloud VPN. Oleh karena itu, saat dilakukan perhitungan total dan rata-rata *throughput* pada skenario pertama ini, nilai *throughput* WireGuard VPN lebih tinggi dibandingkan dengan Classic Cloud VPN.

Pengujian perbandingan uji *throughput* pada skenario ke-2 (pengiriman *file dataset* dari *public* ke *private*), didapatkan sebuah hasil perhitungan total dari jumlah total *throughput* serta rata-rata *throughput* secara keseluruhan nilai *throughput* yang didapatkan dari kedua VPN memiliki nilai *throughput* yang lebih kecil dibandingkan pada rata-rata *throughput* pada pengujian di skenario 1. Pada perbandinganya nilai rata-rata *throughput* yang didapatkan WireGuard VPN memiliki nilai yang jauh lebih tinggi dibandingkan dengan nilai *throughput* Classic Cloud VPN. Untuk detail pengujian *throughput* mengacu pada Tabel 4 dari pengiriman *file* kecil hingga besar WireGuard VPN memiliki kinerja yang sangat jauh dibandingkan dengan Classic Cloud VPN, karena

dibandingkan pada hasil uji pada skenario 1, nilai *throughput* WireGuard VPN selalu unggul dari Classic Cloud VPN dengan selisih nilai yang jauh pada setiap pengiriman *file* dengan ukuran yang berbeda.

Tabel 4. Hasil uji *throughput* skenario 2

Ukuran *File*	Jenis VPN	Waktu Pengiriman (s)	Total *Throughput (kbps)*
500 MB	Classic Cloud VPN	240	0,35
	WireGuard VPN	240	3,09
1 GB	Classic Cloud VPN	300	0,51
	WireGuard VPN	300	4,6
2 GB	Classic Cloud VPN	180	1,54
	WireGuard VPN	300	10,95

Meskipun WireGuard VPN menunjukkan *throughput* yang lebih tinggi pada setiap skenario pengujian transfer *file*, hasil waktu tempuh pengiriman justru menunjukkan bahwa Classic Cloud VPN kerap menyamai atau bahkan mengungguli WireGuard. Anomali ini dipengaruhi oleh beberapa faktor teknis, salah satunya adalah tingginya beban CPU pada VM spesifikasi rendah akibat penggunaan enkripsi ChaCha20-Poly1305 berbasis perangkat lunak pada WireGuard, yang menciptakan *bottleneck* hingga 95% utilisasi CPU. Sebaliknya, Classic Cloud VPN memanfaatkan AES-256 dengan *hardware acceleration*, yang mampu mengurangi beban CPU hingga 70% dan meningkatkan efisiensi proses enkripsi-dekripsi.

Selain itu, *overhead* protokol WireGuard menyebabkan penurunan MTU efektif menjadi 1420 *byte*, memicu fragmentasi paket yang berdampak pada waktu pengiriman. WireGuard juga mengandalkan *routing* statis, yang menambah RTT (*Round-Trip Time*) hingga 40 ms, berbeda dengan Classic Cloud VPN yang memanfaatkan *dynamic* BGP *routing* dan infrastruktur GCP yang dioptimalkan untuk TCP, seperti *auto-tuning window size* dan *congestion control*. Tanpa konfigurasi manual terhadap parameter TCP (misalnya RWIN atau BBR), WireGuard cenderung kurang efisien dalam kondisi jaringan kompleks. Hal ini menunjukkan bahwa *throughput* tinggi tidak selalu merepresentasikan kecepatan transfer *file* yang optimal jika tidak diimbangi dengan efisiensi komputasi dan infrastruktur jaringan yang memadai.

B. Hasil Pengukuran *Packet Loss*

Pengujian *packet loss* dilakukan untuk menghitung jumlah paket yang hilang selama melakukan pengiriman *file dataset* antar lingkungan *cloud*. Jika terdapat nilai presentase *packet loss* yang tinggi perlu ditindak lanjuti apakah terdapat *miss*-konfigurasi pada infrastruktur interkoneksi yang dirancang. Pengujian *packet loss* yang dilakukan pada skenario 1 (pengiriman *file dataset* dari *private* ke *public*) menunjukkan bahwa kedua VPN memiliki persentase *packet loss* sebesar 0%. Hasil ini menunjukkan bahwa kinerja kedua VPN dalam mengirimkan data tidak mengalami kehilangan paket. Oleh karena itu, uji *packet loss* pada skenario 1 memberikan indikator bahwa kedua VPN memiliki kinerja interkoneksi yang sama baiknya, jika ditinjau dari nilai *packet loss* pada skenario pengujian 1.

Tabel 5. Hasil uji *packet loss* skenario 1

Ukuran *File*	Classic Cloud VPN (%)	WireGuard VPN (%)	Selisih
500 MB	0	0	0
1 GB	0	0	0
2 GB	0	0	0

Pengujian perbandingan *packet loss* pada skenario ke-2 (pengiriman *file dataset* dari *public* ke *private*) menunjukkan bahwa kedua VPN mengalami kenaikan persentase *packet loss*. Berdasarkan hasil perhitungan jumlah total dan rata-rata *packet loss* dari kedua VPN, yang ditampilkan pada tabel perhitungan *packet loss* skenario 2, terlihat bahwa Classic Cloud VPN memiliki keunggulan karena persentase *packet loss* yang lebih rendah dibandingkan dengan WireGuard VPN. Detail lengkap persentase *packet loss* pada setiap pengiriman *file* menunjukkan bahwa Classic Cloud VPN memang mengungguli WireGuard VPN dengan memberikan nilai persentase *packet loss* yang lebih kecil. Namun, pada Tabel 6 terlihat bahwa sebenarnya selisih persentase *packet loss* antara kedua VPN tidak terlalu jauh.

Tabel 6. Hasil uji *packet loss* skenario 2

Ukuran *File*	Classic Cloud VPN (%)	WireGuard VPN (%)	Selisih
500 MB	0,94	0,96	0,02
1 GB	0,95	0,97	0,02
2 GB	0,95	0,96	0.01

C. Hasil Pengukuran *Latency*

Pengujian *latency* dilakukan untuk mengukur waktu yang diperlukan bagi sebuah paket atau data untuk dikirimkan dari *host* menuju *destination*. Parameter *latency* menjadi sangat penting sebagai tolak ukur kinerja sebuah infrastruktur jaringan yang berjalan dengan lancar. Jika pengujian *latency* menunjukkan angka yang kecil pada sebuah infrastruktur *hybrid cloud*, maka dapat disimpulkan bahwa infrastruktur interkoneksi pada *hybrid cloud* tersebut sangat baik dalam mendukung operasional perusahaan.

Hasil uji *latency* pada skenario 1 (pengiriman *file dataset* dari *private* ke *public*) menunjukkan perbedaan yang sangat kontras antara WireGuard VPN dan Classic Cloud VPN.

Perhitungan jumlah total dan rata-rata *latency* menunjukkan selisih yang cukup signifikan. Tabel perhitungan menunjukkan bahwa WireGuard VPN memiliki kinerja yang sangat baik dengan total semua *latency* pengiriman *file* sebesar 0,0184 ms. Hasil ini mengindikasikan bahwa WireGuard VPN memiliki kinerja interkoneksi yang sangat baik dalam pengujian skenario 1. Secara detail, semakin besar *file* yang dikirim, semakin kecil *latency* yang dihasilkan pada infrastruktur *hybrid cloud* yang menggunakan WireGuard VPN. Hasil pada pengujian ini menegaskan bahwa WireGuard VPN mampu mempertahankan kinerja *latency* yang optimal bahkan saat mengirim *file* berukuran besar.

Tabel 7. Hasil uji *latency* skenario 1

Ukuran *File*	Classic Cloud VPN (ms)	WireGuard VPN (ms)	Reduksi *Latency* (%)
500 MB	0,38	0,01	97,4
1 GB	0,28	0,006	97,9
2 GB	0,13	0,0024	98,1

Selanjutnya, uji *latency* pada skenario 2 (pengiriman *file dataset* dari *public* ke *private*) memberikan hasil yang kurang memuaskan dari sisi Classic Cloud VPN. Ditinjau dari tabel perhitungan *latency* skenario 2, jumlah total dan rata-rata *latency* WireGuard VPN masih dapat mempertahankan kinerjanya dengan *latency* tidak melebihi 1 ms. Sebaliknya, pada Classic Cloud VPN, rata-rata *latency* meningkat menjadi 6,7 ms. Secara detail, pada setiap pengiriman *file*, WireGuard VPN mampu menjaga konsistensi kinerja interkoneksinya, mirip dengan skenario 1, di mana hasil *latency* akan semakin kecil saat ukuran *file* yang dikirimkan semakin besar. Meskipun Classic Cloud VPN juga menunjukkan kinerja yang relatif mirip dengan memberikan hasil *latency* yang semakin kecil seiring dengan bertambahnya ukuran *file* yang dikirimkan, hasil *latency* yang ditunjukkan masih jauh dibandingkan dengan kinerja WireGuard VPN.

Tabel 8. Hasil uji *latency* skenario 2

Ukuran *File*	Classic Cloud VPN (ms)	WireGuard VPN (ms)	Reduksi *Latency* (%)
500 MB	10,4	0,28	97,3
1 GB	7,55	0,22	97,08
2 GB	2,19	0,08	96,34

D. Hasil Analisis QoS

Dalam hasil pengujian pada setiap skenario untuk masing-masing parameter *Quality of Service* (QoS), ditemukan bahwa WireGuard VPN menunjukkan kinerja interkoneksi yang lebih unggul dibandingkan dengan Classic Cloud VPN. Namun, pada parameter *packet loss*, Classic Cloud VPN sedikit unggul dibandingkan dengan WireGuard VPN, dengan selisih presentase *packet loss* yang sangat kecil. Secara keseluruhan, saat melakukan pengujian menggunakan kedua *tools* VPN, kinerja yang terbilang maksimal terjadi saat dilakukan skenario pengiriman *file* dengan *private cloud* bertindak sebagai pengirim, dibandingkan dengan *public cloud* yang bertindak sebagai pengirim. Fenomena ini terjadi karena bentuk infrastruktur yang dibuat, di mana pemasangan VPN pada lingkungan *public cloud* mengharuskan data keluar dari *public cloud*, melalui internet, masuk ke *server* VPN, dan kemudian masuk ke *private cloud*, yang dapat menyebabkan *latency* dan *bottleneck* tambahan.

WireGuard VPN, menjadi salah satu *tools* VPN terbaik dalam hal kecepatan pengiriman data, menunjukkan kinerja yang lebih baik. Hal ini disebabkan oleh fakta bahwa WireGuard masih merupakan teknologi yang relatif baru, sehingga teknologi yang digunakannya mendukung kinerja WireGuard sebagai *tools* VPN yang memiliki kinerja di atas rata-rata. Di sisi lain, Classic Cloud VPN menggunakan protokol IPSec/IkeV yang telah ada sejak lama. Jadi dari hasil yang diperoleh dapat disimpulkan bahwa kinerja interkoneksi yang ditawarkan oleh WireGuard VPN lebih sesuai untuk diimplementasikan dalam kebutuhan operasional perusahaan yang menggunakan *hybrid cloud* untuk kebutuhan migrasi secara berkala. Hal ini karena WireGuard mampu menjaga stabilitas infrastruktur *hybrid*, terutama saat terjadi lonjakan *traffic*.

E. Perbandingan Harga dan Fleksibiltas Konfigurasi

Pada perbandingan QoS kedua VPN, terlihat bahwa kinerja WireGuard mengungguli Classic Cloud VPN. Selain perbandingan kinerja, penggunaan VPN juga dipengaruhi oleh harga dan fleksibilitas konfigurasi. Perbandingan pertama terkait harga sangat jelas: Classic Cloud VPN adalah alat VPN berbayar dari Google Cloud, sedangkan WireGuard VPN merupakan VPN *open source* yang bersifat gratis jadi perusahaan dan perorangan dapat memakainya tanpa mengeluarkan biaya. Detail biaya Classic Cloud VPN berdasarkan pricing calculator serta VM dari Google Cloud Platform dapat dilihat pada Tabel 9 dan Tabel 10.

Tabel 9. Daftar harga Classic Cloud VPN GCP

Jumlah *Tunnel*	Harga (*region* asia-southeast2)
1	$43,76/ bulan
2	$87,53/ bulan
3	$131,29/ bulan
4	$175,05/ bulan

Tabel 10. Perhitungan harga VM WireGuard *server*

Spesifikasi VM WireGuard *Server*	Harga ($)/bulan	Total Harga ($)/bulan
E2-small (1 shared core vCPU + 2GiB Memory)	16,44	18,39
Balanced Presistend Disk 15 GB	1,95	

Dari segi struktur pembiayaan, tabel perbandingan menunjukkan bahwa terdapat perbedaan mendasar dalam parameter penentuan biaya pada implementasi VPN. Pada layanan Classic Cloud VPN milik GCP, biaya dihitung berdasarkan jumlah *VPN tunnel* yang aktif di masing-masing region, sesuai dengan skema layanan yang telah disediakan oleh GCP. Sementara itu, penggunaan WireGuard VPN bersifat *open source* dan memerlukan *self-hosting*, sehingga komponen biayanya bergantung pada sumber daya *virtual machine (VM)* yang diperkirakan secara mandiri di *platform* GCP. Berdasarkan hasil perhitungan dan analisis parameter biaya pada kedua pendekatan tersebut, dapat disimpulkan bahwa Classic Cloud VPN kurang direkomendasikan bagi perusahaan atau organisasi yang masih dalam tahap berkembang, dengan rancangan infrastruktur yang masih sederhana dan perlu mengalokasikan dana seefektif mungkin. Alternatif seperti WireGuard lebih sesuai karena memberikan fleksibilitas biaya yang lebih tinggi, meskipun membutuhkan pengelolaan infrastruktur secara mandiri.

Pada sisi fleksibilitas, tidak ada kendala signifikan dalam melakukan konfigurasi perancangan kedua VPN. Namun, ada beberapa hal yang perlu digarisbawahi terkait kebijakan masing-masing VPN. Saat melakukan konfigurasi, penulis menemukan bahwa Classic Cloud VPN tidak menerima semua *traffic* masuk dari layanan di luar infrastruktur GCP. Dalam hal ini, *private cloud* yang terhubung dengan *public cloud* GCP tetap memiliki akses terbatas untuk menjalankan layanan yang perlu terintegrasi langsung dengan produk GCP. Analisis ini didasarkan pada pengalaman saat perancangan infrastruktur di mana IP *gateway* Classic Cloud VPN menolak untuk terhubung dengan layanan node_exporter sebagai *agent metrics* untuk kebutuhan monitoring. Kemungkinan, terdapat beberapa konfigurasi keamanan dari sisi GCP yang perlu disesuaikan lebih lanjut. Selain itu, dalam konfigurasi *tunnel*, Classic Cloud VPN hanya menerima IP publik untuk dapat terhubung pada *remote peer address*, sementara untuk koneksi private diperlukan konfigurasi tambahan pada cloud router GCP. Oleh karena itu, dalam hal fleksibilitas koneksi, WireGuard VPN lebih mudah diimplementasikan.

V. SIMPULAN

Setelah menguji konfigurasi VPN, penulis menemukan bahwa Classic Cloud VPN dari GCP jauh lebih rumit dibandingkan dengan WireGuard VPN yang *open source*. Kompleksitas ini muncul karena protokol IPSec/IkeV2 yang digunakan serta kebijakan Google Cloud yang ketat. Meskipun demikian, Classic Cloud VPN punya kelebihan dalam hal pemeliharaan karena otomatis terhubung dengan Cloud Logging GCP yang memudahkan pelacakan masalah *traffic* VPN serta sangat cocok dalam infrastruktur yang lebih kompleks.

Hasil pengujian menunjukkan bahwa WireGuard VPN lebih unggul dalam kinerja. WireGuard mencatat peningkatan *throughput* hingga 676,67% pada uji kedua dan pengurangan *latency* hingga 97,66% pada uji pertama. Ukuran *file* juga berpengaruh signifikan terhadap *throughput*, dengan peningkatan hingga 377,02% saat menggunakan *dataset* besar dibandingkan dengan *dataset* kecil. Namun, lingkungan cloud tempat pengiriman *file* juga mempengaruhi hasil ini.

Penulis merekomendasikan WireGuard VPN untuk perusahaan atau organisasi yang masih berkembang dengan infrastruktur yang masih sederhana karena kinerjanya yang bagus dan sifat *open source*-nya. Namun, untuk perusahaan atau organisasi besar dan sudah mengimplementasikan infrastruktur yang lebih kompleks, perlu diperhatikan bahwa skalabilitas WireGuard masih terbatas karena teknologinya masih baru dan fitur manajemennya belum matang. Untuk penelitian pada masa yang akan datang, penulis menyarankan penambahan parameter pengujian seperti utilitas *node*, perhitungan *jitter*, dan total *bandwidth*. Penggunaan alat pemantauan jaringan yang konsisten serta pengujian QoS dalam skenario *streaming* dan *load balancing* juga disarankan untuk pemahaman lebih mendalam tentang performa VPN.

REFERENSI

[1] N. I. Fitria D, Nurisnaini Putri, and Putri Zahrani, "LITERATURE REVIEW DETERMINASI INFRASTRUKTUR TI: TELEKOMUNIKASI, INTERNET DAN BRAINWARE," *J. Manaj. Pendidik. DAN ILMU Sos.*, vol. 3, no. 2, pp. 561–572, 2022, doi: 10.38035/jmpis.v3i2.1119.

[2] D. Nafis Alfarizi and I. Heidiani Ikasari, "Tinjauan Literatur Terhadap Pemanfaatan Cloud Computing," *JURIHUM J. Inov. dan Hum.*, vol. 01, no. 01, pp. 148–154, 2023, [Online]. Available: https://jurnalmahasiswa.com/index.php/jurihum

[3] A. Budiyanto, "Apa Itu Cloud Computing? Karakteristik dan Jenis Layanannya," Cloud Computing Indonesia. Accessed: May 04, 2024. [Online]. Available: https://www.cloudcomputing.id/pengetahuan-dasar/apa-itu-cloud-computing

[4] S. Mackey, I. Mihov, A. Nosenko, F. Vega, and Y. Cheng, "A Performance Comparison of WireGuard and OpenVPN," *CODASPY 2020 - Proc. 10th ACM Conf. Data Appl. Secur. Priv.*, no. July, pp. 162–164, 2020, doi: 10.1145/3374664.3379532.

[5] M. Pudelko, P. Emmerich, S. Gallenmüller, and G. Carle, "Performance Analysis of VPN Gateways," *IFIP Netw. 2020 Conf. Work. Netw. 2020*, pp. 325–333, 2020.

[6] H. Redzovic, A. Smiljanic, and B. Savic, "Performance evaluation of Software Routers with VPN features," *24th Telecommun. Forum, TELFOR 2016*, pp. 1–4, 2017, doi: 10.1109/TELFOR.2016.7818727.

[7] J. Brassil and I. Kopaliani, "CloudJoin: Experimenting at scale with Hybrid Cloud Computing," *2020 IEEE 3rd 5G World Forum, 5GWF 2020 - Conf. Proc.*, pp. 467–472, 2020, doi: 10.1109/5GWF49715.2020.9221055.

[8] H. Afifi Al-Atsari and I. Suharjo, "Integrasi Server On-Premise dengan Server Cloud Menggunakan Cloud VPN dan Mikrotik Ipsec Untuk Peningkatan Keamanan Koneksi," *J. Syntax Admiration*, vol. 4, no. 11, pp. 1977–1996, 2023, doi: 10.46799/jsa.v4i11.757.

[9] T. A. Cinderatama, Y. Yunhasnawa, and R. Z. Alhamri, "Desain Dan Implementasi Hybrid Cloud Computing Sebagai Infrastruktur Untuk Analisis Big Data Menggunakan Analytic Hierarchy Process(AHP)," *Techno.Com*, vol. 17, no. 4, pp. 404–414, 2018, doi: 10.33633/tc.v17i4.1871.

[10] P. N. P. Hai, H. N. Hong, B. B. Quoc, and T. Hoang, "A

Comparative Research on VPN Technologies on Operating System for Routers," *Int. Conf. Adv. Technol. Commun.*, vol. 2021-Octob, pp. 89–93, 2021, doi: 10.1109/ATC52653.2021.9598334.

[11] P. Thiruvasagam and K. Jijo George, "IPSec: Performance analysis in IPv4 and IPv6," *J. ICT Stand.*, vol. 7, no. 1, pp. 59–76, 2019, doi: 10.13052/jicts2245-800X.714.

[12] J. L. Shah and J. Parvez, "Impact of IPSec on Real Time applications in IPv6 and 6to4 Tunneled Migration Network," *ICIIECS 2015 - 2015 IEEE Int. Conf. Innov. Information, Embed. Commun. Syst.*, pp. 1–6, 2015, doi: 10.1109/ICIIECS.2015.7193114.

[13] K. Ghanem, S. Ugwuanyi, J. Hansawangkit, R. McPherson, R. Khan, and J. Irvine, "Security vs Bandwidth: Performance Analysis between IPsec and OpenVPN in Smart Grid," *2022 Int. Symp. Networks, Comput. Commun. ISNCC 2022*, pp. 1–5, 2022, doi: 10.1109/ISNCC55209.2022.9851717.

[14] J. Brassil and I. Kopaliani, "CloudJoin: Experimenting at scale with Hybrid Cloud Computing," *2020 IEEE 3rd 5G World Forum, 5GWF 2020 - Conf. Proc.*, pp. 467–472, 2020, doi: 10.1109/5GWF49715.2020.9221055.

[15] S. Shekhar and I. Researcher, "Comparative Analysis Of Optimizing Hybrid Cloud Environments Using AWS , Azure , And GCP," vol. 10, no. 8, pp. 791–806, 2022.

Monitoring Keamanan *Runtime* pada Kubernetes Menggunakan Falco

Ryan Fadhillah[1], Nur Rohman Rosyid[1,*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
ryanfadhillah@mail.ugm.ac.id
[*]Korespondensi: nrohmanr@ugm.ac.id;

Abstract – The use of Kubernetes as a container management platform has significantly increased in recent years. With the growth of Kubernetes usage, security has become a crucial issue that must be taken seriously. Threat detection during the runtime phase is the most important security capability that Kubernetes should possess. This is because runtime security serves as the last line of defense in a security system. To address this issue, this research focuses on developing a runtime security monitoring tool using Falco. Additionally, this project develops methods for tuning rules to enhance the detection capabilities of Falco's default rules. The detection capability testing process is conducted against three attack scenarios included in the OWASP Top 10 Cloud-Native Application Security risks, including Remote Code Execution (RCE), exfiltration using common Linux binaries, and privileged container. By tuning several rules according to the testing scenarios, this project successfully improves the detection effectiveness of malicious activities in a Kubernetes environment. This significantly contributes to organizations that aim to implement Falco for securing the runtime phase in their Kubernetes infrastructure.

Keywords – Kubernetes, Runtime Security, Falco, Rule Tuning, OWASP Top 10 Cloud-Native Application Security

Intisari – Penggunaan Kubernetes sebagai platform manajemen kontainer telah semakin meluas dalam beberapa tahun terakhir. Seiring dengan pertumbuhan penggunaan Kubernetes, keamanan menjadi isu krusial yang perlu diperhatikan secara serius. Deteksi ancaman pada fase *runtime* merupakan kapabilitas keamanan Kubernetes paling penting untuk dimiliki. Hal ini dikarenakan keamanan pada fase *runtime* merupakan garis pertahanan terakhir dalam sebuah sistem keamanan. Untuk mengatasi masalah tersebut, penelitian ini berfokus pada pengembangan alat monitoring keamanan *runtime* menggunakan Falco. Selain itu, penelitian ini juga mengembangkan metode untuk melakukan *tuning rules* untuk meningkatkan kapabilitas deteksi dari *rule default* Falco. Proses pengujian kapabilitas deteksi dilakukan terhadap tiga skenario serangan yang termasuk kedalam daftar resiko yang ada pada OWASP Top 10 *Cloud-Native Application Security*, di antaranya yaitu *Remote Code Execution* (RCE), *exfiltration using common Linux binaries,* dan *privileged container*. Melalui proses *tuning* terhadap beberapa *rule* sesuai dengan skenario pengujian, penelitian ini berhasil meningkatkan efektivitas deteksi terhadap *malicious activity* pada lingkungan Kubernetes. Hal ini memberikan kontribusi signifikan bagi organisasi yang ingin mengimplementasikan Falco dalam mengamankan fase *runtime* pada infrastruktur Kubernetes.

Kata kunci – Kubernetes, *Runtime Security*, Falco, *Tuning rule*, OWASP Top 10 *Cloud-Native Application Security*

I. PENDAHULUAN

Penggunaan Kubernetes sebagai platform manajemen kontainer telah semakin meluas dalam beberapa tahun terakhir. Kubernetes adalah salah satu bukti nyata masifnya perkembangan teknologi komputasi awan, virtualisasi, dan otomasi jaringan. Berdasarkan data dari Cloud Native Computing Foundation, Kubernetes mendapatkan predikat sebagai salah satu proyek teratas di Github yang diukur dengan jumlah bintang yang diberikan oleh pengguna Github, dengan dokumentasi yang sangat baik dan memiliki komunitas yang besar [1]. Hal ini dikarenakan Kubernetes memberikan fleksibilitas dan skalabilitas yang tinggi dalam pengelolaan aplikasi berbasis kontainer.

Kubernetes berasal dari pengalaman Google dengan mengelola sistem terdistribusi yang luas melalui sistem manajemen klaster internal yang dikenal sebagai Borg, yang menjalankan ribuan aplikasi di ribuan komputer. Kubernetes mengadopsi prinsip Borg dan mewarisi keunggulan skalabilitas, ketersediaan tinggi, dan efisiensi, yang menjadikannya platform *open-source* yang dapat diandalkan untuk pengelolaan aplikasi modern [1].

Namun, seiring dengan pertumbuhan penggunaan Kubernetes, keamanan menjadi isu krusial yang perlu diperhatikan secara serius. Sebuah studi terbaru, mengungkapkan bahwa *human-error* merupakan penyebab utama dalam 95% *data breach* yang terjadi di dunia [2]. Dalam lingkungan yang semakin kompleks dan terdistribusi, menjaga keamanan Kubernetes menjadi semakin menantang karena adanya potensi serangan dan ancaman yang dapat muncul dari berbagai sumber. Ditambah lagi, konfigurasi pada Kubernetes memang tidak didesain *secure* secara *default* [1]. Penelitian [3] mengatakan bahwa miskonfigurasi merupakan penyebab utama dari *security incident* yang terjadi di Kubernetes dengan persentase 53% responden mengalaminya.

Ketika terdapat miskonfigurasi yang menyebabkan adanya *security incident*, *indicator of attack* akan selalu muncul untuk menandakan terjadinya suatu *security incident*. Semua *indicator of attack* akan tercermin pada waktu *runtime* dikarenakan *runtime security* berperan sebagai garis pertahanan terakhir dalam sistem keamanan [4]. Berdasarkan NIST.SP-800-190 *framework*, penggunaan alat pertahanan berbasis *runtime* yang sadar akan kontainer menjadi sangat penting untuk dapat mencegah, mendeteksi, dan merespons

ancaman yang ada pada kontainer selama *runtime*. Maka dari itu, pemantauan keamanan *runtime* menjadi sangat penting dalam mendeteksi serangan secara *real-time* dan merespons ancaman dengan cepat.

Berdasarkan penelitian yang dilakukan oleh [3] dinyatakan bahwa Falco merupakan satu-satunya alat *open-source* yang paling banyak digunakan untuk melakukan monitoring terhadap kemanan *runtime* pada Kubernetes. Referensi [5] juga menyatakan bahwa Falco dapat melakukan monitoring keamanan *runtime* berbasis aturan untuk mendeteksi berbagai potensi *malicious activity* dari beban kerja di lingkungan terkontainerisasi.

Untuk mengatasi berbagai permasalahan di atas, penelitian ini menitikberatkan pada pengembangan sistem deteksi keamanan *runtime* pada Kubernetes untuk mendeteksi *malicious activity* dengan memanfaatkan perangkat lunak *open source* bernama Falco. Falco secara *default* menggunakan peristiwa *syscall* untuk mengidentifikasi pola aktivitas berbahaya. Namun, untuk meningkatkan efektivitas deteksi, administrator perlu mengatur aturan deteksi dalam format YAML yang sesuai dengan kasus penggunaan tertentu. Meskipun Falco dilengkapi dengan aturan bawaan, beberapa di antaranya masih memerlukan penyesuaian tambahan untuk keperluan spesifik. Maka dari itu, penilitian ini juga akan mengusulkan metode untuk melakukan *tuning rules* atau melakukan peningkatan kapabilitas *rules* terhadap tiga skenario serangan yang akan diujikan. Pemilihan skenario serangan didasarkan pada serangan-serangan yang termasuk kedalam daftar resiko yang ada pada OWASP *Top 10 Cloud-Native Application Security*. Skenario serangan yang akan diuji dalam penelitian ini meliputi *Remote Code Execution* (RCE), *exfiltration using common Linux binaries* dan *deployment of privileged container*.

II. DASAR TEORI

A. Kubernetes dan *Containerization*

Kubernetes merupakan alat orkestrasi kontainer yang digunakan untuk melakukan manajemen aplikasi yang dikontainerisasi. Kontainerisasi merujuk pada proses pengemasan aplikasi beserta semua komponen yang diperlukan seperti file konfigurasi terkait, *library*, dan dependensi yang diperlukan untuk memastikan aplikasi dapat beroperasi secara efisien [6].

Perusahaan memiliki beragam opsi platform yang tersedia untuk menggunakan Kubernetes. Jika anggaran perusahaan terbatas, pilihan yang tepat adalah menggunakan platform Kubernetes *self-manage* yang *open source* seperti Minikube, K3s, Kind, MicroK8s, dan lain sebagainya. Namun, hal ini mungkin menimbulkan tantangan dalam pengelolaan dan pemeliharaan sehingga mengharuskan perusahaan untuk memperhatikan berbagai aspek teknis. Sebaliknya, jika perusahaan memiliki anggaran lebih besar, disarankan untuk mempertimbangkan penggunaan *managed* Kubernetes (Kubernetes *as a service*) seperti Google GKE (Google Kubernetes Engine), Amazon EKS (Elastic Kubernetes Service), Microsoft AKS (Azure Kubernetes Service), Openshift Kubernetes Engine, Digital Ocean Kubernetes (DOKS), dan lain sebagainya. Dengan menggunakan layanan ini, perusahaan tidak perlu repot dalam mengurus atau mengelola Kubernetes secara mandiri, sehingga dapat fokus pada pengembangan aplikasi dan peningkatan produktivitas [7].

B. *Container Runtime Security*

Keamanan *runtime* kontainer bukan hanya aspek tambahan, namun merupakan elemen fundamental dan langkah proaktif untuk menjaga keamanan dan keandalan aplikasi yang terkontainerisasi selama fase *runtime*-nya. Lingkungan terkontainerisasi ini dirancang untuk ringan, portabel, dan dapat diskalakan. Namun, mereka juga membuka potensi kerentanan keamanan baru selama *runtime*. Oleh karena itu, diperlukan langkah-langkah khusus untuk memastikan keamanannya selama fase *runtime* [8], [9].

Kontainer memiliki siklus hidup yang jelas, di antaranya yaitu *build, ship*, dan *run*. Meskipun telah diberikan penekanan yang signifikan pada keamanan selama tahap *build* (misalnya, *container image scanning*) dan tahap *ship* (misalnya, orkestrasi kontainer yang aman), tahap *runtime* sering kali kurang diperiksa. Padahal, tahap ini sangat penting karena pada saat itu aplikasi berjalan dan memproses data secara langsung, menjadikannya target yang menarik bagi penyerang potensial.

Menurut penelitian [3], *runtime threat detection/response* dianggap sebagai kapabilitas yang sangat penting untuk diterapkan dalam lingkungan Kubernetes. Sebanyak 69% responden mengidentifikasikannya sebagai kapabilitas keamanan yang harus dimiliki, diikuti dengan *configuration management* dan *image scanning/vulnerability management* masing-masing dengan persentase 68% dan 65%.

C. *System Call*

Dalam komputasi, *system call* atau yang sering disingkat *syscall* adalah cara terprogram di mana suatu program komputer meminta layanan dari kernel sistem operasi di mana program tersebut dijalankan. Sebuah program komputer akan melakukan *system call* ketika melakukan permintaan kepada kernel sistem operasi. *System call* adalah satu-satunya titik masuk ke sistem kernel [10]. Oleh karena itu, dengan memantau *syscall* pada Linux Kernel yang dibuat oleh proses terkontainerisasi, semua *malicious activity* dapat dideteksi ataupun diblok.

Cara kerja *syscall* erat hubungannya dengan *user mode* dan *kernel mode*. *User mode* merupakan mode terbatas yang membatasi akses ke sumber daya sistem, sementara *kernel mode* merupakan mode yang memiliki *privileged* yang memungkinkan akses ke sumber daya system. Aplikasi yang berada di *user mode* harus membuat *system call* untuk dapat mengakses sumber daya yang ada di *kernel mode* atau melakukan operasi yang memiliki *privileged* tinggi [10].

D. eBPF

eBPF (*extended Berkeley Packet Filter*) merupakan teknologi canggih yang berasal dari kernel Linux, memungkinkan eksekusi program terisolasi dalam konteks kernel tanpa perlu memodifikasi kode sumber atau memuat modul tambahan. Teknologi ini merevolusi cara kerja sistem operasi dalam memperluas fungsi observabilitas, keamanan, dan jaringan secara dinamis dan efisien. Dengan eBPF, pengembang dapat menambahkan fitur ke sistem operasi saat runtime, yang sebelumnya sulit dilakukan karena sifat kernel yang stabil dan sensitif terhadap perubahan.

Saat ini, eBPF telah digunakan dalam beragam aplikasi seperti *load balancing* dan *networking* berkinerja tinggi di lingkungan *cloud-native*, penguatan keamanan aplikasi, dan kontainer, serta observabilitas sistem dengan *overhead* rendah. Salah satu implementasi nyatanya adalah dalam proyek *open-source* Falco, sebuah *tool* untuk deteksi ancaman *runtime*. Melalui integrasi eBPF, Falco dapat menganalisis *system call* secara *real-time* dengan lebih efisien dan aman, terutama dalam lingkungan modern seperti Kubernetes, di mana pendekatan tradisional seperti *kernel probe* memiliki keterbatasan. Kombinasi ini memperluas kapabilitas deteksi dan respons keamanan yang adaptif di ekosistem *cloud* masa kini [11].

E. Falco

Falco adalah proyek keamanan *runtime* pada *cloud native* pertama yang bergabung dengan CNCF (Cloud Native Computing Foundation) sebagai proyek dengan *maturity graduated. Maturity graduated* merupakan *maturity* tertinggi yang diberikan kepada proyek yang masuk kedalam CNCF. Jika suatu proyek telah berada di tingkat ini, itu menunjukkan bahwa proyek tersebut sudah siap untuk diimplementasikan di lingkungan *production*, memiliki komunitas yang aktif serta komitmen terhadap keberlanjutan proyek tersebut [12]. Falco dirancang untuk mendeteksi keamanan *runtime* dan memberikan *alert* jika ada perilaku abnormal dan potensi ancaman keamanan secara *real-time*. Falco bertindak sebagai kamera keamanan yang terus mendeteksi perilaku tak terduga, perubahan konfigurasi, intrusi, dan pencurian data secara *real-time* [13].

F. *Mean Time to Detect* (MTTD)

Mean Time to Detect (MTTD) adalah ukuran yang mengacu pada waktu yang diperlukan dari saat masalah pertama kali muncul hingga saat ditemukan oleh orang atau sistem monitoring. Dalam dunia *cybersecurity*, MTTD dapat diartikan sebagai berapa rata-rata waktu yang dibutuhkan sejak sebuah *event* serangan siber terjadi hingga *event* tersebut di deteksi oleh sistem *monitoring*. KPI ini melacak seberapa efektif departemen IT dan organisasi dalam menghindari gangguan jangka panjang yang dapat timbul. Semakin cepat *event* di deteksi, semakin cepat pula tim keamanan internal dapat melakukan *incident handling* untuk meminimalisir dampak resiko yang bisa muncul akibat adanya serangan tersebut [14].

Rumus *Mean Time to Detect* (MTTD) dapat dilihat pada (1) [14]. Untuk menghitung MTTD, dapat menjumlahkan semua waktu deteksi tiap insiden, lalu membaginya dengan jumlah insiden. Misalnya, jika total waktu deteksi untuk bulan Januari adalah 850 menit dan terdapat 12 insiden yang dilaporkan, maka nilai *Mean Time to Detect* akan menjadi 850 menit dibagi 12 menjadi 70.83 menit. Dengan memantau MTTD secara teratur, organisasi dapat mengidentifikasi pola waktu deteksi dan mengevaluasi efektivitas sistem deteksi masalah untuk meningkatkan kinerja dan meminimalkan dampak negatif pada bisnis.

$$MTTD = \frac{\sum_{i=1}^{n}\left(t_{deteksi_i} - t_{insiden_1}\right)}{n} \quad (1)$$

dengan:

$t_{insiden_i}$ = waktu terjadinya insiden ke-i

$t_{deteksi_i}$ = waktu insiden ke-i terdeteksi

n = Jumlah total insiden yang dilaporkan

III. METODOLOGI

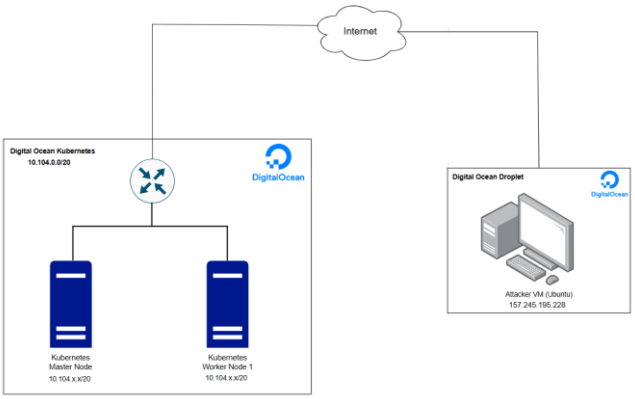
A. Lingkungan Eksperimen

Penelitian ini menggunakan *Cloud Service Provider (CSP)* dari Digital Ocean. Digital Ocean merupakan salah satu *cloud provider* yang memiliki banyak layanan. Penelitian ini menggunakan dua layanan dari Digital Ocean di antaranya yaitu Digital Ocean Droplets dan Digital Ocean Kubernetes. Tabel 1 akan menunjukkan spesifikasi tiap layanan yang digunakan dalam penelitian ini.

Tabel 1. Spesifikasi aset

Aset	Role	Spesifikasi
Digital Ocean Kubernetes	Klaster Kubernetes	8 vCPU, 8 GB RAM, 160 GB SSD
Digital Ocean Droplets	VM Attacker	1 vCPU, 2 GB RAM, 50 GB SSD

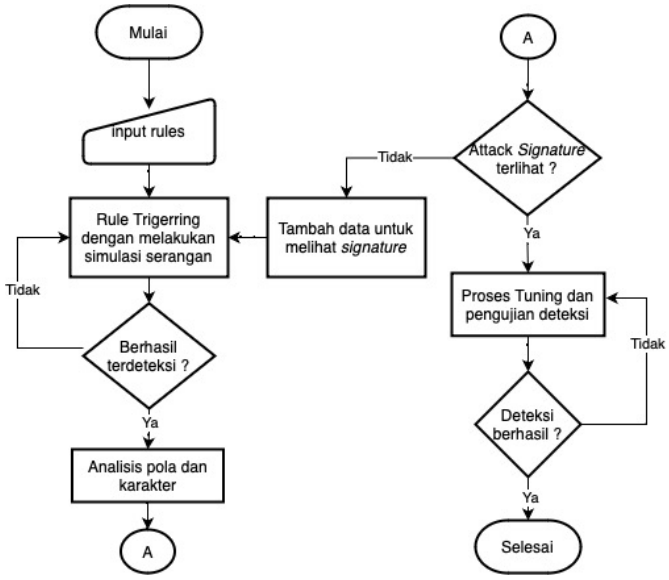
Topologi jaringan dari sistem yang digunakan pada penelitian ini dapat dilihat pada Gambar 1. Rancangan sistem yang digunakan pada penelitian ini menggunakan layanan Digital Ocean Kubernetes (DOKS) dengan satu *master node* dan satu *worker nodes*. Seluruh sistem Kubernetes memiliki alamat IP dengan subnet 10.104.0.0/20. Selain itu, penelitian ini juga menggunakan layanan Digital Ocean Droplets sebagai VM Penyerang yang akan melakukan skenario serangan dengan Alamat IP 157.245.195.228.



Gambar 1. Topologi jaringan sistem

B. Penyetelan *Rule*

Secara *default,* Falco telah memiliki beberapa *rule* default yang dapat langsung digunakan. Akan tetapi, beberapa di antaranya masih memerlukan proses *tuning* lebih lanjut untuk skenario yang lebih spesifik. Gambar 2 memperlihatkan proses *tuning* yang dilakukan.



Gambar 2. Proses *tuning rule*

Proses *tuning* Falco diawali dengan menyimulasikan berbagai skenario serangan menggunakan aturan *default* dari repositorinya. Langkah ini bertujuan mengevaluasi efektivitas deteksi awal serta mengidentifikasi aspek yang perlu disesuaikan untuk meningkatkan akurasi deteksi. Hasil awal dari langkah ini memberikan wawasan awal mengenai performa *rule* dalam mengenali aktivitas berbahaya dan menjadi dasar proses penyetelan selanjutnya.

Langkah selanjutnya adalah memeriksa pola atau *signature* khas yang terkait dengan setiap skenario yang di-*trigger* pada langkah pertama. Analisis digunakan untuk memahami detail khusus dari *malicious activity* yang terdeteksi, memberikan wawasan tentang ciri-ciri uniknya.

Jika tidak melihat karakteristik ataupun pola yang jelas dari output yang diberikan, maka perlu menambahkan data tambahan untuk membuat *output* lebih informatif. Penambahan ini bertujuan untuk memberikan pemahaman yang lebih jelas tentang perbedaan karakteristik di setiap pengujian skenario. Untuk daftar lengkap *event* yang tersedia pada dokumentasi Falco [15].

Dengan mempertimbangkan wawasan yang diperoleh dari analisis pola, langkah selanjutnya yaitu melakukan *tuning rule* Falco terhadap skenario tertentu. Proses *tuning rule* dilakukan dengan mempertimbangkan skenario spesifik dari sebuah ancaman. Dalam hal ini, perlu pemahaman mendalam dari sebuah skenario serangan agar memudahkan dalam proses *tuning rule*. Proses *tuning* ini bertujuan untuk meningkatkan kemampuan Falco dalam mendeteksi pola-pola unik yang terkait dengan berbagai ancaman.

Setelah melakukan proses *tuning rule*, aturan tersebut dapat disimpan pada direktori "/etc/falco". Secara *default,* terdapat dua file dalam direktori tersebut, di antaranya yaitu file "falco.yaml" untuk konfigurasi Falco dan "falco_rules.yaml" untuk aturan *default*. Jika ingin menambahkan aturan baru di luar aturan *default*, pengguna dapat membuat file baru dengan nama "falco_rules.local.yaml". Setelah itu, dapat mengulangi proses *tuning* seperti pada Gambar 2.

Pada penelitian ini, proses *tuning rule* berfokus pada tiga skenario yang diujikan, di antaranya yaitu *Remote Code Execution, exfiltration using common Linux binaries*, dan *deployment privileged container*. Berikut penjelasan dari tiap-tiap skenario.

1. Skenario *Remote Code Execution*

Untuk skenario serangan *Remote Code Execution* (RCE), penelitian ini menggunakan aturan yang berasal dari Repositori Falco di Github: "falco_rules.yaml". Secara *default*, aturan ini sudah diaktifkan ketika menggunakan Falco untuk pertama kali. Namun, selama pengujian, beberapa skenario serangan tidak terdeteksi oleh aturan ini, sehingga perlu dilakukan proses *tuning* lebih lanjut. Gambaran secara umum cara kerja *rule* RCE pada Falco dapat dilihat diagram YAML pada Gambar 3.

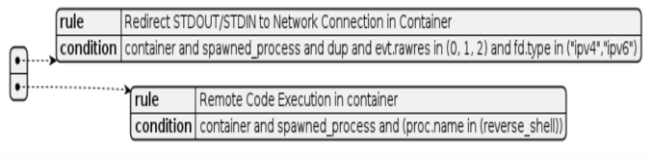
rule	Redirect STDOUT/STDIN to Network Connection in Container
condition	dup and container and evt.rawres in (0, 1, 2) and fd.type in ("ipv4", "ipv6")

Gambar 3. Diagram YAML *rule* sebelum *tuning* dengan RCE

Setelah dilakukan berbagai percobaan serangan dan melihat pola dari tiap-tiap serangan, langkah selanjutnya yaitu melakukan modifikasi atau penambahan kondisi pada aturan tersebut untuk meningkatkan kapabilitas aturan Falco sesuai dengan meningkatnya variasi serangan untuk skenario

Remote Code Execution (RCE). Proses *tuning* ini secara khusus terkait dengan penggunaan biner Linux umum yang sering digunakan dalam aktivitas *reverse shell*. Penelitian ini melakukan penyempurnaan aturan dengan menambahkan kondisi (*spawned_process and* (*proc.name in* (*reverse_shell*)) untuk mendeteksi penggunaan biner Linux umum, seperti *nc*, *ncat*, *socat*, *telnet*, *zsh*, dan *rushcat*, sebagai indikator potensial dari upaya *Remote Code Execution* (RCE). Penambahan kondisi ini dapat mengurangi adanya *false negative* dalam aturan ini. Selain itu, penelitian ini juga menambahkan kondisi pengecualian terhadap *proc.name in* ("kubelet", "dockerd") agar aturan ini tidak akan menandainya sebagai peringatan dikarenakan process tersebut merupakan process yang dilakukan oleh komponen internal di dalam kubernetes dan bukan merupakan proses yang berbahaya. Penambahan kondisi pengecualian ini ditujukan untuk mengurangi adanya *false positive* dalam aturan ini.

Gambaran secara umum terhadap cara kerja *rule* RCE pada Falco dapat dilihat diagram YAML pada Gambar 4.

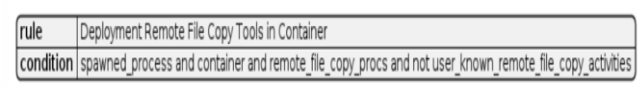


Gambar 4. Diagram YAML *rule* setelah *tuning* dengan RCE

2. Skenario *Exfiltration using Common Linux Binaries*

Untuk skenario serangan *exfiltration using common Linux binaries*, penelitian ini menggunakan aturan yang berasal dari repositori Falco di Github: "falco-incubating_rules.yaml", menandakan bahwa aturan tersebut diidentifikasi oleh para ahli sebagai aturan yang akan menangani skenario penggunaan terhadap serangan yang lebih spesifik, yang mungkin relevan untuk beberapa pengguna, tetapi tidak relevan terhadap pengguna lainnya. Secara *default*, aturan ini sudah diaktifkan ketika menggunakan Falco untuk pertama kali. Namun, aturan *default* hanya mendefinisikan empat biner Linux umum terkait eksfiltrasi, sehingga memerlukan modifikasi tambahan untuk meningkatkan kemampuan deteksi untuk skenario eksfiltrasi. Biner Linux yang masuk kedalam aturan *default* di antaranya *rsync*, *scp*, *sftp*, dan *dcp*.

Namun, selama pengujian, beberapa skenario serangan tidak terdeteksi oleh aturan ini, sehingga perlu dilakukan proses *tuning* lebih lanjut. Cara kerja *rule exfiltration using common Linux binaries* pada Falco sebelum proses *tuning* dapat dilihat diagram YAML pada Gambar 5.



Gambar 5. Diagram YAML *rule* sebelum *tuning* dengan *Exfiltration Using Common Linux Binaries*

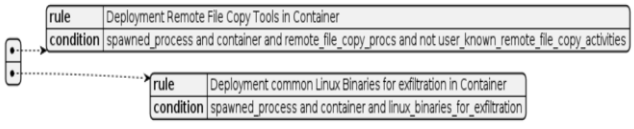
Setelah dilakukan berbagai percobaan serangan dan melihat pola dari tiap-tiap serangan, penelitian ini melakukan modifikasi atau penambahan kondisi pada aturan tersebut untuk meningkatkan kapabilitas aturan Falco sesuai dengan meningkatnya variasi serangan untuk skenario *exfiltration using common Linux binaries*. Proses *tuning* ini secara khusus terkait dengan penggunaan biner Linux umum yang sering digunakan dalam lingkungan Linux yang dapat disalahgunakan untuk melakukan eksfiltrasi, seperti *wget*, *whois*, *bash*, *openssl*, *curl*, dan *bash*. Penambahan kondisi pada aturan dilakukan untuk meningkatkan kapabilitas deteksi dari skenario *exfiltration using common Linux binaries* dapat dilihat pada Tabel 2.

Tabel 2. Penambahan kondisi pada *rule exfiltration using common Linux binaries*

Binari Linux	**Penambahan Kondisi**
wget	(proc.name = "wget" and (proc.cmdline contains "--post-file" or proc.cmdline contains "--post-data"))
whois	(proc.name = "whois" and (proc.cmdline contains "-h" or proc.cmdline contains "-p"))
bash	(proc.name = "bash" and proc.cmdline contains "-c")
curl	(proc.name = "curl" and (proc.cmdline contains "-x POST" or proc.cmdline contains "-d"))
KSH	(proc.name = "ksh" and proc.cmdline contains "-c")
openssl	(proc.name = "openssl" and (proc.cmdline contains "-connect" or proc.cmdline contains "s_client"))

Penambahan kondisi pada Tabel 2 dilakukan dengan mendeteksi *event* "proc.name" untuk mendeteksi binari Linux yang berjalan serta "proc.cmdline" untuk menentukan parameter apa saja pada binari linux tersebut yang dapat disalahgunakan untuk melakukan eksfiltrasi. Hal ini dilakukan karena beberapa dari binari Linux tersebut juga masih sering digunakan oleh *developer* internal, sehingga kondisi pada *rules* perlu disesuaikan untuk mengurangi kemungkinan terjadinya *false positive*.

Secara umum, cara kerja *rule exfiltration using common Linux binaries* pada Falco setelah proses *tuning* dapat dilihat diagram YAML pada Gambar 6.



Gambar 6. Diagram YAML *rule* setelah *tuning* dengan *exfiltration using Common Linux Binaries*

3. Skenario *Deployment of Privileged Container*

Untuk skenario *deployment of privileged container*, penelitian ini menggunakan aturan yang berasal dari repositori Falco di Github: "falco-incubating_rules.yaml" yang menandakan bahwa aturan tersebut diidentifikasi oleh para ahli sebagai aturan yang akan menangani skenario penggunaan terhadap serangan yang lebih spesifik, yang mungkin untuk beberapa pengguna namun tidak relevan terhadap pengguna lainnya. Secara *default*, aturan ini sudah diaktifkan ketika menggunakan Falco untuk pertama kali. Gambaran secara umum cara kerja *rule deployment of privileged container* pada Falco sebelum proses *tuning* dapat dilihat diagram YAML pada Gambar 7.

rule	Deployment of Privileged container
condition	container_started and container and container.privileged=true

Gambar 7. Diagram YAML *rule* sebelum *tuning* dengan *Deployment of Privileged Container*

Setelah dilakukan berbagai percobaan serangan dan melihat pola dari tiap-tiap serangan, penelitian ini melakukan modifikasi atau penambahan kondisi pada aturan tersebut untuk meningkatkan kapabilitas aturan Falco sesuai dengan meningkatnya variasi serangan untuk skenario *deployment of privileged container*. Proses *tuning* ini dilakukan untuk mengurangi adanya *false positive* dari aturan ini terhadap pembuatan *container* dengan *privilege* tinggi yang dilakukan oleh tim internal dalam melakukan pengembangan. Maka dari itu, perlu penambahan kondisi pada *macro* "user_privileged_containers", terkait dengan daftar "proc.name" yang diizinkan untuk dijalankan dengan mode *privilege* tinggi milik *developer*. Gambaran secara umum cara kerja *rule deployment of privileged container* pada Falco setelah proses *tuning* dapat dilihat diagram YAML pada Gambar 8.

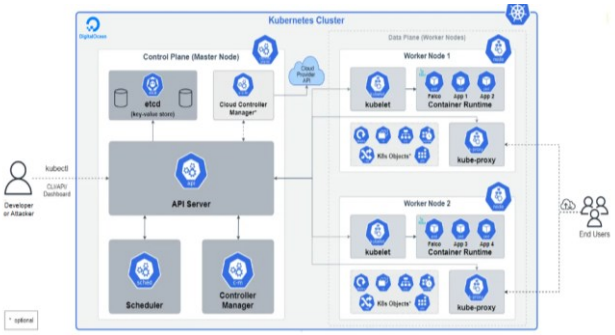
rule	Deployment of Privileged container
condition	container_started and container and container.privileged=true

rule	Exception for Deployment of Developer Privileged container
condition	spawned_process and container and not user_privileged_containers

Gambar 8. Diagram YAML *rule* setelah *tuning* dengan *deployment of privileged container*

C. Pengujian Performa *Rule*

Topologi dari skenario serangan yang akan dilakukan pada penelitian ini dapat dilihat pada Gambar 9. Dalam hal ini, *attacker* sudah mendapat akses terhadap *API Server* dari klaster Kubernetes sehingga dapat mengakses *pod* secara langsung untuk menjalankan *malicious activity* di dalam *pod* tersebut. Di sisi lain, masing-masing *worker node* akan terinstal satu buah agen Falco yang akan melakukan monitoring terhadap *malicious activity* pada *node*-nya.



Gambar 9. Topologi serangan

Pengujian performa *rule* mencakup komparasi dari kapabilitas deteksi Falco menggunakan *rule* sebelum dan setelah *tuning*. Pengujian dilakukan terhadap tiga skenario *malicious activity* yang sebelumnya sudah di-*tuning*, di antaranya *Remote Code Execution, exfiltration using common Linux binaries*, dan *deployment of privileged container.*

1. Skenario *Remote Code Execution*

Skenario *Remote Code Execution* dirancang untuk mengevaluasi efektivitas Falco dalam mendeteksi aktivitas yang terkait dengan eksekusi *Remote Code Execution* (RCE) dalam lingkungan terkontainerisasi. Fokus utamanya adalah mengidentifikasi potensi ancaman keamanan yang timbul dari upaya eksekusi kode yang tidak sah atau berbahaya. Evaluasi ini melibatkan simulasi berbagai teknik eksekusi *Remote Code Execution* (RCE) yang mungkin digunakan oleh penyerang untuk mengompromi aplikasi terkontainerisasi. Skenario ini mempertimbangkan metode umum seperti penggunaan alat seperti Socat, SQLite3, Ncat, Telnet dan bahasa pemrograman seperti Python, Ruby, dan PHP untuk eksekusi kode. Tujuan menguji set aturan Falco secara komprehensif terhadap teknik-teknik *Remote Code Execution* (RCE) adalah untuk memastikan deteksi yang tepat waktu dan akurat, memungkinkan respons proaktif untuk mengurangi risiko keamanan.

2. Skenario *Exfiltration using Common Linux Binaries*

Skenario *exfiltration using common Linux binaries* berfokus untuk mengevaluasi kemampuan deteksi Falco terkait dengan aktivitas eksfiltrasi data menggunakan biner linux yang umum digunakan seperti wget, whois, bash, openssl, curl, KSH, rsync, SCP dan SFTP. Pada dasarnya biner Linux tersebut merupakan biner Linux yang umum digunakan dalam lingkungan Linux, akan tetapi jika disalahgunakan, biner Linux tersebut dapat digunakan untuk melakukan hal-hal yang berbahaya seperti eksfiltrasi data. Eksfiltrasi data merupakan ancaman yang signifikan. Penyerang dapat menggunakan biner Linux umum untuk mengirimkan informasi sensitif keluar dari lingkungan terkontainerisasi.

3. Skenario *Deployment of Privileged Container*

Deployment of privileged container melibatkan pengujian kemampuan deteksi Falco dalam mengidentifikasi pembuatan kontainer dengan *privilege* tinggi. Fokus utamanya adalah pada pemeriksaan dan identifikasi aktivitas berpotensi merugikan dalam kontainer yang memiliki *privilege* yang tinggi. Jika seorang penyerang berhasil membuat dan menggunakan kontainer dengan *privilege* tinggi, hal itu membuka pintu untuk tindakan berbahaya dengan konsekuensi yang besar, seperti akses tidak sah ke data penting, manipulasi pengaturan sistem penting, atau eksploitasi kerentanan.

Pada dasarnya menjalankan *privileged container* dapat mengizinkan tim internal untuk mendapat akses kritikal ke sumber daya yang ada di *host*. Namun, jika disalahgunakan, akses istimewa ini dapat menimbulkan risiko keamanan yang serius. Oleh karena itu, penting untuk memantau pembuatan *privileged container* secara cermat guna meminimalkan kemungkinan terjadinya kejadian yang tidak diinginkan dan mencegah potensi kerentanan keamanan yang dapat dieksploitasi

D. Pengujian *Mean Time To Detect (MTTD)*

Pengujian *Mean Time to Detect* (MTTD) dilakukan untuk menghitung rata-rata waktu yang dibutuhkan sejak sebuah serangan siber terjadi hingga serangan tersebut di deteksi oleh sistem *monitoring* Falco. Untuk melakukan pengujian ini dibutuhkan data berupa waktu ketika serangan dijalankan serta waktu ketika Falco mendeteksi serangan. Kemudian, akan dihitung selisihnya untuk mendapatkan nilai MTTD sejak sebuah serangan siber terjadi hingga serangan tersebut dideteksi oleh sistem monitoring Falco.

Untuk mendapatkan data waktu ketika serangan dijalankan, penelitian ini menggunakan utilitas *echo* dan *date* pada VM penyerang untuk mencatat waktu saat *payload* di jalankan. Penggunaan utilitas *time* dilakukan dengan cara menuliskan *command echo* dan *time* diikuti dengan *payload* serangan yang akan diujikan. Setelah itu, akan tertampil *output* berupa waktu ketika *payload* tersebut dijalankan. Format penggunaan dari utilitas *echo* dan *time* dapat dilihat pada Gambar 10 dengan nilai x diganti dengan *script payload* yang akan digunakan.

```
echo $(date; x;)
```

Gambar 10. Format informasi waktu ketika *payload* dijalankan pada VM penyerang

Sebagai contoh, untuk skenario pengujian *Remote Code Execution* dengan utilitas Socat, dapat dilihat pada Gambar 11.

```
root@privileged-pod:/# echo $(date;socat TCP:157.245.195.228:34223 EXEC:'sh'
,pty,stderr,setsid,sigint,sane)
Mon Jun 10 07:07:13 UTC 2024
```

Gambar 11. Contoh penggunaan format RCE

Untuk mendapatkan data waktu ketika Falco mendeteksi serangan, penelitian ini menggunakan data notifikasi yang berada pada *log* Falco. Jika Falco berhasil mendeteksi adanya *malicious activit*y, maka pada *log* Falco akan tercatat juga waktu ketika Falco mendeteksi serangan tersebut. Dapat dilihat pada Gambar 12 sebagai contoh untuk mengambil informasi waktu pada *log* Falco.

```
07:07:13.216120914: Notice Remote Code Execution (gparent=bash ggparent=cont
ainerd-shim gggparent=systemd fd.sip=<NA> connection=<NA> lport=<NA> rport=<
NA> fd_type=<NA> fd_proto=fd.l4proto evt_type=execve user=root user_uid=0 us
er_loginuid=-1 process=socat proc_exepath=/usr/bin/socat parent=socat comman
d=socat TCP:157.245.195.228:34223 EXEC:sh,pty,stderr,setsid,sigint,sane term
inal=34844 exe_flags=EXE_WRITABLE|EXE_UPPER_LAYER container_id=1c69f5fb61ce
container_image=docker.io/library/nginx container_image_tag=7383c266ef252ad7
0806f3072ee8e63d2a16d1e6bafa6146a2da867fc7c41759 container_name=privileged-p
od k8s_ns=default k8s_pod_name=privileged-pod evt_type=execve)
```

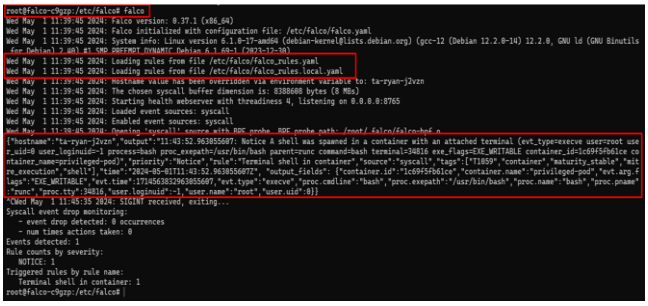
Gambar 12. Informasi waktu pada log Falco

Setelah mendapatkan kedua data tersebut, penelitian ini menghitung selisih dari kedua data tersebut untuk mendapatkan nilai MTTD. Pengujian ini dilakukan untuk melihat komparasi durasi waktu deteksi antara *rule* sebelum dan sesudah dilakukan *tuning*.

IV. HASIL DAN PEMBAHASAN

A. Pengujian Fungsionalitas

Pengujian fungsionalitas sistem dilakukan dengan tujuan untuk memastikan fungsionalitas Falco sudah berjalan dengan baik. Pengujian dilakukan dengan cara memasukkan seluruh *rule* Falco, baik sebelum dilakukan *tuning*, maupun setelah dilakukan *tuning*. Setelah itu, untuk menjalankan Falco dapat dengan menjalankan perintah “falco”. Jika terdapat kesalahan pada penulisan *rule*, maka akan muncul pesan eror yang menunjukkan letak kesalahan penulisan *rule*. Akan tetapi, jika tidak terdapat kesalahan pada penulisan *rule*, maka Falco akan berjalan dan pengguna dapat melihat *output event* yang terdeteksi secara langsung pada *shell*.



Gambar 13. Hasil Pengujian Fungsionalitas Falco

Gambar 13 menunjukkan bahwa ketika menjalankan Falco, tidak terdapat pesan eror. Jika terdapat kesalahan penulisan dalam pembuatan *rules*, Falco akan memberikan pesan eror dan memberitahu pengguna untuk memperbaiki penulisan *rule* tersebut sebelum menjalankan Falco. Selain itu, ketika mencoba melakukan sebuah *malicious activity* dengan cara menjalankan shell pada salah satu *pod*, Falco berhasil mendeteksi *malicious activity* tersebut. Hal ini

menunjukkan bahwa Falco telah berjalan dengan baik dalam mendeteksi *malicious activity* sesuai dengan *rule* yang terdapat didalamnya.

B. Pengujian Performa *Rule*

1. Skenario *Remote Code Execution*

Dari hasil pengujian terhadap aturan Falco sebelum dan sesudah dilakukan proses *tuning*, terlihat pada Tabel 3, setelah dilakukan proses *tuning rule*, terdapat kenaikan persentase kapabilitas deteksi terhadap skenario *Remote Code Execution* dari 44% ke 100%. Dalam artian, setelah dilakukan proses *tuning*, aturan tersebut dapat mendeteksi seluruh skenario pengujian yang dilakukan. Kenaikan persentase ini dicapai dikarenakan proses *tuning* melibatkan berbagai percobaan untuk memahami *signature* dari tiap serangan. Dalam hal ini, skenario *Remote Code Execution* dapat dideteksi berdasarkan "proc.name" dari penggunaan binari Linux yang sering digunakan dalam melakukan *Remote Code Execution*. Binari Linux yang dapat dideteksi dari serangan ini di antaranya *nc*, *ncat*, *socat*, *telnet*, *zsh*, dan *rushcat*. Kondisi "proc.name" ini juga disandingkan dengan operator *boolean* "dan" terhadap makro "spawned_process" yang memiliki kondisi (*evt.type in (execve, execveat) and evt.dir=<*). Makro tersebut memang telah tersedia pada aturan falco dan biasanya penggunaannya disandingkan dengan kondisi "proc.name" untuk mendeteksi eksekusi dari binari Linux yang didefinisikan pada kondisi "proc.name".

Tabel 3. Hasil pengujian performa *rule* dengan RCE

Skenario Pengujian	Hasil Pengujian	
	Sebelum *Tuning*	**Setelah *Tuning***
Socat 1	×	✓
Socat 2	×	✓
Sqlite3 NC MKFIFO	×	✓
NC MKFIFO	×	✓
PHP Shell Exec	✓	✓
Telnet	×	✓
Python	✓	✓
Ruby	✓	✓
ZSH	✓	✓
Total	**4** dari **9** Berhasil	**9** dari **9** Berhasil
Persentase	**44%**	**100%**

2. Skenario *Exfiltration using Common Linux Binaries*

Dari hasil pengujian terhadap aturan Falco sebelum dan sesudah dilakukan proses *tuning*, terlihat pada Tabel 4, setelah dilakukan proses *tuning rule*, terdapat kenaikan persentase kapabilitas deteksi terhadap skenario *exfiltration using common Linux binaries* dari 33% ke 100%. Dalam artian, sebelum dilakukan *tuning* aturan tersebut hanya dapat mendeteksi tiga skenario yang diujikan sedangkan setelah dilakukan proses *tuning*, aturan tersebut dapat mendeteksi seluruh skenario pengujian yang dilakukan. Kenaikan persentase ini dicapai dikarenakan proses *tuning* melibatkan berbagai percobaan untuk memahami *signature* dari tiap serangan. Dalam hal ini, skenario *exfiltration using common Linux binaries* dapat dideteksi berdasarkan proc.name beberapa binary yang termasuk dalam pengujian, di antaranya *wget*, *whois*, *bash*, *openssl*, *curl*, KSH, SCP, *Rsync* dan SFTP. Tiap-tiap binari juga disandingkan dengan parameter *boolean* "dan" terhadap *event* "proc.cmdline" untuk memastikan *event* yang terdeteksi merupakan *event* yang berkaitan dengan kegiatan eksfiltrasi data. Hal ini dikarenakan binari Linux umum tersebut sangat sering dijumpai penggunaannya, tetapi hanya perlu melakukan monitoring terhadap parameter yang terkait dengan proses eksfiltrasi. Sehingga, ketika terdapat penggunaan binari Linux umum yang tidak terkait dengan eksfiltrasi data, maka falco tidak akan mendeteksi nya sebagai *malicious activity*.

Tabel 4. Hasil pengujian performa *rule* dengan *Exfiltration Using Common Linux Binaries*

Skenario Pengujian	Hasil Pengujian	
	Sebelum *Tuning*	**Setelah *Tuning***
wget	×	✓
whois	×	✓
bash	×	✓
openssl	×	✓
curl	×	✓
KSH	×	✓
rsync	✓	✓
SCP	✓	✓
SFTP	✓	✓
Total	**3** dari **9** Berhasil	**3** dari **9** Berhasil
Persentase	**33%**	**100%**

3. Skenario *Deployment of Privileged Container*

Dari hasil pengujian terhadap aturan Falco sebelum dan sesudah dilakukan proses *tuning*, terlihat pada Tabel 5, setelah dilakukan proses *tuning rule*, terdapat kenaikan persentase kapabilitas deteksi terhadap skenario *deployment of privileged container* dari 66% ke 100%. Dalam artian, setelah dilakukan proses *tuning*, aturan tersebut dapat mendeteksi seluruh skenario pengujian yang dilakukan. Kenaikan persentase ini dicapai dikarenakan proses *tuning* melibatkan berbagai percobaan untuk memahami *signature* dari tiap serangan. Dalam hal ini *rule* pada skenario *deployment of privileged container* dapat mengurangi adanya *false positive* dengan mengurangi *alert* yang dihasilkan ketika proses *deployment container* dengan *privilege* tinggi yang dimiliki oleh developer. Hal ini dilakukan dengan cara menambahkan kondisi pengecualian dengan memberikan daftar "proc.name" apa saja yang diizinkan untuk dijalankan dengan *privilege* tinggi.

Tabel 5. Hasil pengujian performa *rule* dengan *Privileged Container*

Skenario Pengujian	Hasil Pengujian	
	Sebelum *Tuning*	**Setelah *Tuning***
Deployment privileged container using command line	✓	✓
Deployment privileged container using manifest file	✓	✓
Deployment developer privileged container (exception)	×	✓
Total	**2** dari **3** Berhasil	**2** dari **3** Berhasil
Persentase	**33%**	**100%**

C. Pengujian *Mean Time To Detect (MTTD)*

Pengujian selanjutnya yaitu pengujian *Mean Time to Detect* (MTTD). Pengujian ini dilakukan untuk menentukan rata-rata waktu yang dibutuhkan dalam mendeteksi sebuah serangan sejak serangan tersebut diluncurkan hingga terdeteksi oleh sistem monitoring Falco.

1. Skenario *Remote Code Execution*

Berdasarkan hasil perhitungan menggunakan rumus MTTD pada Gambar 2, bisa dilihat pada Tabel 6, bahwa tidak terdapat perbedaan signifikan antara MTTD menggunakan *rule* sebelum dan sesudah dilakukan *tuning*. Sebelum dan setelah dilakukan *tuning*, hasil pengujian menunjukkan bahwa rata-rata MTTD untuk semua skenario adalah sama-sama 0 detik. Hal ini menunjukkan respon sistem Falco yang sangat cepat dalam mendeteksi serangan RCE.

Tabel 6. Hasil pengujian MTTD dengan RCE

Skenario Pengujian	Hasil Pengujian (detik)	
	Sebelum *Tuning*	**Setelah *Tuning***
Socat 1	0	0
Socat 2	0	0
Sqlite3 NC MKFIFO	0	0
NC MKFIFO	0	0
PHP Shell Exec	0	0
Telnet	0	0
Python	0	0
Ruby	0	0
ZSH	0	0
MTTD	**0**	**0**

2. Skenario *Exfiltration using Common Linux Binaries*

Berdasarkan hasil perhitungan menggunakan rumus MTTD pada Gambar 2, bisa dilihat pada Tabel 7, hasil perhitungan *Mean Time to Detect* (MTTD) pada skenario *exfiltration using common Linux binaries* sebelum dan setelah *tuning*, didapatkan nilai MTTD dengan angka yang sama sebesar 0 detik. Hasil ini menunjukkan respons yang cepat dari sistem monitoring Falco terhadap serangan yang dilakukan menggunakan berbagai binari Linux seperti *wget*, *whois*, *bash*, *openssl*, *curl*, KSH, *Rsync*, SCP dan SFTP.

Tabel 7. Hasil pengujian MTTD dengan *Exfiltration Using Common Linux Binaries*

Skenario Pengujian	Hasil Pengujian (detik)	
	Sebelum *Tuning*	**Setelah *Tuning***
wget	0	0
whois	0	0
bash	0	0
openssl	0	0
curl	0	0
KSH	0	0
rsync	0	0
SCP	0	0
SFTP	0	0
MTTD	**0**	**0**

3. Skenario *Deployment of Privileged Container*

Berdasarkan hasil perhitungan menggunakan rumus MTTD pada Persamaan (1), hasil perhitungan MTTD pada Tabel 8 menunjukkan bahwa tidak terdapat perbedaan signifikan antara MTTD menggunakan *rule* sebelum dan sesudah dilakukan *tuning*. Sebelum dilakukan *tuning*, hasil pengujian menunjukkan bahwa rata-rata MTTD untuk semua skenario *privileged container* adalah 2,67 detik. Setelah dilakukan *tuning*, rata-rata MTTD tercatat di angka 4,33 detik untuk semua skenario. Perbandingan antara rata-rata MTTD sebelum dan setelah *tuning* menunjukkan bahwa tidak ada perbedaan yang signifikan. Hal ini menunjukkan efektivitas sistem Falco yang baik dalam mendeteksi *malicious activity*.

Jika dibandingkan dengan skenario *Remote Code Execution* (RCE) dan *exfiltration using common Linux binaries*, proses pendeteksian pada skenario *deployment of privileged container* memerlukan waktu lebih lama sekitar 2-4 detik. Hal ini disebabkan karena pada proses *deployment privileged pod*, dibutuhkan waktu tambahan yang dibutuhkan oleh Kubernetes untuk dapat menjalankan *container* tersebut dari proses *ContainerCreating* hingga proses *running*. Meskipun demikian, hasil perhitungan menunjukkan bahwa waktu yang dibutuhkan Falco untuk mendeteksi adanya *malicious activity* di lingkungan Kubernetes masih relatif singkat. Hal ini menunjukkan efektivitas yang baik dari sistem Falco dalam mendeteksi aktivitas berbahaya yang memungkinkan proses *incident handling* bisa lebih cepat untuk meminimalisir resiko yang dapat timbul.

Tabel 8. Hasil pengujian MTTD dengan *Privileged Container*

Skenario Pengujian	Hasil Pengujian (detik)	
	Sebelum *Tuning*	**Setelah *Tuning***
Deployment privileged container using command line	4	0
Deployment privileged container using manifest file	4	0
Deployment developer privileged container (exception)	0	0
Total	**2,67**	**0**

V. SIMPULAN

Berdasarkan data yang diperoleh dan analisis yang dilakukan dari penelitian ini, dapat diambil beberapa kesimpulan. Pertama, implementasi Falco dapat menjawab masalah diperlukan adanya monitoring pada fase *runtime* untuk mendeteksi *malicious activity* di Kubernetes. Selanjutnya, berdasarkan hasil pengujian performa *rule*, proses *tuning rule* yang telah dilakukan berhasil meningkatkan kapabilitas deteksi skenario *Remote Code Execution* sebesar 56%, skenario *exfiltration using common Linux binaries* sebesar 67%, dan skenario *deployment privileged container* sebesar 33%. Terakhir, berdasarkan data yang diperoleh dari pengujian skenario *Remote Code Execution* (RCE), *exfiltration using common Linux binaries*, dan *deployment privileged container*, tidak terdapat perbedaan durasi deteksi serangan sebelum dan setelah proses *tuning*. Rata-rata nilai MTTD atau waktu yang dibutuhkan untuk mendeteksi serangan tetap konsisten di bawah lima detik, menunjukkan respons cepat dari sistem *monitoring* Falco dalam menghadapi berbagai serangan pada lingkungan Kubernetes. Hal ini menunjukkan bahwa implementasi Falco dan proses *tuning rule* secara efektif meningkatkan kapabilitas deteksi tanpa mengorbankan waktu deteksi yang cepat.

REFERENSI

[1] B. Yuen, A. Matyushentsev, T. Ekenstam, and J. Suen, GitOps and Kubernetes Continuous Deployment with Argo CD, Jenkins X, and Flux. 2021.

[2] M. McLennan, SK Group, and Zurich Insurance Group, “The Global Risks Report 2022 17th Edition,” 2022.

[3] Redhat, “State of Kubernetes security report,” 2022. Accessed: May 03, 2024. [Online]. Available: https://www.redhat.com/rhdc/managed-files/cl-state-of-kubernetes-security-report-2022-ebook-f31209-202205-en.pdf

[4] S. Singh, “Cloud Computing : An Overview,” Int J Sci Res Sci Eng Technol, pp. 63–70, Jan. 2019, doi: 10.32628/IJSRSET196120.

[5] H. Gantikow, C. Reich, M. Knahl, and N. Clarke, “Rule-Based Security Monitoring of Containerized Environments,” 2020, pp. 66–86. doi: 10.1007/978-3-030-49432-2_4.

[6] Nutanix, “What is Kubernetes?” Accessed: May 13, 2024. [Online]. Available: https://www.nutanix.com/info/what-is-kubernetes#definition

[7] Abhilash, “Self-managed Kubernetes Vs. Kubernetes-as-a-service (Managed Kubernetes).” Accessed: Apr. 25, 2024. [Online]. Available: https://www.ozone.one/self-managed-kubernetes-vs-kubernetes-as-a-service-managed-kubernetes/

[8] A. Y. Wong, E. G. Chekole, and M. O. and J. Zhou, “Threat Modeling and Security Analysis of Containers: A Survey,” arXiv preprint arXiv:2111.11475, Nov. 2021. Accessed: May 25, 2025. [Online]. Available: https://arxiv.org/abs/2111.11475

[9] M. Sroor, R. Mohanani, R. Colomo-Palacios, S. Dasanayake, and T. Mikkonen, “Managing Security Issues in Software Containers: From Practitioners Perspective,” arXiv preprint arXiv:2504.07707, Apr. 2025. Accessed: May 25, 2025. [Online]. Available: https://arxiv.org/abs/2504.07707

[10] GeeksForGeeks, “Introduction of System Call.” Accessed: May 13, 2024. [Online]. Available: https://www.geeksforgeeks.org/introduction-of-system-call/

[11] A. Bansal, R. Awasthi, and A. Venkatrao, “Tracing System Calls Using eBPF - Part 1 | Falco.” Accessed: May 03, 2024. [Online]. Available: https://falco.org/blog/tracing-syscalls-using-ebpf-part-1/

[12] S. Das, “(14) What is Cloud Native & CNCF? Graduated Projects? Info Insight! | LinkedIn.” Accessed: May 03, 2024. [Online]. Available: https://www.linkedin.com/pulse/what-cloud-native-cncf-graduated-projects-info-insight-sandip-das-nhy9e/

[13] O. Yaşar, “What is Falco?” Accessed: May 13, 2024. [Online]. Available: https://medium.com/adessoturkey/what-is-falco-837298a066ee

[14] Plutora, “MTTD (Mean Time to Detect): Defined and Explained.” Accessed: May 03, 2024. [Online]. Available: https://www.plutora.com/blog/mttd-mean-time-to-detect-defined-explained

[15] Falco Authors, “Supported Fields for Conditions and Outputs,” The Falco Project, 2023. Accessed: May 25, 2025. [Online]. Available: https://falco.org/docs/reference/rules/supported-fields/

Perancangan Ulang *User Interface* dan *User Experience* pada *Website* Pelayanan PT Sukabumi Sinar Vision dengan Mempertimbangkan Faktor *Website Accessibility* bagi Pengguna Penderita *Color Vision Deficiency*

Harine Amalia Rahma[1], Margareta Hardiyanti[1,*]
Depertemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
harineamalia125@mail.ugm.ac.id
[*]Korespondensi: margareta.hardiyanti@ugm.ac.id ;

Abstract – Digitalization has become a primary focus of transformation, significantly impacting various operational elements of companies and their approaches to serving consumers. As entities responsible for providing digital services, it is crucial to pay attention to integrating accessibility for disabled users into the products offered. Accessibility can be achieved by implementing standards recognized both nationally and internationally. One widely known standard is the Web Content Accessibility Guidelines (WCAG). However, there are often issues in implementing these standards, such as color contrast and interface structure, which are frequently overlooked by web developers. These issues can hinder access for users with color blindness, necessitating special attention from web developers. Therefore, a redesign of the service website interface for PT Sukabumi Sinar Vision was carried out, prioritizing accessibility and employing a User-Centered Design (UCD) approach while considering WCAG guidelines for users with a history of color blindness. The redesign involved 10 users with color blindness. The results of the redesign showed improvements in various evaluation aspects compared to the evaluations of the previous official website interface. The System Usability Scale (SUS) score increased from 70.25 to 80.5, the success rate from 75% to 92.27%, and the overall average score of the Short User Experience Questionnaire (UEQ-S) improved from -0.050 in the "bad" category to 1.900 in the "excellent" category. These findings indicate that the application of the UCD method and WCAG guidelines effectively enhances the accessibility of website interfaces for users with a history of color blindness, thus creating a more user-friendly website for them.

Keywords – Color Vision Deficiency, Web Content Accessibility Guidelines, User-Centered Design, Usability Testing, User Experience Questionnaire

Intisari – Digitalisasi telah menjadi fokus utama transformasi yang memberikan dampak signifikan terhadap berbagai elemen operasional perusahaan serta pendekatan dalam melayani konsumen. Sebagai pihak yang bertanggung jawab atas penyediaan layanan digital, penting untuk memperhatikan integrasi aksesibilitas bagi pengguna disabilitas dalam produk-produk yang disediakan. Aksesibilitas dapat dicapai melalui penerapan standar yang diakui baik secara nasional maupun internasional. Salah satu standar yang dikenal secara luas adalah *Web Content Accessibility Guideline* (WCAG). Akan tetapi, kerap terjadi permasalahan dalam penerapan standar tersebut, seperti kontras warna dan struktur antarmuka yang sering diabaikan oleh pengembang situs web. Permasalahan tersebut tentu saja dapat menghambat akses bagi pengguna yang memiliki kondisi buta warna, sehingga perlu adanya perhatian khusus dari pengembang situs web. Oleh karena itu, dilakukan perancangan ulang antarmuka situs web pelayanan PT Sukabumi Sinar Vision yang mengedepankan aksesibilitas dan menerapkan pendekatan metode *User-Centered Design* (UCD) dengan mempertimbangkan pedoman WCAG untuk pengguna dengan riwayat buta warna. Perancangan ulang situs web PT Sukabumi Sinar Vision melibatkan 10 pengguna dengan kondisi buta warna. Hasil dari perancangan ulang menunjukkan peningkatan nilai dalam berbagai aspek evaluasi, dibandingkan dengan hasil evaluasi antarmuka situs web resmi sebelumnya. Nilai *System Usability Scale* (SUS) dari 70,25 menjadi 80,5, *success rate* dari 75% menjadi 92,27%, dan rata-rata keseluruhan nilai *User Experience Questionnaire (UEQ)* dari -0,050 masuk pada kategori *bad* menjadi kategori *excellent* dengan nilai rata-rata keseluruhan 1,900. Temuan ini menunjukkan bahwa penerapan metode UCD dan pedoman WCAG efektif dalam meningkatkan aksesibilitas antarmuka situs web bagi pengguna dengan riwayat buta warna, sehingga menciptakan situs web yang lebih ramah pengguna bagi mereka.

Kata kunci – Buta Warna, *Web Content Accessibility Guidelines*, *User-Centered Design*, *Usability Testing*, *User Experience Questionnaire*

I. PENDAHULUAN

Digitalisasi telah menjadi fokus utama transformasi yang berdampak signifikan pada operasi perusahaan dan pendekatan layanan konsumen. Kepopuleran layanan digital mendorong berbagai mitra untuk beralih ke platform *online*. Perusahaan yang melakukan digitalisasi harus mengintegrasikan aksesibilitas sebagai inti dari produk digital mereka, terutama untuk pengguna dengan disabilitas. Standar internasional yang diakui untuk aksesibilitas adalah *Web Content Accessibility Guideline* (WCAG), yang dikembangkan oleh World Wide Web Consortium (W3C) [1]. WCAG menyediakan pedoman untuk merancang situs web yang mudah diakses oleh pengguna dengan berbagai kondisi disabilitas, termasuk penglihatan, pendengaran, fisik, bicara, kognitif, bahasa, pembelajaran, dan neurologis [2].

Hasil evaluasi terhadap 34 situs web pemerintah provinsi di Indonesia menggunakan WCAG 2.1 menemukan bahwa masalah kontras warna adalah salah satu permasalahan umum

yang mempengaruhi aksesibilitas bagi pengguna dengan buta warna [3]. Kontras warna merupakan komponen penting bagi pengguna buta warna untuk melakukan navigasi dan interaksi dengan objek berwarna [4]. Menurut data Riset Kesehatan Dasar (RISKESDAS) 2007, prevalensi buta warna di Indonesia mencapai 7,4% dari populasi, dan jumlah ini meningkat seiring waktu [5].

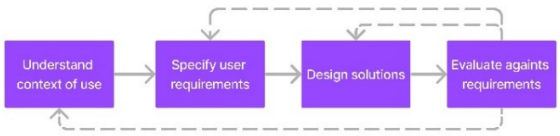
Sebagai perusahaan yang bergerak dalam bidang *Internet Service Provider* (ISP), pemasangan TV *Cable*, dan penyedia saluran TV lokal, PT Sukabumi Sinar Vision memiliki cakupan latar belakang pelanggan yang begitu luas. Oleh karena itu, perusahaan perlu memperhatikan aksesibilitas dalam pengembangan situs web pelayanan yang dimiliki. Pihak PT Sukabumi Sinar Vision menyatakan bahwa situs web yang berhasil dikembangkan masih mengalami beberapa kekurangan. Salah satunya adalah tampilan antarmuka belum mempertimbangkan aspek aksesibilitas khususnya bagi pengguna buta warna. Terdapat beberapa tampilan antarmuka dengan pemilihan warna yang masih belum konsisten. Selain itu, kontras warna dari setiap elemen situs web masih tidak terlihat jelas oleh pengguna. Kekurangan tersebut berdampak terhadap kepercayaan pelanggan dan *branding* PT Sukabumi Sinar Vision sebagai penyedia layanan. Untuk memenuhi kebutuhan pengguna buta warna, maka diperlukan perancangan ulang menggunakan metode yang mempertimbangkan aspek kegunaan, aksesibilitas, dan pengalaman pengguna secara bersamaan [6].

II. DASAR TEORI

A. Metode Perancangan

1. *User-Centered Design* (UCD)

User-Centered Design (UCD) merupakan sebuah praktik atau sebuah metode desain yang tujuan utamanya untuk digunakan oleh manusia yang melibatkan manusia di dalam proses desainnya. Dapat disimpulkan bahwa *User-Centered Design* merupakan desain proses yang melibatkan pengguna dalam proses nya. Proses UCD membutuhkan eksperimen, iterasi, dan pengalaman ketika mengalami kegagalan dalam pengoperasiannya [7]. Proses UCD ditampilkan pada Gambar 1.



Gambar 1. Proses *User-Centered Design* 1

a. *Understand context of use*

Perancang sistem perlu mengerti konteks dari penggunaan sistem seperti siapa pengguna aplikasi, untuk apa pengguna menggunakan aplikasi tersebut, dan pada situasi seperti apa mereka menggunakan aplikasi tersebut. Tahapan ini mencakup riset dan analisis kebutuhan pengguna dan mengetahui target pasar dari produk yang dikembangkan.

b. *Specify user requirements*

Setelah perancang mengerti konteks penggunaan dari aplikasi, perancang dapat melanjutkan kepada proses selanjutnya yaitu menentukan kebutuhan pengguna. Pada proses ini, perancang perlu menentukan kebutuhan pengguna dari segi bisnis maupun dari segi pencapaian objektif. Hal tersebut mencakup perancangan konsep pada produk yang akan dikembangkan.

c. *Design solutions*

Proses selanjutnya adalah merancang solusi dan mengimplementasikan kebutuhan pengguna yang telah di dapat pada proses sebelumnya.

d. *Evaluate against requirements*

Proses evaluasi dilakukan bersama pengguna yang akan menggunakan produk yang sedang dikembangkan.

2. *Web Content Accessibility Guideline* (WCAG)

Web Content Accessibility Guidelines (WCAG) merupakan dokumen yang menjelaskan tata cara membuat konten situs web yang lebih mudah diakses oleh pengguna. WCAG versi 2 berfokus pada memudahkan akses bagi pengguna dengan disabilitas dan telah diakui sebagai standar ISO: ISO/IEC 40500:2012. WCAG 2 memiliki 12-13 panduan yang dikelompokkan menjadi empat prinsip utama: dapat dipersepsi (*perceivable*), dapat dioperasikan (*operable*), dapat dipahami (*understandable*), dan kuat (*robust*). Setiap panduan memiliki kriteria keberhasilan yang dapat diuji, terbagi dalam tiga level: A, AA, dan AAA, yang menentukan kesesuaian dengan WCAG 2.

WCAG 2.1 memberikan rekomendasi luas untuk meningkatkan aksesibilitas konten web bagi individu dengan berbagai disabilitas, termasuk kebutaan, gangguan penglihatan, tuli, gangguan pendengaran, keterbatasan gerak, dan disabilitas bicara. Kriteria keberhasilan WCAG 2.1 diformulasikan sebagai pernyataan yang dapat diuji, dengan panduan khusus untuk memenuhi kriteria tersebut dalam teknologi spesifik [8].

3. Figma

Figma merupakan sebuah aplikasi desain untuk membuat *prototype* dan *user interface* untuk produk digital seperti aplikasi berbasis gawai dan aplikasi situs web. Figma memiliki alat khusus yang dirancang untuk membuat prototype dan desain UI. Figma memiliki fasilitas untuk menganimasikan prototype sehingga bisa menciptakan pengalaman interaktif bagi pengembang dan juga subjek untuk pengujian [9].

4. *Prototype*

Prototype merupakan representasi dari gagasan desain yang digunakan untuk proses uji coba dan evaluasi. Karakteristik prototype dapat bervariasi dalam tingkat kemiripan dengan produk akhir dan jenis media yang digunakan, termasuk alam bentuk komputer atau kertas [10].

B. Metode Pengujian

1. *Usability Testing*

Usability testing merupakan proses krusial yang ada pada desain produk dan pengembangan sebuah sistem. *Usability testing* berfokus pada mengevaluasi pengalaman pengguna pada sebuah produk. Hal ini melibatkan pengguna dalam berinteraksi dengan sebuah produk dan melakukan sebuah *task* yang sangat penting. Tujuan dari *usability testing* adalah untuk mengidentifikasi berbagai permasalahan penggunaan, memahami apa yang dapat bekerja dengan baik dan apa yang tidak bagi pengguna, serta mengumpulkan berbagai umpan balik dari pengguna untuk meningkatkan sebuah desain dan fungsionalitas sebuah produk [11].

Usability testing memuat berbagai pemahaman yang penting terhadap bagaimana pengguna berinteraksi dengan sebuah produk dan membantu desainer dan pengembang untuk membuat keputusan yang mengandung berbagai informasi penting. Salah satu pengukuran yang paling utama pada *usability testing* adalah pengukuran *success rate*. *Success rate* merupakan matriks yang mengukur seberapa jauh pemahaman pengguna terhadap sistem yang telah dibuat [12]. Perhitungan *success rate* ditampilkan pada (1).

$$Success\ Rate = \frac{Task\ sukses + (Task\ sebagian\ sukses \times 0.5)}{Jumlah\ task} \times 100\% \quad (1)$$

2. *System Usability Test* (SUS)

System Usability Scale (SUS) merupakan skala *usability* yang sederhana dan dapat diandalkan. SUS dapat memberikan penilaian yang subjektif terhadap kegunaan. SUS diukur dari skala 1 sampai 5 untuk mengukur ketergunaan seluruh system. Skala yang diberikan ini didasarkan pada metodologi skala Likert, yang dimana responden menunjukan tingkat persetujuan atau ketidaksetujuan responden terhadap 10 pernyataan pada kegunaan sistem. Nilai dari SUS dihitung dari menjumlahkan setiap nilai yang ada pada item dan mengalikan jumlah dari nilai tersebut dengan 2,5. Rentang nilai yang dihasilkan dari kalkulasi tersebut adalah dimulai dari 0 hingga 100. Semakin besar nilai yang dihasilkan, maka semakin baik kegunaan dalam sistem yang diuji [12]. Tabel 1 menampilkan daftar pernyataan SUS.

Tabel 1. Daftar pernyataan SUS

Kode	Pernyataan
QS-1	Saya berpikir akan menggunakan fitur yang ada pada situs web PT Sukabumi Sinar Vision lagi
QS-2	Saya merasa fitur yang ada pada situs web ini rumit untuk digunakan
QS-3	Saya merasa fitur pada situs web ini mudah untuk digunakan
QS-4	Saya membutuhkan bantuan dari orang lain dalam menggunakan fitur yang ada pada situs web ini
QS-5	Saya merasa fungsi-fungsi dalam situs web ini berjalan dengan semestinya
QS-6	Saya merasa ada banyak hal yang tidak konsisten pada rancangan situs web ini
QS-7	Saya merasa orang lain akan memahami cara menggunakan situs web ini dengan cepat
QS-8	Saya merasa fitur pada situs web ini membingungkan
QS-9	Saya merasa tidak ada hambatan dalam menggunakan fitur pada situs web ini
QS-10	Saya perlu membiasakan diri terlebih dahulu sebelum menggunakan fitur yang ada pada situs web ini

Responden menjawab pernyataan yang diberikan menggunakan skala Likert. Skala penilaian dimulai memberikan nilai 1 sampai 5. Nilai 1 menyatakan sangat tidak setuju, 3 menyatakan netral, dan 5 menyatakan sangat setuju dengan pernyataan yang diberikan. Berdasarkan daftar pernyataan pada tabel 1, pernyataan dikategorikan menjadi dua kategori. Pernyataan dengan kategori negatif dan kategori positif. Pernyataan kategori negatif ditandai dengan pernyataan urutan genap dan kategori positif ditandai dengan pernyataan urutan ganjil. Berikut merupakan cara untuk menghitung nilai dari SUS:

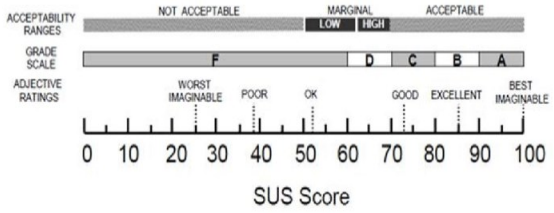
- Setiap pertanyaan kategori positif nilai yang diberikan dikurangi oleh 1. Sebagai contoh jika a=nilai yang diberikan maka (a-1).
- Setiap pertanyaan dengan kategori negatif mengurangkan nilai yang diberikan dengan 5. Sebagai contoh jika b=nilai yang diberikan maka (5-b).
- Tambahkan nilai-nilai dari kategori positif dan negatif. Hasil penjumlahan tersebut dikalikan dengan 2,5.

Setelah menghitung semua hasil nilai yang didapatkan, maka menjumlahkan semua total nilai yang didapat dari responden dan menghitung rata-rata dari nilai yang dijumlahkan. Untuk menghitung rata-rata keseluruhan dari nilai SUS menggunakan (2) sebagai berikut [13].

$$Rata - Rata\ Nilai\ SUS = \frac{Total\ nilai}{Jumlah\ responden} \quad (2)$$

Selanjutnya, untuk menentukan parameter keberhasilan dari hasil pengukuran menggunakan SUS diberikan

benchmark tingkat *acceptance*. Gambar 2 menunjukan *benchmark* tingkat *acceptance* dari nilai *usability* yang telah dihasilkan [13].



Gambar 2. *Benchmark* SUS

3. *Short User Experience Questionnaire* (UEQ-S)

Skala yang digunakan oleh *User Experience Questionnaire* (UEQ) dapat menyajikan kesan pengalaman pengguna secara menyeluruh. Hasil dari UEQ menentukan kualitas dari antar muka dan pengalaman pengguna yang sedang diuji. UEQ mengukur kedua aspek *usability*, yaitu aspek dari *user interface* (*efficiency, perspicuity, dependability*) dan aspek dari *user experience* (*novelty* dan *stimulation*). UEQ menggunakan skala tujuh tahap untuk mengurangi bias kecenderungan yang berpusat pada hal yang sudah familiar di mata pengguna. Bias tersebut mengakibatkan penilaian pengguna menjadi tidak seimbang atau tidak akurat. Tujuan dari penggunaan UEQ berupaya untuk mengurangi bias tersebut sehingga penilaian yang digunakan lebih objektif. UEQ-S berbeda dengan UEQ versi biasa. UEQ-S mengukur secara keseluruhan dan menyimpulkan hasil dari nilai kelompok kualitas pragmatis yang mencakup aspek evaluasi antarmuka (*Efficiency, Perspicuity, Dependability*) dan kelompok kualitas hedonis yang mencakup evaluasi pengalaman pengguna (*Stimulation, Novelty*) secara keseluruhan [14].

4. *Relative Luminance*

Relative luminance merupakan salah satu komponen penting bagi perhitungan rasio kontras [15]. Perhitungan rasio kontras dan *relative luminance* pada umumnya menggunakan sebuah komputasi atau aplikasi pengukur rasio kontras. Berikut merupakan perhitungan *relative luminanc*e sampai menjadi rasio kontras secara manual. Persamaan (3) menampilkan perhitungan *relative luminance* secara manual.

$$L = 0{,}2126 \times R^{\gamma} + 0{,}7152 \times G^{\gamma} + 0{,}0722 \times B^{\gamma} \quad (3)$$

L = *Relative Luminance* yang akan dihitung
R = nilai intensitas warna merah (0-255) dalam format RGB normal.
G = nilai intensitas warna hijau (0-255) dalam format RGB normal.
B = nilai intensitas warna biru (0-255) dalam format RGB normal.
Γ = parameter gamma yang biasanya bernilai 2,2 (sesuai dengan kebanyakan layar komputer)

Untuk mendapatkan R^{γ}, G^{γ}, dan B^{γ} perlu merubah nilai RGB yang telah dinormalisasi (RsRGB, GsRGB, dan BsRGB) menjadi RGB linear atau berdasarkan ketentuan berdasarkan (4), (5), dan (6) [8].

$$Jika\ RsRGB\ \leq 0{,}03928\ maka\ R = \frac{RsRGB}{12{,}92}$$
$$Jika\ tidak\ R = \left(\frac{RsRGB + 0{,}055}{1{,}055}\right)^{2{,}4} \quad (4)$$

$$Jika\ GsRGB\ \leq 0{,}03928\ maka\ G = \frac{RsRGB}{12{,}92}$$
$$Jika\ tidak\ G = \left(\frac{GsRGB + 0{,}055}{1{,}055}\right)^{2{,}4} \quad (5)$$

$$Jika\ BsRGB\ \leq 0{,}03928\ maka\ B = \frac{RsRGB}{12{,}92}$$
$$Jika\ tidak\ B = \left(\frac{RsRGB + 0{,}055}{1{,}055}\right)^{2{,}4} \quad (6)$$

Untuk mendapatkan nilai RsRGb, GsRGB, dan BsRGB, perlu menormalisasi nilai RGB dengan membagi nilai RGB asli dengan 255 untuk mendapatkan nilai dari rentang 0 sampai 1. Contoh persamaan dapat dilihat pada persamaan (7), (8), dan (9).

$$\text{RsRGB} = \frac{R_{8bit}}{255} \quad (7)$$
$$\text{GsRGB} = \frac{G_{8bit}}{255} \quad (8)$$
$$\text{BsRGB} = \frac{B_{8bit}}{255} \quad (9)$$

Selanjutnya, menghitung rasio kontras yang dimiliki oleh warna *background* dan *foreground.* Perhitungan rasio kontras dapat dilihat pada persamaan (10) [8].

$$Rasio\ Kontras = \frac{(L1 + 0{,}05)}{(L2 + 0{,}05)} \quad (10)$$

L1 = *Relative luminance* dari warna terang (*foreground color*)
L2 = *Relative luminance* dari warna gelap (*background color*)

III. METODOLOGI

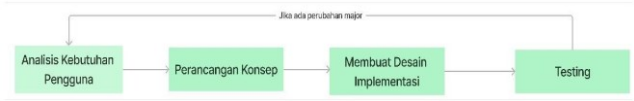
A. Alat Perangkat Lunak

Dalam penelitian ini, menggunakan beberapa alat perangkat lunak, seperti Figma dan Maze sebagai alat perancang antarmuka. Selanjutnya, Google Meet, Google Form, Maze, dan Short UEQ Data Analysis Tools sebagai alat evaluasi *usability test.* Terakhir, sebagai alat pengujian rasio kontras menggunakan WAVE, *Simple WCAG 2.1 Contrast*

Checker, Use Contrast, dan *Color Check For ADA Image Compliance.*

B. Tahapan Penelitian

Tahapan penelitian yang digunakan pada pelaksanaan penelitian mengadopsi tahapan dari metode UCD, gambaran tahapan penelitian secara keseluruhan ditampilkan pada Gambar 3.



Gambar 3. Tahapan penelitian

1. Analisis Kebutuhan Pengguna

Proses ini diawali dengan tahap perekrutan responden. Peneliti merekrut responden dari berbagai komunitas yang ada pada seluruh platform media sosial, termasuk Komunitas Buta Warna Parsial Indonesia yang terdapat pada platform Facebook dan Telegram. Setelah melewati masa perekrutan dan penyeleksian responden, terkumpul 10 responden buta warna Dikromatik yaitu 6 responden dengan kondisi Tritanopia dan 4 responden dengan kondisi Protanopia. Peneliti juga merekrut 15 orang responden dengan kondisi pengelihatan normal untuk memastikan bahwa kedua rancangan ulang dapat diakses oleh masyarakat umum.

Setelah berhasil merekrut responden, peneliti melakukan *in-depth interview* bersama responden buta warna. Tujuan dari *in-depth interview* untuk mengetahui informasi kebiasaan dan kendala yang dialami oleh responden. Tabel 2 menampilkan kesimpulan dari kebiasaan responden terhadap pemakaian situs web yang sesuai dengan kelompok-kelompok tertentu. Hasil dari kesimpulan tersebut disajikan kedalam bentuk *user group*.

Kendala yang dialami oleh responden buta warna ketika menggunakan situs web secara umum meliputi tumpang tindih warna latar belakang dengan teks, variasi warna yang berlebihan, bahasa yang membingungkan, ikon yang tidak representatif dan *action button* yang kurang jelas.

Tabel 2. *User group* pengguna buta warna

Karakteristik Pengguna	Karakteristik Pengguna Berdasarkan Kelompok	
	Tritanopia	Protanopia
Keterbatasan Fisik	Memiliki gangguan penglihatan terhadap warna kuning-biru	Memiliki gangguan penglihatan terhadap warna merah-hijau
Penggunaan Situs web	Dari skala 6-10 jam setiap hari	
Motivasi	Sangat termotivasi untuk membuka situs webnpada bidang pendidikan, hiburan dan media sosial.	
Sikap	Situs web dengan keragaman sikap, terutama pada alur atau komponen kontras minim, dapat mendorong rasa ingin tahu tinggi pada pengguna untuk memahami informasi dan elemen yang kurang jelas. Beberapa juga memilih untuk menyerah untuk memahami informasi jika komponen penyedia informasi tidak jelas.	

Setelah melakukan *in-depth interview* bersama responden buta warna. Peneliti menguji situs web resmi PT Sukabumi Sinar Vision dengan metode *usability test.* Hasil dari *usability test* menunjukan bahwa *success rate* yang dihasilkan mencapai angka 75%, nilai *satisfaction rate* (SUS) mencapai 70,25, hasil dari *benchmark* UEQ-S termasuk kedalam kategori *bad.* Tabel 3 menampilkan daftar *task* yang dilakukan oleh responden dan Tabel 4 menampilkan rincian kegagalan yang umum terjadi oleh responden pada setiap *task.*

Tabel 3. Daftar *task* pengujian situs web resmi

Kode	*Task*
TS-1	Masuk Akun
TS-2	Menghubungi kontak narahubung secara langsung
TS-3	Mendaftar layanan
TS-4	Mengakses konten *Video on Demand*
TS-5	Siaran live
TS-6	Profil perusahaan

Tabel 4. Rincian kegagalan *task* situs web resmi

Kode *Task*	Permasalahan	Komponen
TS-2	Warna yang tumpang tindih dengan warna latar belakang.	*Floating button* pada halaman beranda.
TS-3	Fitur tidak ada pada daftar navigasi, sehingga responden tidak dapat mengidentifikasi fitur.	Fitur mendaftar layanan.
TS-4	Kesalahpahaman responden terhadap konten dari fitur yang dimaksud dengan konten fitur lain yang disediakan.	Fitur "*Video on Demand*" dengan konten siaran *live* pada halaman beranda.
TS-6	Penggunaan bahasa yang asing bagi penamaan fitur.	Penamaan fitur profil perusahaan dengan nama "*Company Profile*".

Selanjutnya, setelah selesai melakukan *usability test*, peneliti membandingkan situs web resmi PT Sukabumi Sinar

Vision dengan tiga situs web perusahaan kompetitor yang bergerak pada bidang yang serupa. Tabel 5 menunjukan hasil *competetive analysis* yang disajikan kedalam bentuk analisis SWOT. Peneliti membandingkan hasil pengujian contrast error yang terjadi pada situs web menggunakan alat pengujian aksesibilitas WAVE dan membandingkan fitur-fitur yang tersedia pada masing-masing perusahaan.

Tabel 5. Hasil Analisis SWOT

Strengths	***Weaknesses***
Alur yang cukup mudah untuk setiap fiturnya	Penamaan fitur dengan istilah yang asing sehingga membuat pelanggan/pengguna bingung dan gagal mengakses fitur
Situs web masuk ke dalam kategori *acceptable* untuk kepuasan pengguna.	Beberapa komponen penting untuk standar situs web masih kurang penerapannya.
	Jumlah kesalahan kontras yang banyak dibanding kompetitornya.
Opportunities	***Threats***
Menyediakan pelayanan dan fitur yang dapat bersaing dengan kompetitornya.	Inkonsistensi desain dan pemilihan kata dalam penamaan fitur yang tidak familiar mempengaruhi psikologis pelanggan/pengguna untuk beralih jasa pelayanan.
	Beberapa kompetitor memiliki fitur dan layanan serupa yang antarmuka situs webnya lebih baik

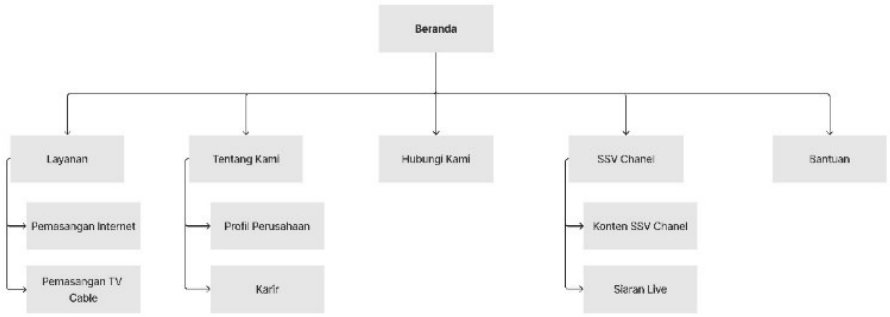
2. Perancangan Konsep

Berdasarkan pengkajian ulang hasil analisis kebutuhan pengguna, dapat disimpulkan bahwa ada beberapa bagian yang diperlukan perbaikan dan juga penambahan. Hasil *usability test* yang dilakukan pada situs web resmi PT Sukabumi Sinar Vision menunjukan bahwa mayoritas responden mengalami kesalahan akses pada fitur menghubungi kontak narahubung secara langsung dan fitur "*Video on Demand*". Selain itu, responden mengalami kegagalan saat melakukan *task* yang mengacu pada fitur mendaftar layanan dan profil perusahaan.

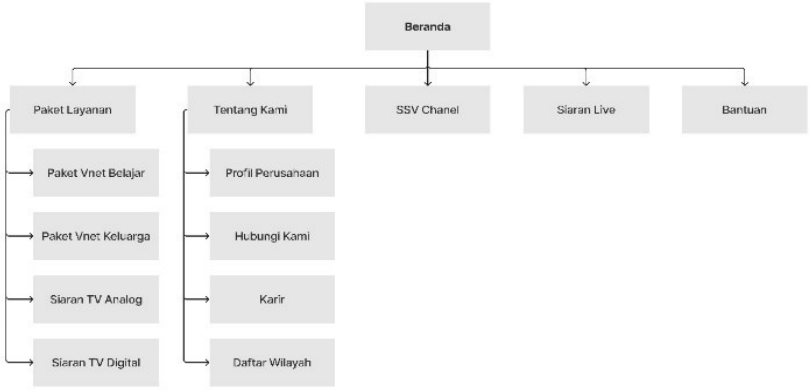
Hasil *competitive analysis* yang berupa analisis SWOT dari segi *weakness* ditemukan bahwa kelengkapan fitur standar perlu diperhatikan. Kelengkapan fitur yang perlu diperhatikan adalah fitur masuk akun yang belum lengkap kebutuhannya. Selain kelengkapan fitur, penggunaan bahasa yang masih asing untuk pengguna juga perlu diperhatikan. Hasil analisis SWOT dari segi *opportunities* menekankan bahwa dengan menghadirkan fitur yang dapat bersaing dengan perusahaan sebidang, dapat membantu perkembangan situs web PT Sukabumi Sinar Vision. Pihak PT Sukabumi Sinar Vision menegaskan perlunya penambahan fitur karir yang sejalan dengan tujuan perusahaan. Selain itu, menambahkan fitur registrasi untuk melengkapi fitur masuk akun yang sudah ada. Uraian ini menjadi salah satu pedoman peneliti dalam merancang *user flow* situs web PT Sukabumi Sinar Vision. Peneliti merancang dua versi *user flow* yang berbeda untuk kedua versi desain. Peneliti menata konten yang ada pada *user-flow* ke dalam struktur navigasi. Gambar 4 dan Gambar 5 menampilkan struktur navigasi dari kedua versi desain.

3. Membuat Desain Implementasi

Berdasarkan uraian kendala yang dialami oleh pengguna buta warna dan hasil analisis SWOT, peneliti berupaya untuk menyelesaikan kendala dan permasalahan yang ada dengan pertimbangan perancangan ulang antarmuka yang memenuhi berbagai kriteria keberhasilan pedoman *Web Content Accessibility Guidelines* (WCAG) versi 2.1 yang relevan dengan perancangan UI/UX. Kriteria keberhasilan mencakup kriteria keberhasilan *non-text content* (A*), use of color* (A), *headings and labels* (AA), *contrast (minimum)* (AA), *non-text*



Gambar 4. Struktur navigasi desain versi 1



Gambar 5. Struktur navigasi desain versi 2

contrast (AA), dan *text spacing* (AA). Tabel 6 menampilkan rincian dari kriteria keberhasilan yang menjadi pedoman pada tahap ini.

Tabel 6. Rincian kriteria keberhasilan

Kriteria Keberhasilan	Teknik yang digunakan
1.1.1.Non-text Content (A)[b]	G96: Menyediakan alternatif teks pendek yang dapat memberikan deskripsi singkat terhadap konten *non-text.* G92: Menyediakan deskripsi panjang bagi konten *non-text* yang memiliki tujuan dan informasi yang serupa.
1.4.1.*Use of Color (A)*	G14: Memastikan informasi yang disajikan melalui perbedaan warna juga dapat diakses melalui teks. G111: Menggunakan pola dan warna
1.4.3.*Contrast (Minimum) (AA)*	G18: Rasio kontras minimal 4.5:1 antara teks (atau gambar teks) dan latar belakang teks.
1.4.11. *Non-text Contrast*[2] *(AA)*	G145: Rasio kontras minimal 3:1 antara teks (atau gambar teks) dan latar belakang di balik teks. Teknik ini melonggarkan persyaratan rasio kontras 4,5:1 untuk teks yang setidaknya berukuran 18 poin (jika tidak tebal) atau setidaknya 14 poin (jika tebal).
1.4.12.*Text Spacing (AA)*	Spasi baris minimal 1,5 kali ukuran huruf; jarak antar paragraf setidaknya 2 kali ukuran huruf; spasi antar huruf setidaknya 0,12 kali ukuran huruf; spasi antar kata minimal 0,16 kali ukuran huruf.
2.4.6 *Headings and Labels (AA)*	G130: Memberikan judul yang menjelaskan secara rinci. G131: Memberikan label yang menjelaskan secara rinci.

Peneliti menggunakan kombinasi warna pada kedua versi desain untuk mengetahui preferensi pengguna. Desain versi 1 memiliki kombinasi warna terang dan desain versi 2 yang memiliki kombinasi warna gelap. Untuk memastikan bahwa kombinasi warna tersebut sudah sesuai dengan kriteria keberhasilan WCAG, peneliti menguji tampilan desain menggunakan *plug in* yang disediakan oleh *platform* Figma yaitu *Simple WCAG 2.1 Contrast Checker* dan *Use Contrast.*

4. *Testing*

Setelah kedua rancangan produk selesai dirancang, tahap pengujian dilaksanakan dengan melibatkan 10 responden buta warna dengan jenis buta warna Tritanopia dan Protanopia. Selain melibatkan responden buta warna, peneliti juga memastikan agar desain yang telah dirancang dapat diakses oleh masyarakat umum. Oleh karena itu, peneliti melibatkan 15 orang responden dengan pengelihatan normal untuk menguji kedua jenis desain yang telah dirancang oleh peneliti. Pendekatan yang digunakan pada proses ini adalah pendekatan dengan metode pengujian *usability test*. Pada akhir tahap *testing*, responden diminta untuk memilih desain mana yang menurut mereka memberikan kenyamanan dan preferensi terbesar. Hasil dari pemilihan ini akan digunakan untuk menentukan desain akhir dari perancangan situs web pelayanan yang disediakan oleh PT Sukabumi Sinar Vision. Pengukuran yang digunakan pada penelitian ini adalah *satisfaction rate* menggunakan SUS untuk mengukur kepuasan pengguna, *success rate* untuk mengukur sejauh mana penalaran responden informasi yang disajikan, dan UEQ-S guna mengukur kelayakan kualitas pengalaman pengguna dan antar muka yang telah berhasil dirancang.

IV. HASIL DAN PEMBAHASAN

A. Hasil Proses Membuat Desain Implementasi

Hasil dari proses ini menghasilkan rancangan antar muka dari dua versi desain. Untuk memenuhi kriteria keberhasilan *use of color* (A), peneliti menggunakan teknik G111 dengan pemilihan warna yang telah ditentukan pada bagian sebelumnya. Gambar 6 menampilkan hasil desain UI halaman beranda situs web dari kedua versi. Desain versi 1 terdapat

pada sebelah kiri dan desain versi 2 terdapat pada sebelah kanan. Untuk memastikan apakah penggunaan warna pada elemen-elemen situs web sudah memenuhi kriteria keberhasilan, peneliti menggunakan dua kriteria keberhasilan WCAG 2.1 sebagai acuan. Dua kriteria keberhasilan yang digunakan untuk pengukuran penggunaan warna adalah *contrast* (minimum) (AA) yang menguji kontras elemen teks dengan latar belakang dan kriteria keberhasilan *non-text contrast* (AA) yang mengukur kontras elemen seperti ikon, gambar, sampai button aksi pada situs web. Peneliti menguji elemen-elemen berwarna menggunakan *Simple WCAG 2.1 Contrast Checker*, *Use Contrast,* dan *Color Check For ADA Image Compliance.* Tabel 7 dan Tabel 8 menampilkan rincian dari rasio kontras yang dihasilkan.

Gambar 6. Hasil Perancangan Ulang *User Interface*

Tabel 7. Rasio kontras desain versi 1

Teknik yang digunakan	*Contrast (Minimum)*	*Non-Text Contrast*
G145	6.03:1, 7.05:1, dan 14.88:1.	*Button* dan navigasi menghasilkan kontras 6.0:1, 3.9:1, 3.3:1, 3.44:1, 6.3:1, 3.9:1, dan 8.1. Kontras dari gambar menghasilkan rasio 4.13:1, 7.78:1, dan 4.18:1

Tabel 8. Rasio kontras desain versi 2

Teknik yang digunakan	*Contrast (Minimum)*	*Non-Text Contrast*
G18	8.31:1, 10.92:1, 13.15:1, dan 17.30:1	*Button* dan navigasi. Menghasilkan kontras 8.3:1, 13.3:1, dan 13.2:1. Kontras dari gambar
G145	-	menghasilkan rasio 6.5:1, 5.32:1, dan 3.64:1

B. Hasil *Usability Test*

Pengujian dilakukan terhadap dua versi desain yang telah berhasil dirancang. Tabel 9 menampilkan daftar *Task* untuk *usability test.*

Tabel 9. Daftar *task usability test*

Kode	*Task*
TS-1	Membuat akun baru
TS-2	Lupa password
TS-3	Masuk akun
TS-4	Profil perusahaan
TS-5	Mencari narahubung perusahaan
TS-6	Mendaftar daerah jangkauan layanan
TS-7	Mendaftar lowongan kerja
TS-8	Mendaftar paket layanan
TS-9	Menikmati konten SSV
TS-10	Menikmati siaran langsung
TS-11	Mengelola profil

1. *Usability Test* Desain Versi 1

Pengukuran pertama yang peneliti lakukan ketika *usability test* adalah mengukur seberapa jauh pemahaman responden terhadap tampilan desain versi 1 berdasarkan task yang diberikan. Pengukuran yang digunakan adalah pengukuran *success rate.* Tabel 10 menampilkan jumlah *task* sukses, *task* setengah sukses, dan kegagalan yang dialami oleh responden buta warna.

Tabel 10. Hasil *usability test* responden buta warna desain versi 1

Jumlah *Task* Keseluruhan	*Task* Sukses	*Task* Setengah Sukses	Gagal
110	92	16	2

Setelah mengetahui hasil tersebut, peneliti memasukan hasil kedalam perhitungan *success rate*.

$$Success\ Rate = \frac{Task\ sukses + (Task\ sebagian\ sukses \times 0{,}5)}{Jumlah\ task} \times 100\%$$

$$Success\ Rate = \frac{92 + (16 \times 0{,}5)}{110} \times 100\%$$

$$Success\ Rate = 90\%$$

Untuk memastikan bahwa desain versi 1 dapat diterima oleh masyarakat umum, peneliti melakukan pengujian bersama responden pengelihatan normal. Tabel 11 menampilkan rincian hasil pengerjaan *task.*

Tabel 11. Hasil *usability test* responden pengelihatan normal desain versi 1

Jumlah *Task* Keseluruhan	*Task* Sukses	*Task* Setengah Sukses	Gagal
165	154	11	0

$$Success\ Rate = \frac{Task\ sukses + (Task\ sebagian\ sukses \times 0{,}5)}{Jumlah\ task} \times 100\%$$

$$Success\ Rate = \frac{154 + (11 \times 0{,}5)}{165} \times 100\%$$

$$Success\ Rate = 96{,}6\%$$

Setelah menghitung *success rate*, menghitung hasil dari pengujian menggunakan SUS dari kedua sudut pandang responden. Nilai Akhir SUS yang dihasilkan dari responden buta warna sebanyak 800 dan responden pengelihatan normal sebanyak 1357,5. Kedua nilai akhir tersebut dimasukan pada persamaan rata-rata nilai SUS. Berikut merupakan rata-rata dari responden buta warna.

$$Rata - rata\ Nilai\ SUS = \frac{Total\ nilai}{Jumlah\ responden}$$

$$Rata - rata\ Nilai\ SUS = \frac{800}{10}$$

$$Rata - rata\ Nilai\ SUS = 80$$

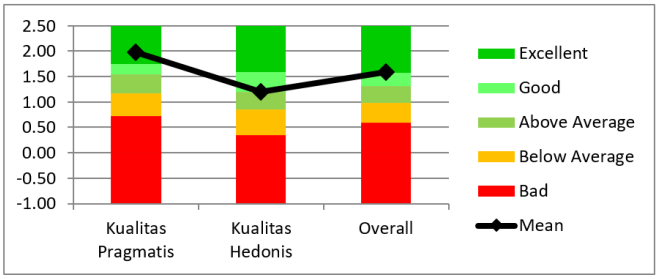
Selanjutnya, menghitung rata-rata dari responden pengelihatan normal.

$$Rata - rata\ Nilai\ SUS = \frac{Total\ nilai}{Jumlah\ responden}$$

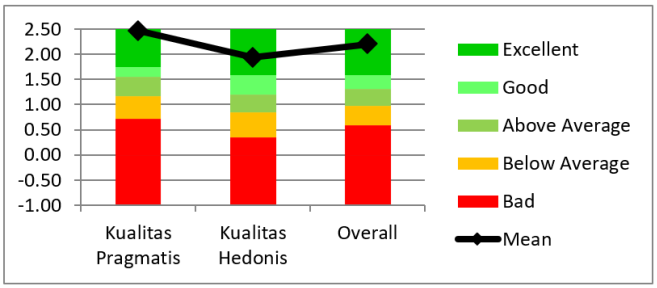
$$Rata - rata\ Nilai\ SUS = \frac{1357{,}5}{15}$$

$$Rata - rata\ Nilai\ SUS = 90{,}5$$

Berdasarkan hasil dari kedua perhitungan SUS dapat disimpulkan bahwa pengguna normal dan buta warna sangat puas dengan desain versi 1. Berdasarkan benchmark SUS, desain versi 1 memasuki kategori *acceptable* dan *best imaginable.* Setelah berhasil mengukur SUS, peneliti mengukur UEQ-S menggunakan *Short UEQ Data Analysis Tools*. Gambar 7 dan Gambar 8 merupakan hasil dari kedua sudut pandang responden menurut benchmark UEQ-S.



Gambar 7. Hasil UEQ-S responden buta warna desain versi 1



Gambar 8. Hasil UEQ-S responden pengelihatan normal desain versi 1

Berdasarkan hasil dari kedua perhitungan UEQ-S pada uraian sebelumnya, dapat disimpulkan bahwa desain versi 1 memiliki kualitas pengalaman pengguna dan antar muka yang sangat baik bagi kedua sudut pandang responden.

2. *Usability Test* Desain Versi 2

Pengukuran masih serupa dengan pengukuran yang dilakukan pada desain versi 1. Tabel 12 dan Tabel 13 menampilkan hasil *usability test* responden buta warna dan responden pengelihatan normal.

Tabel 12. Hasil *usability test* responden buta warna desain versi 2

Jumlah *Task* Keseluruhan	*Task* Sukses	*Task* Setengah Sukses	Gagal
110	94	16	0

$$Success\ Rate = \frac{Task\ sukses + (Task\ sebagian\ sukses \times 0.5)}{Jumlah\ task} \times 100\%$$

$$Success\ Rate = \frac{94 + (16 \times 0{,}5)}{110} \times 100\%$$

$$Success\ Rate = 92{,}27\%$$

Tabel 13. Hasil *usability test* responden pengelihatan normal desain versi 2

Jumlah *Task* Keseluruhan	*Task* Sukses	*Task* Setengah Sukses	Gagal
165	161	4	0

$$Success\ Rate = \frac{Task\ sukses + (Task\ sebagian\ sukses \times 0{,}5)}{Jumlah\ task} \times 100\%$$

$$Success\ Rate = \frac{161 + (4 \times 0{,}5)}{165} \times 100\%$$

$$Success\ Rate = 98{,}7\%$$

Nilai Akhir SUS yang dihasilkan dari responden buta warna sebanyak 805 dan responden pengelihatan normal sebanyak 1387,5. Kedua nilai akhir tersebut dimasukan pada persamaan rata-rata nilai SUS. Berikut merupakan rata-rata dari responden buta warna.

$$Rata - rata\ Nilai\ SUS = \frac{Total\ nilai}{Jumlah\ responden}$$

$$Rata - rata\ Nilai\ SUS = \frac{805}{10}$$

$$Rata - rata\ Nilai\ SUS = 80{,}5$$

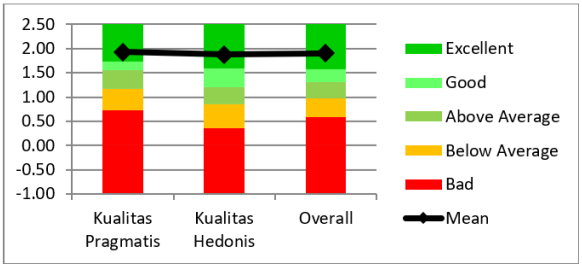
Selanjutnya, menghitung rata-rata dari responden pengelihatan normal.

$$Rata - rata\ Nilai\ SUS = \frac{Total\ nilai}{Jumlah\ responden}$$

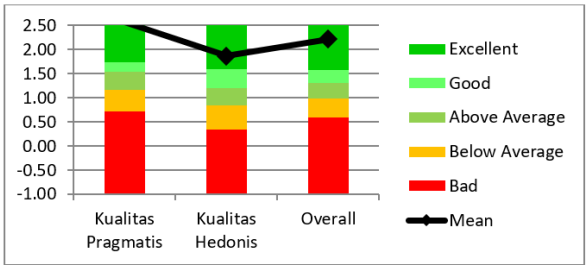
$$Rata - rata\ Nilai\ SUS = \frac{1387{,}5}{15}$$

$$Rata - rata\ Nilai\ SUS = 92{,}5$$

Berdasarkan benchmark SUS, desain versi 2 meemasuki kategori *acceptable dan best imaginable.* Setelah berhasil mengukur SUS, peneliti mengukur UEQ-S menggunakan *Short UEQ Data Analysis Tools*. Gambar 9 dan Gambar 10 merupakan hasil dari kedua sudut pandang responden menurut benchmark UEQ-S.



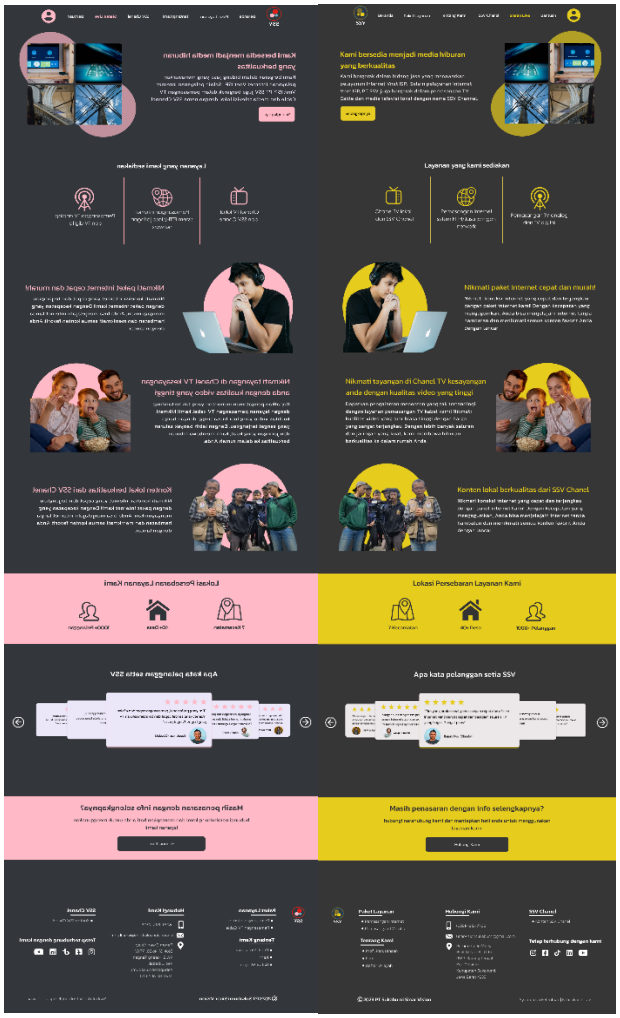
Gambar 9. Hasil UEQ-S responden buta warna desain versi 2



Gambar 10. Hasil UEQ-S responden pengelihatan normal desain versi 2

Berdasarkan hasil dari kedua perhitungan UEQ-S pada uraian sebelumnya, dapat disimpulkan bahwa desain versi 2 memiliki kualitas pengalaman pengguna dan antar muka yang sangat baik bagi kedua sudut pandang responden.

Hasil dari pemungutan suara untuk pemilihan desain final mayoritas responden buta warna dan responden pengelihatan normal memilih desain versi 2 sebagai desain final sebagai tampilan situs web resmi PT Sukabumi Sinar Vision. Gambar 11 menampilkan desain versi 2 dari sudut pandang pengguna Tritanopia dan Protanopia. Gambar yang berada disebelah kanan merupakan sudut pandang dari pengguna Tritanopia dan gambar yang berada disebelah kiri merupakan sudut pandang dari pengguna Protanopia.



Gambar 11. Sudut pandang desain final dari pengguna Tritanopia dan Protanopia

V. SIMPULAN

Berdasarkan hasil *in-depth interview*, meskipun memiliki perbedaan jenis, pengguna buta warna menunjukkan kebiasaan serupa dalam menggunakan situs web. Mereka mengalami kesulitan dengan kontras rendah, warna tumpang tindih, variasi warna berlebihan, bahasa membingungkan,

ikon tidak representatif, dan tombol aksi yang tidak jelas. Untuk meningkatkan aksesibilitas bagi pengguna buta warna, situs web perlu memenuhi kriteria WCAG 2.1 diantaranya adalah *Non-text Content*, *Use of Color*, *Contrast Minimum*, *Non-text Contrast*, *Text Spacing*, dan *Headings and Labels*.

Berdasarkan pengujian dari dua versi desain perancangan ulang, mayoritas pengguna memilih desain versi 2 dengan mode gelap. Hasil *Usability test* dari desain versi 2 dan situs web resmi mengalami peningkatan yang cukup signifikan. Hasil *usability test* menunjukkan *success rate* dari 75% menjadi 92,27%, *satisfaction rate* (SUS) dari 70,25 menjadi 80,5, dan hasil *benchmark* UEQ-S dari kategori "*bad*" menjadi kategori "*excellent*". Hal ini menandakan bahwa metode UCD dan pedoman WCAG merupakan kombinasi metode yang tepat untuk perancangan antarmuka yang inklusif bagi pengguna buta warna.

REFERENSI

[1] S. Horton, "Empathy Cannot Sustain Action in Technology Accessibility," *Front Comput Sci*, vol. 3, no. 1, 2021, doi: 10.3389/fcomp.2021.617044.

[2] Gronseth Susie, "Inclusive Design for Online and Blended Courses: Connecting Web Content Accessibility Guidelines and Universal Design for Learning," 2018.

[3] V. R. S. Nastiti, A. Deastu, and G. I. Marthasari, "Accessibility Analysis of Websites of Provincial Governments in Indonesia," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, Feb. 2022, doi: 10.22219/kinetik.v7i1.1406.

[4] D. B. Yandikaputri, H. Isnaeni, E. Nuraeny, and N. R. Kusuma, "The importance of inclusivity in supporting colour-blindness study case: Gelora Bung Karno," in *IOP Conference Series: Earth and Environmental Science*, IOP Publishing Ltd, Feb. 2021. doi: 10.1088/1755-1315/673/1/012041.

[5] N. K. A. T. Yasa, I. W. M. M. Putra, and M. Y. Andari, "Defek Pengelihatan Warna: Mengenal Perbedaan Buta Warna Kongenital dan Didapat," *Jurnal Kedokteran Unram*, vol. 2021, no. 3, pp. 1021–1027, 2021.

[6] J. R. João and N. M. C. Valentim, "An Exploratory Study about Accessibility, Usability and User Experience with the Visually Impaired using Mobile Applications," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Dec. 2020. doi: 10.1145/3439961.3439998.

[7] M. Agarina and A. Suryadi Karim, "5 th ICITB User-Centered Design Method in the Analysis of User Interface Design of the Department of Informatics System's Website," 2019.

[8] World Wide Web Consortium, "WCAG 2 Overview." Accessed: Jan. 26, 2024. [Online]. Available: https://www.w3.org/WAI/standards-guidelines/wcag/

[9] P. E. Hidayanti, R. I. Handayani, and B. Rifai, "UI/UX Design of Online Tickets for Situ Pasir Maung Tourism in Dago Village Using the Figma Application," *SinkrOn*, vol. 8, no. 4, Oct. 2023, doi: 10.33395/sinkron.v8i2.12098.

[10] A. Z. Mansor, "Managing Student's Grades and Attendance Records using Google Forms and Google Spreadsheets," *Procedia - Social and Behavioral Sciences*, vol. 59, pp. 420–428, Oct. 2012, doi: 10.1016/j.sbspro.2012.09.296.

[11] C. M. Barnum, *Usability Testing Essentials : Ready, Set...Test!* Elsevier, 2011.

[12] F. Febrika *et al.*, "Perancangan UI/UX Fitur Asrama Mahasiswa Berbasis Website dengan Pendekatan User Centered Design," *Jurnal Riset Komputer)*, vol. 10, no. 3, pp. 2407–389, 2023, doi: 10.30865/jurikom.v10i3.6154.

[13] A. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," *Journal of usability studies 4*, vol. no.3, pp. 144–123, 2009.

[14] M. Schrepp, "User Experience Questionnaire Handbook," 2023. Accessed: May 15, 2023. [Online]. Available: www.ueq-online.org

[15] MDN, "Web Accessibility: Understanding Colors and Luminance - Accessibility | MDN," 2024 Accessed: Jan. 26, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Accessibility/Guides/Colors_and_Luminance

Analisis Perbandingan Performa Metode Pengujian *Black Box Equivalence Class Partition* dan *State Transition* pada Aplikasi Visit Techno

Hubertus Rino Augenio[1], Margareta Hardiyanti[1,*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
hubertusrino@mail.ugm.ac.id
*Korespondensi: margareta.hardiyanti@ugm.ac.id;

Abstract - Testing is a crucial stage in software development. It plays a pivotal role in determining the viability of the developed software. Black-box testing is one of the methods used to assess the functionality of software. PT Cipta Sedya Digital operates a digital service line named Techno Center. Techno Center is currently developing a room reservation application called Visit Techno. During the development process of Visit Techno, there are shortcomings in the testing phase as it hasn't been conducted comprehensively across all features. Additionally, the lack of proper identification of suitable testing methods has resulted in suboptimal previous testing. This final project was conducted to provide a solution to Techno Center to enhance the testing process for greater effectiveness. The project utilized an analysis that compared black-box testing methods: Equivalence Class Partition (ECP) and State Transition (ST) in testing the Visit Techno application. Research findings indicate that the Test Case Failed percentage for ST was 13.51% and for ECP was 10.89%. Test Case Execution stood at 100% for ST and 95.05% for ECP. Test Cases Not Executed were at 0% for ST and 4.95% for ECP. The Fault Detection Rate was 10.8/hour for ST and 14.4/hour for ECP. Severity Point for ST was 7, whereas for ECP it was 5. The Average Percentage Fault Detection was 0.72 for ST and 0.67 for ECP. Based on these results, ST proves more effective compared to ECP for use in testing the Visit Techno application. The research outcomes are expected to provide insights to improve the efficiency of black-box testing for the Visit Techno application and similar applications being developed by PT CSDI.

Keywords – Software Testing, Black-box Testing, Equivalence Class Partition, State Transition

Intisari - Pengujian merupakan salah tahap yang penting dalam proses pengembangan perangkat lunak. Pengujian berperan dalam menentukan kelayakan dari perangkat lunak yang dikembangkan. *Black-box testing* merupakan salah satu metode pengujian yang dilakukan untuk mengetahui fungsionalitas perangkat lunak. PT Cipta Sedya Digital mempunyai lini layanan digital dengan nama Techno Center. Techno Center sedang mengembangkan aplikasi peminjaman ruang dengan nama Visit Techno. Pada proses pengembangan Visit Techno, masih terdapat kekurangan pada tahap pengujian karena belum dilakukan secara menyeluruh pada semua fitur. Kurangnya proses identifikasi metode pengujian yang sesuai juga menyebabkan pengujian yang dilakukan sebelumnya tidak berjalan dengan optimal. Proyek akhir ini dilaksanakan untuk memberikan solusi kepada Techno Center dalam meningkatkan proses pengujian agar lebih efektif. Proyek akhir ini menggunakan analisis perbandingan pengujian *black-box Equivalence Class Partition* (ECP) dan *State Transition* (ST) pada pengujian aplikasi Visit Techno. Hasil penelitian yang didapatkan pada perhitungan *Test Case Failed* ST 13,51% dan ECP 10,89%. *Test Case Executed* ST 100% dan ECP 95,05%. *Test Case Not Executed* ST 0% dan ECP 4,95%. *Rate of Fault Detection* ST 10,8/jam dan ECP 14,4/jam. *Severity Point* ST pada level 7 dan ECP pada level 5. *Average Percentage Fault Detection* ST 0,72 dan ECP 0,67. Berdasarkan hasil tersebut ST lebih efektif dibandingkan ECP untuk digunakan dalam pengujian aplikasi Visit Techno. Hasil penelitian diharapkan memberikan wawasan untuk meningkatkan efisiensi pengujian *black-box* aplikasi Visit Techno dan aplikasi sejenis yang sedang dikembangkan PT CSDI.

Kata kunci – Pengujian Perangkat Lunak, *Black-box Testing, Equivalence Class Partition*, *State Transition*

I. PENDAHULUAN

Pengujian perangkat lunak merupakan tahapan yang penting dalam *Software Development Life Cycle* (SDLC) untuk menentukan kelayakan dari sebuah *software* yang dikembangkan. Pengujian bertujuan untuk menunjukan kepada *end-user* bahwa *software* sudah memenuhi persyaratan yang diberikan dan menemukan adanya kesalahan yang tidak diinginkan [1]. Terdapat dua pendekatan dalam melaksanakan proses pengujian perangkat lunak yaitu *white-box* dan *black-box*. Pengujian *white-box* merupakan pengujian fungsional yang berfokus pada struktur internal aplikasi sehingga membutuhkan pengetahuan mengenai kode program, sedangkan *black-box* secara umum menguji aplikasi secara fungsional, tetapi tidak memperhatikan detail kode implementasi [2] . Pada proses pengujian *black-box*, terdapat beberapa tahapan yang dilakukan oleh seorang penguji, salah satu tahapan yang penting adalah pembuatan kasus uji atau *test case*. Pembuatan kasus uji merupakan tahapan yang penting dalam pengujian karena berpengaruh terhadap efektifitas dan efisiensi [3]. Pemilihan metode dalam pengujian *black-box* akan mempengaruhi kasus uji yang dihasilkan sehingga perlu disesuaikan dengan perangkat lunak yang diuji.

PT Cipta Sedya Digital Indonesia (CSDI) memiliki lini layanan berbasis digital dengan nama Techno Center yang melayani pengembangan layanan digital bagi perusahaan pembiayaan di bawah Astra Grup serta perusahaan lain yang memerlukan jasa terkait [4]. Techno Center sedang mengembangkan aplikasi peminjaman ruang bernama Visit Techno. Pada proses pengembangan Visit Techno pengujian dilaksanakan secara manual dan masih terdapat kekurangan. Kekurangan tersebut terdapat pada pengujian fungsional aplikasi Visit Techno yang belum dilakukan secara menyeluruh dan terbatas pada beberapa fitur. Selain itu, kurangnya proses identifikasi metode pengujian yang tepat terhadap fitur aplikasi Visit Techno menyebabkan pengujian yang dilakukan sebelumnya tidak berjalan dengan optimal.

Hal tersebut membuat aplikasi Visit Techno masih terdapat kekurangan dari segi fungsionalitas.

Oleh karena itu, agar pengujian fungsionalitas aplikasi Visit Techno dapat dilakukan dengan optimal, maka dilakukan analisis performa metode *black-box* untuk diimplementasikan pada pengujian aplikasi Visit Techno. Penggunaan *black-box* dalam pengujian dapat meningkatkan kesesuaian sistem dengan desain yang telah diterapkan [5]. Pada studi literatur yang dilakukan oleh [3] bahwa *Equivalence Class Partition* (ECP) dan *State Transition* (ST) merupakan teknik *black-box* yang banyak digunakan dalam pengujian fungsional. ECP memiliki karakteristik berdasarkan domain atau wilayah data *input*, sedangkan ST berbasis pendekatan spesifikasi perangkat lunak. Kedua metode tersebut cocok jika digunakan dalam pengujian fungsional aplikasi Visit Techno yang memiliki banyak elemen *input* dan interaksi komponen menu. Implementasi pengujian *automation* dengan Katalon Studio digunakan pada penelitian ini karena *open source* serta mendukung *record-replay* dan *script mode*. Melalui analisis perbandingan, diharapkan dapat menentukan penggunaan metode yang tepat antara ECP dan ST untuk melakukan pengujian pada aplikasi Visit Techno maupun aplikasi peminjaman sejenis yang dikembangkan PT CSDI.

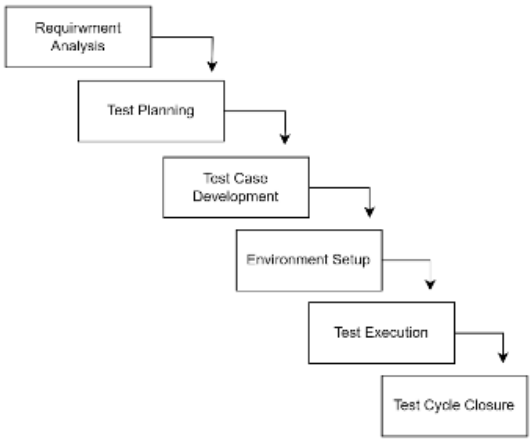
II. DASAR TEORI

A. *Software Testing*

Menurut [6] *software testing* adalah suatu proses atau serangkaian proses yang dirancang untuk memastikan kode komputer melakukan apa yang telah dirancang untuk dilakukan. Perangkat lunak yang baik adalah yang dapat memenuhi hasil yang telah dirancang pada awal tahapan. Penelitian [7] mendefinisikan konsep *testing* sebagai sebuah proses yang perlu untuk direncanakan pada tahap awal *development*. Penelitian [7] membagi *test*ing ke dalam empat tahap yaitu *planning, design, construction, maintenance, dan execution*. Tahap yang dikenalkan oleh Gelperin dan Hetzel disebut sebagai *prevention period* dan banyak diadaptasi untuk digunakan sampai saat ini.

B. *Software Testing Life Cycle*

Software Testing Life Cycle (STLC) adalah siklus hidup atau kerangka kerja yang digunakan dalam pengembangan perangkat lunak untuk merencanakan, merancang, mengelola, dan melaksanakan aktivitas pengujian dengan tujuan memastikan kualitas dan keandalan perangkat lunak. STLC adalah salah satu komponen kunci dalam siklus hidup pengembangan perangkat lunak secara keseluruhan. Setiap STLC memiliki *Entry Criteria*, *Exit Criteria*, *Activities* dan *Deliverable* [8]. Langkah dalam proses STLC seperti pada Gambar 1.



Gambar 1. *Software Testing Life Cycle*

C. *Black Box Testing*

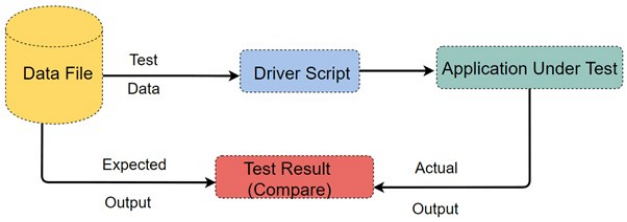
Black-box testing adalah metode pengujian perangkat lunak yang fokus pada pengujian fungsionalitas perangkat lunak tanpa memperhatikan struktur internal kode sumber atau desain perangkat lunak. Pengujian *black-box* juga dikenal dengan isitlah *data driven* atau *input/output-driven testing* [6]. Dalam metode pengujian ini, pengujian dilakukan dari perspektif eksternal, seperti yang akan dilakukan oleh pengguna akhir, dan pengujian dilakukan tanpa pengetahuan tentang bagaimana perangkat lunak diimplementasikan. Teknik yang digunakan dalam pengujian *black-box* berfokus pada domain informasi perangkat lunak, dengan cara membagi domain *input* dan *output* program sehingga memberikan cakupan pengujian yang menyeluruh.

D. *Test Case*

Test case atau uji kasus adalah serangkaian tindakan yang untuk melakukan verifikasi dan validasi pada fitur atau fungsi tertentu. *Test case* merupakan inti dari sebuah pengujian perangkat lunak. *Test case* terdiri dari komponen berupa identifikasi nomor, tujuan*, bisnis role,* deskripsi persyaratan, masukan aktual, keluaran yang diharapkan, kondisi akhir yang diharapkan, dan riwayat eksekusi [9].

E. *Data Driven Testing*

Data Driven Testing (DDT) adalah pendekatan pengujian perangkat lunak yang didasarkan pada penggunaan data eksternal sebagai *input* untuk menjalankan berbagai skenario pengujian [10]. Masukan pengujian dan hasil keluaran yang diharapkan disimpan dalam file data terpisah (biasanya dalam format tabel). Proses *data driven testing* dapat dilihat pada Gambar 2.



Gambar 2. Kerangka Kerja *Data Driven*

F. *Equivalence Class Partition*

Equivalence Class Partition merupakan salah satu teknik dasar dalam pengujian *black-box* yang membagi *domain input* menjadi beberapa kelas atau wilayah [6]. *Domain input* yang telah dibagi ke dalam kelas yang sesuai. Dalam teknik ECP, kelas ekuivalen dibentuk dengan mempertimbangkan kondisi *valid* dan *invalid*. Kondisi *valid* terjadi ketika *input* yang diberikan memberikan informasi yang sesuai. Kondisi *Invalid* terjadi ketika *input* memberikan informasi yang tidak sesuai.

G. *State Transition*

State transition merupakan teknik pengujian *black-box* berbasis *specification* atau model yaitu dengan menggunakan diagram UML untuk menghasilkan *test case*. Prinsip dari teknik ini adalah dengan mendefinisikan keadaan (*state*) dan perubahan (*transition*) dari persyaratan yang sudah ditentukan. *State Transition Diagram* menggambarkan semua keadaan yang dapat dimiliki suatu objek dimana objek tersebut mengubah keadaan (*transition*), melihat kondisi yang harus dipenuhi sebelum transisi terjadi (*guard*), dan aktivitas yang harus dipenuhi selama objek hidup (*action*) [11]. Penelitian [11] menjelaskan pembuatan *state diagram* dari *behavior model diagram* seperti *use case* dan *activity diagram.*

H. *Klasifikasi Bug*

Severity atau tingkat keparahan merupakan konsep untuk menunjukan tingkat keparahan dari kesalahan atau *bug* yang berpengaruh pada fungsionalitas seluruh sistem. Penelitian [12] membagi *severity* ke dalam beberapa level dibawah ini dari tingkat rendah ke tinggi. Kemudian menurut [13] karena tingkat keparahan berbeda antara sistem pelacakan *bug* maka dilakukan pemetaan klasifikasi *bug* berdasarkan *severity level* dengan skala 10 poin yang seragam. Pembagian skala tersebut dapat dilihat pada Tabel 1. Perhitungan *average severity* (1) menggunakan total *severity point* dan jumlah *bug*.

Tabel 1. Klasifikasi *bug* berdasarkan *severity level*

Poin	Tingkat Keparahan	Klasifikasi *Bug*
1	*Enhancement*	
2	*Trivial/Tweak*	*Low*
5	*Minor/Low/Small*	
6	*Normal/Medium*	*Medium*
7	*Major/High*	
9	*Critiacal/Clash*	*High*
10	*Blocker*	

$$Average\ Severity\ ST = \frac{Total\ Severity\ Point}{Jumlah\ Bug} \quad (1)$$

I. *Testing Metric*

Software Testing metric merupakan pengukuran kuantitatif untuk melihat estimasi , produktifitas, dan progres dari pengujian perangkat lunak. Tujuan dari adanya pengukuran melalui *metric* adalah menghasilkan data yang dapat dijadikan acuan perbandingan performa. Penggunaan *testing metric* sangat penting karena dapat meningkatkan efektivitas proses pengujian, berfungsi sebagai indikator tingkat efisiensi dan kebenaran, serta analisis metrik yang ditentukan [2]. Penelitian [14] melakukan pengukuran performa pengujian dengan menghitung *total percentage test case fail* (2), *percentage test case not executed* (3), dan *percentage test case executed* (4).

$$\text{TC } Failed\ (\%) = \left(\frac{Total\ TC\ Failed}{Total\ TC}\right) * 100\% \quad (2)$$

$$\text{TC } Executed\ (\%) = \left(\frac{Total\ TC\ Executed}{Total\ Test\ Case}\right) * 100\% \quad (3)$$

$$\text{TC } Not\ Executed\ (NE)\ (\%) = \left(\frac{Total\ TC\ NE}{Total\ TC}\right) * 100\% \quad (4)$$

J. *Rate of Fault Detection*

Penelitian [15] menjabarkan parameter *Rate of Fault Detection* atau RFT pada (5) untuk mengukur tingkat kesalahan yang berhasil ditemukan. RFT adalah jumlah kesalahan per satuan waktu yang diekspos oleh *test case*.

$$RFT = \left(\frac{Total\ number\ of\ faults}{Total\ time\ execution}\right) \quad (5)$$

K. *Fault Impact*

Fault Impact atau FI pada (6) digunakan untuk menghitung pengaruh kesalahan berdasarkan tingkat keparahan kesalahan tersebut [15]. Pada setiap nilai tingkat *keparahan* ditetapkan berdasarkan dampak yang disebabkan pada produk. FI dihitung dengan membagi jumlah *severity* yang ditemukan pada fitur atau *class* (si) dengan nilai *severity* tertinggi dari fitur atau *class* (Max(s)).

$$FI = \left(\frac{Si}{Max(S)}\right) * 10 \quad (6)$$

L. *Test Case Weight*

Test case weight atau TCW pada (7) digunakan untuk menentukan pentingnya suatu kasus uji dalam pengujian perangkat lunak. Bobot ini dapat digunakan untuk menentukan seberapa penting atau berat sebuah skenario uji dalam rangka untuk mengetahui seberapa baik perangkat lunak dapat beroperasi dalam situasi tersebut [15]. TCW dihitung dengan menjumlahkan *Rate of Fault Detection* dan *Faul Impact* pada masing-masing tes uji pada *class* atau fitur.

$$TCW = RFT + FI \quad (7)$$

M. *Average Percentage Fault Detection*

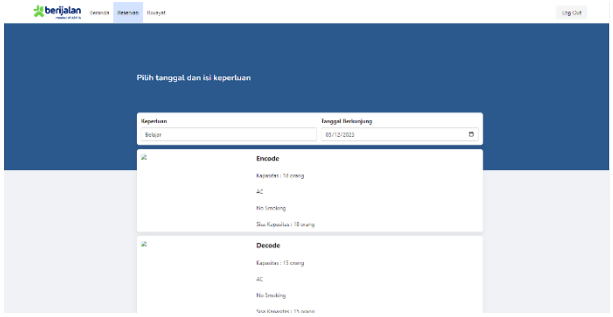
Average Percentege Fault Detection atau APFD merupakan acuan untuk melihat tingkat deteksi kesalahan berdasarkan persentase eksekusi rangkaian pengujian. APFD dihitung dengan mengambil rata-rata dari persentase kesalahan yang terdeteksi selama pelaksanaan rangkaian pengujian [16]. APFD dihitung memasukan data jumlah *test case* yang dijalankan (n), jumlah kesalahan yang berhasil ditemukan (m), dan posisi *test case* dimana terdapat kesalahan atau *fault* Fk (Pos(Fk)) pada (8).

$$APFD = \left(1 - \frac{\sum_{k=1}^{m} Pos(Fk)}{nm} + \frac{1}{2n}\right) \quad (8)$$

III. METODOLOGI

A. Objek Penelitian

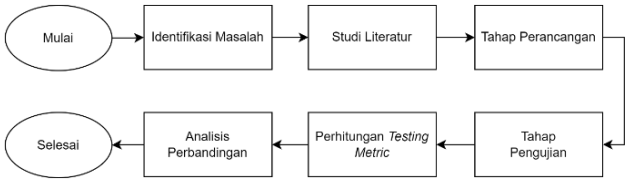
Bahan yang digunakan dalam penelitian ini melibatkan sistem peminjaman ruang berbasis aplikasi web bernama Visit Techno. Visit Techno digunakan sebagai objek yang akan diuji dengan menerapkan teknik ECP dan ST menggunakan pengujian *black-box*. Kemudian untuk data primer yang digunakan dari dokumentasi pengembangan projek aplikasi Visit Techno berupa *Technical Design Spesification* sebagai bahan penunjang dalam proses perancangan *test case*. Tampilan Aplikasi Visit Techno dapat dilihat pada Gambar 3.



Gambar 3.Tampilan aplikasi Visit Techno

B. Tahapan Penelitian

Alur penelitian dibagi ke dalam beberapa tahapan yang dimulai dari identifikasi masalah, studi literatur, tahapan perancangan, tahapan pengujian, perhitungan *testing metric*, dan terakhir analisis perbandingan. Pada tahapan perancangan dan pengujian dilakukan berdasarkan pada alur *Software Testing Life Cycle* (STLC) meliputi *requirement analysis, test planning, test case development, test environment setup*, dan *test case execution*. Tahapan proyek akhir secara garis besar dilakukan berdasarkan pada Gambar 4.

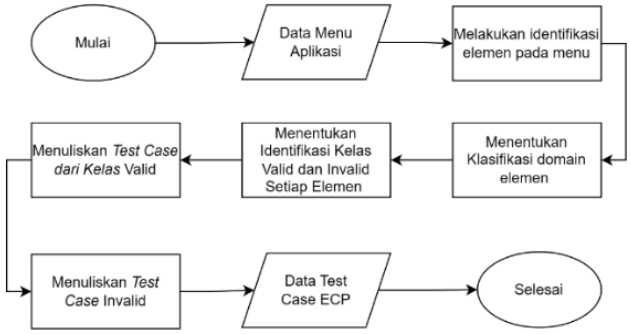


Gambar 4. Tahapan penelitian proyek akhir

C. Perancangan *Test Case*

1. Metode ECP

Pada Gambar 5 merupakan alur perancangan metode ECP. Tahapan dimulai dengan melakukan identifikasi menu, kemudian dilakukan melakukan klasifikasi wilayah domain dari masing-masing elemen seperti input, radio button, dsb. Klasifikasi dilakukan berdasarkan dua wilayah yaitu valid dan invalid. Contoh klasifikasi wilayah pada elemen menu autentikasi user admin pada tipe elemen input dapat dilihat pada Tabel 2. Hasil dari identifikasi wilayah valid dan invalid kemudian akan digunakan menjadi test case dari metode ECP.



Gambar 5. Diagram alir metode ECP

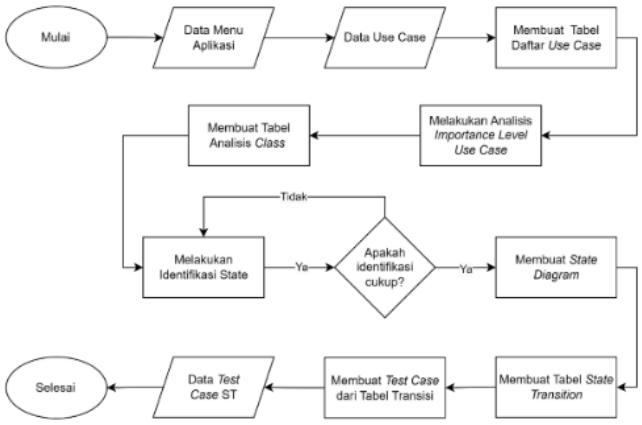
Tabel 2. Identifikasi Wilayah Autentikasi *User Admin*

Elemen	Domain	Valid	Invalid
Username	Panjang karakter	1 - 50	0
	Status akun	Terdaftar	Tidak terdaftar
Password	Panjang karakter	1 - 50	0
	Status akun	Terdaftar	Tidak terdaftar

2. Metode ST

Gambar 6 menunjukkan alur perancangan metode ST yang dimulai dengan analisis *use case* dari *technical document specification*. *Use case* dikelompokkan berdasarkan tingkat kepentingan, dan hasil analisisnya digunakan untuk membentuk *Class*. Identifikasi perilaku dasar seperti manipulasi data dilakukan pada setiap *class* untuk menentukan komponen *state*. Setiap *state* pada *class* digunakan untuk memodelkan perilaku sistem. Setelah semua *state* tercakup, dibuatlah *state transition diagram*. Informasi dari *state transition diagram* diubah menjadi tabel *state*

transition. Contoh tabel *state transition user* admin dapat dilihat pada Tabel 3.



Gambar 6. Diagram alir metode ST

Tabel 3. *State transition class user* admin

State Awal	*Event*	*Guard*	*Action*	State Akhir
S1	Masuk akun	Isian lengkap	Masuk akun	S2
S2	Akun valid	-	Setujui *login*	S3
S3	Masuk sistem	-	Tampilkan *homepage*	S4
S4	Keluar sistem	-	*Logout*	S5
S5	Berhasil keluar	-	Tampilkan *login* menu	S6
S2	Akun tidak valid	-	Tolak *login*	S7

IV. HASIL DAN PEMBAHASAN

Pengujian *automation* pada aplikasi Visit Techno dilakukan dengan menggunakan *tools* Katalon Studio. Pada setiap hasil *test case* yang sudah dibentuk dilakukan implementasikan ke dalam pengujian *automation*. Hasil pengujian yang telah dilakukan dengan *automation testing* menghasilkan pengelompokan *test case* yang dibagi menjadi *pass*, *failed*, dan *not executed*. Kemudian dilakukan perhitungan *metric* untuk mendapatkan hasil perbandingan performa ECP dan ST.

A. Hasil Pengujian

1. ECP

Pengujian yang dilaksanakan menghasilkan 101 *test case*. Total waktu pengujian sebesar 2.750,4 detik atau 45 menit 48 detik (0,73 jam). Hasil rangkuman dapat dilihat pada Tabel 4.

Tabel 4. Hasil rangkuman pengujian ECP

Menu/*Class*	*Pass*	*Fail*	*Not Executed*	Total	*Time (s)*
Registrasi *User* Pengunjung	36	7	3	46	396,9
Masuk Akun Pengunjung	7	0	0	7	185,6
Peminjaman	14	0	1	15	677,8
Masuk Akun Admin	8	0	0	8	165,1
User Management	2	1	0	3	110,4
Room Management	8	2	1	11	442,2
Room Management	10	1	0	11	772,2
Total	**85**	**11**	**5**	**101**	**2750, 4**

2. ST

Pengujian yang dilaksanakan menghasilkan 74 *test case* Total waktu eksekusi pengujian sebesar 3.371,829 detik atau 55 menit 6 detik (0,94 jam). Hasil rangkuman dapat dilihat pada Tabel 5.

Tabel 5. Hasil rangkuman pengujian ST

Menu/*Class*	*Pass*	*Fail*	*Not Executed*	*Total*	*Time (s)*
Ruangan	9	2	0	11	629,6
Peminjaman	15	2	0	17	1021
Riwayat Peminjaman	5	4	0	9	451,3
User Pengunjung	15	0	0	15	604,9
User Admin	7	0	0	7	303,8
User Management	13	2	0	15	361,3
Total	**64**	**10**	**0**	**74**	**3371, 3**

B. Hasil Identifikasi Kesalahan

1. ECP

Bug tersebut didapatkan dari hasil *test case fail*. Identifikasi *bug* pada metode ECP dapat dilihat pada Tabel 6. Hasil perhitungan *average severity* ECP sebesar 4,8 yang termasuk klasifikasi *bug low*.

Tabel 6. Identifikasi *bug* ECP

Menu/Class	Detail *Bug*	*Severity Level*
Registrasi *User* Pengunjung	Menyimpan *input* NIK selain angka (huruf dan simbol)	*Trivial* (2)
	Menyimpan *input* nomor HP lebih dari 13 karakter	*Minor* (5)
	Menyimpan *input* nomor HP dalam huruf dan simbol	*Trivial* (2)
	Menyimpan input nomor HP dengan spasi (setiap dua angka)	*Trivial* (2)
	Menyimpan *input* nomor HP dengan karakter *dash* (-) (setiap 4 angka)	*Trivial* (2)
	Menyimpan *input* nomor HP tanpa diawali dengan '08'	*Minor* (5)
	Tidak sukses menyimpan *input* keahlian lainnya	*Critical* (9)
Pencarian *User*	Menyimpan *input* karakter kolom *search* lebih dari 50 karakter	*Minor* (5)
Tambah Ruangan	Menyimpan kapasitas ruangan dengan nilai < 0	*Major* (8)
	Menyimpan kapasitas ruangan dengan nilai > 20	*Major* (8)
Edit Ruangan	Menyimpan nama ruangan lebih dari 50 karakter	*Minor* (5)
	Total Severity	**53**

$$Average\ Severity = \frac{53}{11} = 4{,}8$$

2. ST

Bug tersebut didapatkan dari hasil *test case fail*. Identifikasi *bug* pada metode ECP dapat dilihat pada Tabel 7. Hasil perhitungan *average severity* ST sebesar 7,3 yang termasuk klasifikasi *bug high*.

Tabel 7. Identifikasi *bug* ST

Menu/Class	Detail *Bug*	*Severity Level*
Ruangan	Sistem menonaktifkan ruangan yang sedang dipesan tanpa ada peringatan	*Medium* (6)
Peminjaman	Sistem dapat memberikan pemesenan kurang dari tanggal di hari *user* pinjam	*Critical* (9)
	Sistem menampilkan ruangan sama yang sudah dipesan oleh *user* lain pada tanggal sama	*Major* (8)
	Sistem memberikan akses pembatalan pada tanggal saat pemesanan berlangsung	*Major* (8)
	Sistem memberikan akses pembatalan pemesanan grup pada tanggal saat pemesanan berlangsung	*Major* (8)
Riwayat Peminjaman	Sistem membatalkan pemesanan grup pada salah satu akun, akun lain yang tercantum tidak terhapus	*Medium* (6)
	Sistem tidak memberikan akses untuk melakukan pemesanan kembali ruangan yang sudah dibatalkan pada pemesanan grup	*Medium* (6)
User Management	Sistem tidak menampilkan foto KTP *user* yang telah diunggah	*Major* (8)
	Sistem tidak menampilkan foto KTP *user lain* yang telah diunggah	*Major* (8)
	Total Severity	**73**

$$Average\ Severity = \frac{73}{10} = 7{,}3$$

C. Hasil *Testing Metric*

Hasil pengujian kemudian dihitung berdasarkan testing metris yang telah ditentukan. *Testing metric* menggambarkan persentase kinerja metode ketika digunakan dalam pengujian aplikasi.

1. Test *Case Failed*

$$TC\ Failed\ ECP\ (\%) = \left(\frac{11}{101}\right) * 100\% = 10{,}89\%$$

$$TC\ Failed\ ST\ (\%) = \left(\frac{10}{72}\right) * 100\% = 13{,}51\%$$

2. *Test Case Executed*

$$TC\ Executed\ ECP\ (\%) = \left(\frac{96}{101}\right) * 100\% = 95\%$$

$$TC\ Executed\ ST\ (\%) = \left(\frac{74}{74}\right) * 100\% = 100\%$$

3. *Test Case Not Executed*

$$TC\ Not\ Executed\ ECP\ (\%) = \left(\frac{5}{101}\right) * 100\% = 4{,}95\%$$

$$TC\ Not\ Executed\ ST\ (\%) = \left(\frac{0}{101}\right) * 100\% = 0\%$$

D. Hasil *Rate of Fault Detection*

$$RFTi\ ECP = \left(\frac{11}{0{,}76}\right) = 14{,}4\ bug/jam$$

$$RFTi\ ST = \left(\frac{10}{0{,}94}\right) = 10{,}6\ bug/jam$$

E. Hasil *Average Percentage Fault Detection*

Pada hasil pengujian dilakukan pengelompokan *bug* dalam *fault matrix*. Kemudian *test case* pada menu/*class* diurutkan berdasarkan *Test Case Weight* (TCW) terbesar sampai terkecil.

1. ECP

Fault matrix pada ECP dapat dilihat pada Tabel 8. Data *fault matrix* digunakan dalam perhitungan APFD. Hasil perhitungan menunjukan pada ECP menghasilkan nilai APFD 0,67.

Tabel 8. *Fault matrix* ECP

Test Case /Faults	T1-T46	T47-T57	T58-T60	T61-T71	T72-T78	T79-T93	T94-T101
F1	T5						
F2	T17						
F3	T18						
F4	T20						
F5	T22						
F6	T24						
F7	T41						
F8		T50					
F9		T52					
F10			T60				
F11				T63			

$$APFD\ ECP = \left(1 - \frac{372}{1111} + \frac{1}{202}\right) = 0{,}67$$

2. ST

Fault matrix pada ST dapat dilihat pada Tabel 9. Data *fault matrix* digunakan dalam perhitungan APFD. Hasil perhitungan menunjukan pada ST menghasilkan nilai APFD 0,72.

Tabel 9. *Fault matrix* ST

Test Case /Faults	T1-T9	T10-T26	T27-T41	T42-T52	T53-T67	T68-T74
F1	T3					
F2	T7					
F3	T8					
F4	TC9					
F5		T13				
F6		T15				
F7			T30			
F8			T34			
F9				T46		
F10				T50		

$$APFD\ ST = \left(1 - \frac{215}{740} + \frac{1}{148}\right) = 0{,}72$$

F. Analisis Perbandingan Hasil

Pada hasil pengujian aplikasi Visit Techno dengan metode ECP dan ST diperoleh masing-masing hasil dari perhitungan *metric*. Perbandingan hasil dapat dilihat pada Tabel 10.

Tabel 10. Perbandingan hasil ECP dan ST

Hasil Pengukuran	**ECP**	**ST**
Percentage Test Case Failed	10,89%	13,51%
Percentage Test Case Executed	95,05%	100%
Percentage Test Case Not Executed	4,95%	0%
Rate of Fault Detection	14,4	10,8
Average Severity Point	4,8	7,3
Average Percantege Fault Detection	0,67	0,72

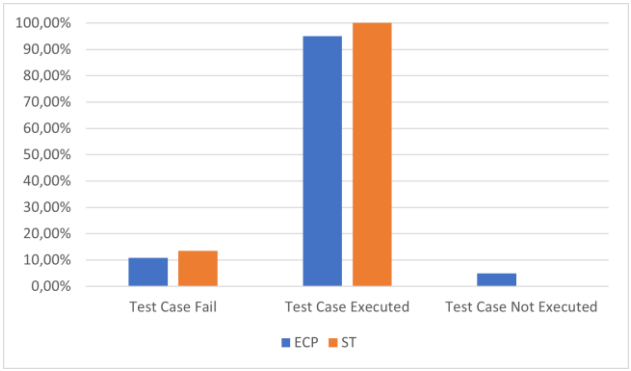
Hasil dari perbandingan diperoleh bahwa pada perhitungan *Percentage Test Case Failed* ST lebih banyak mengungkap kesalahan pada aplikasi dengan persentase 13,51% berbanding 10,89%. Pada perhitungan *Percentage Test Case Executed* ST lebih banyak menghasilkan *test* case yang berhasil dieksekusi dengan 100% kasus uji yang dijalankan dibandingkan dengan ECP dengan persentase 95,05%.

Kemudian perhitungan *Percentage Test Case Not Executed* metode ST unggul karena persentase yang dihasilkan lebih kecil dengan nilai 0% tanpa ada *test* case yang tidak dijalankan, dibandingkan dengan metode ECP yang menyisakan 4,95% *test* case yang tidak dijalankan. Pada perhitungan *Rate of Fault Detection* ECP lebih unggul karena menghasilkan *rate* nilai 14 *bug* yang berhasil dideteksi dalam satu jam dibandingkan dengan ST yang mendeteksi 11 *bug* dalam waktu satu jam. Pada perhitungan tingkat keparahan atau *severity point* ECP mendapatkan skor 4,8 yang termasuk ke dalam kategori rendah , sedangkan ST mendapatkan skor 7,3 yang termasuk ke dalam kategori tinggi. Perhitungan yang

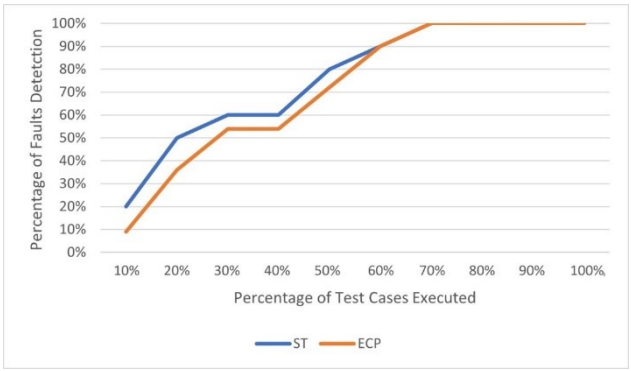
terakhir yaitu APFD dimana metode ST lebih unggul sebesar 0,72 daripada metode ECP sebesar 0,67.

G. Grafik Perbandingan Hasil

Grafik persentase hasil pengujian pada Gambar 7 dibuat untuk menunjukan perbandingan performa pengujian Visit Techno dengan metode ECP dan ST. Kemudian berdasarkan hasil perhitungan APFD didapatkan perbandingan grafik persentase APFD ECP dan ST yang dapat dilihat pada Gambar 8. Pada awal grafik, ST mendeteksi 20% kesalahan dari ECP yang sebesar 10% saja pada eksekusi *test case* sebanyak 10%. Pada grafik tersebut ST selalu unggul dari ECP sampai persentase *fault detection* mencapai 100%. Hal tersebut sesuai dengan perbandingan perhitungan APFD ST lebih unggul dari ECP.



Gambar 7. Grafik perbandingan persentase performa ECP dan ST



Gambar 8. Grafik perbandingan APFD ECP dan ST

V. SIMPULAN

ECP yang berfokus pada elemen pengujian *input* menghasilkan kasus uji sebanyak 101 buah, sedangkan ST yang berfokus pada perilaku pengguna terhadap respon sistem menghasilkan kasus uji 74 buah. Dalam menentukan metode yg efektif dari ST dan ECP, dapat dilihat dari hasil perhitungan *percentage test case fail, test case executed*, dan *average percentage of fault detection*. Hasil perhitungan tersebut menunjukan dengan metode ST menghasilkan angka yang tinggi daripada ECP sehingga metode ST lebih unggul.

Pada perhitungan persentase *not executed* dengan metode ST menghasilkan angka yang kecil daripada ECP sehingga metode ST lebih unggul. Berdasarkan hal tersebut metode ST lebih efektif dibandingkan dengan ECP untuk digunakan dalam pengujian aplikasi Visit Techno. Kemudian pada hasil identifikasi ECP berhasil menemukan *bug* dalam klasifikasi *low* pada komponen input sedangkan ST berhasil menemukan *bug* dalam klasifikasi *high* berupa respon sistem terhadap aksi atau perintah yang diberikan oleh *user*. Berdasarkan hasil tersebut, *aplikasi* Visit Techno perlu melakukan perbaikan kembali karena masih ditemukan *bug* dalam kategori tinggi maupun rendah.

VI. UCAPAN TERIMA KASIH

Terima kasih kepada PT Cipta Sedya Digital Indonesia (CSDI) atas fasilitas dan dukungan dalam penelitian ini. Terakhir terima kasih kepada kolega, teman-teman, dan individu lain yang turut berkontribusi dalam menyempurnakan penulisan jurnal ini.

REFERENSI

[1] I. Sommerville, Software engineering. Pearson, 2011.

[2] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, "Software testing techniques: A literature review," in *Proceedings - 6th International Conference on Information and Communication Technology for the Muslim World, ICT4M 2016*, Institute of Electrical and Electronics Engineers Inc., Jan. 2017, pp. 177–182. doi: 10.1109/ICT4M.2016.40.

[3] N. Setiani, R. Ferdiana, P. I. Santosa, and R. Hartanto, "Literature review on test case generation approach," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jan. 2019, pp. 91–95. doi: 10.1145/3305160.3305186.

[4] F. Ardi and H. P. Putro, "Pengujian Black Box Aplikasi Mobile Menggunakan Katalon Studio (Studi Kasus: ACC Partner PT. Astra Sedaya Finance)," *AUTOMATA*, vol. 2, no. 1, 2021.

[5] Y. F. Achmad and A. Yulfitri, "PENGUJIAN SISTEM PENDUKUNG KEPUTUSAN MENGGUNAKAN BLACK BOX TESTING STUDI KASUS E-WISUDAWAN DI INSTITUT SAINS DAN TEKNOLOGI AL-KAMAL," 2020.

[6] G. J. Myers, Badgett Tom, and Sandler Corey, *The Art of Software Testing Third Edition*, Third Edition. Hoboken, New Jersey: John Wiley & Sons, 2012.

[7] D. Gelperin and B. Hetzel, "The Growth of Software Testing," *Commun ACM*, vol. 31, pp. 687–695, Sep. 1988, doi: 10.1145/62959.62965.

[8] Auliyaa Tri Nur, "Software Testing Life Cycle." Accessed: Sep. 18, 2023. [Online]. Available: https://sis.binus.ac.id/2020/07/06/software-testing-life-cycle/

[9] P. C. Jorgensen, *Software Testing Fourth Edition A Craftsman's Approach*, 4th edition. Boca Raton, Fla: Auerbach Publications, 2013.

[10] A. Lu, W. Fang, C. Xu, S. C. Cheung, and Y. Liu, "Data-driven testing methodology for RFID systems," *Front Comput Sci China*, vol. 4, no. 3, pp. 354–364, 2010, doi: 10.1007/s11704-010-0387-6.

[11] R. S. Presman, *Software Engineering: A Practitioner's Approach*, Seventh Edition. New York: McGraw-Hill COmpanies, 2010. [Online]. Available: www.mhhe.com/pressman.

[12] B. Kucuk and E. Tuzun, "Characterizing Duplicate Bugs: An Empirical Analysis," in *Proceedings - 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2021*, Institute of Electrical and Electronics Engineers Inc., Mar. 2021, pp. 661–668. doi: 10.1109/SANER50967.2021.00084.

[13] B. Zhou, I. Neamtiu, and R. Gupta, "Experience report: How do

bug characteristics differ across severity classes: A multi-platform study," in *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*, 2015, pp. 507–517. doi: 10.1109/ISSRE.2015.7381843.

[14] I. G. S. Aryandana, A. E. Permanasari, and T. B. Adji, "Comparing method equivalence class partitioning and boundary value analysis with study case add medicine module," in *IOP Conference Series: Materials Science and Engineering*, Institute of Physics Publishing, Jan. 2020. doi: 10.1088/1757-899X/732/1/012072.

[15] K. Gopinath and D. Sureshkumar, "Test Case Prioritization for Regression Testing based on Severity of Fault," *International Journal on Computer Science and Engineering*, vol. 2, Aug. 2010.

[16] R. Pradeepa and K. Vimala Devi, "Effectiveness of Testcase Prioritization using APFD Metric: Survey," *Int J Comput Appl*, pp. 975–8887, 2013.

Peramalan Harga Saham Perbankan dengan Menggunakan Metode *Moving Average* dan *Exponential Smoothing*

Ach Sadili[1], Anis Zubair[1,*]
[1]Sistem Informasi, Universitas Merdeka Malang;
achsadili8625@gmail.com
*Korespondensi: anis.zubair@unmer.ac.id;

Abstract – Forecasting banking stock prices is crucial for investors to minimize the risk of capital loss and make better investment decisions. But lack of knowledge and fear of stock price fluctuations often hinder potential investors. This study uses secondary data stock prices BBRI, BRIS, BBCA, BMRI, BBNI, BNIS, and BBTN of www.investing.com the period from January 2 to September 23, 2023, which was analyzed by Moving Average (MA) and Exponential Smoothing (ES) methods with adaptive parameters to forecast stock prices, and evaluated for accuracy using MAE, MSE, and MAPE metrics. The results showed that both models had low MAPE (0.8215%-5.9682%), with ES being superior on BBCA (MAPE 0.8215%) and BBRI (MAPE 1.00816%), but high MASE (23.06761–104.1996) indicated both models were less effective than naive methods, and negative MSE anomalies on ES for BBNI, BNIS, and BBTN indicated data problems. Overall, ES is more precise than MA, but both require parameter adjustments and data verification.

Keywords – Stock Forecasting, Prediction Accuracy, Moving Average, Exponential Smoothing

Intisari – Peramalan harga saham perbankan menjadi krusial bagi investor untuk meminimalkan risiko capital loss dan membuat keputusan investasi yang lebih baik. Namun kurangnya pengetahuan dan ketakutan terhadap fluktuasi harga saham sering menghambat calon investor. Penelitian ini menggunakan data sekunder harga saham BBRI, BRIS, BBCA, BMRI, BBNI, BNIS, dan BBTN dari www.investing.com periode 2 Januari hingga 23 September 2023, yang dianalisis dengan metode *Moving Average* (MA) dan *Exponential Smoothing* (ES) dengan parameter adaptif untuk meramalkan harga saham, serta dievaluasi akurasinya menggunakan metrik MAE, MSE, dan MAPE. Hasil menunjukkan bahwa kedua model memiliki MAPE rendah (0,8215%–5,9682%), dengan ES lebih unggul pada saham BBCA (MAPE 0,8215%) dan BBRI (MAPE 1,00816%), namun MASE tinggi (23,06761–104,1996) menunjukkan kedua model kurang efektif dibandingkan metode naif, dan anomali MSE negatif pada ES untuk BBNI, BNIS, dan BBTN mengindikasikan masalah data. Secara keseluruhan, ES lebih presisi dibandingkan MA, tetapi keduanya memerlukan penyesuaian parameter dan verifikasi data.

Kata kunci – Peramalan Saham, Akurasi Prediksi, *Moving Average*, *Exponential Smoothing*

I. PENDAHULUAN

Bank adalah lembaga keuangan yang sangat penting untuk mengumpulkan dana dari masyarakatt, memberikan pinjaman, dan menyediakan berbagai layanan keuangan. Fungsi utama bank adalah sebagai perantara antara orang yang membutuhkan dana (peminjam atau kreditur) dan orang yang memiliki dana (nasabah atau deposito). Perbankan adalah segala sesuatu yang menyangkut tentang bank, mencakup kelembagaan, kegiatan usaha serta cara dan proses dalam melaksanakan kegiatan usahanya [1].

Investasi merupakan memanfaatkan dana saat ini untuk mendapatkan keuntungan di masa mendatang. Investasi mempunyai beberapa jenis, yaitu investasi emas, properti, deposito, obligasi, reksadana, dan saham [2].

Penelitian ini berfokus pada investasi saham. Saham adalah bukti bahwa seseorang atau pihak, juga dikenal sebagai badan usaha, berpartisipasi dalam suatu perusahaan atau perseroan terbatas. Investasi saham adalah proses dimana para investor membeli kepemikan dalam suatu perusahaan atau entitas bisnis dengan harapan mendapatkan keuntungan dari pertumbuhan nilai saham atau dari pembayaran dividen. Keuntungan pada investasi saham ada dua yaitu *capital gain* dan dividen [3].

Setiap investor berharap untuk mendapatkan keuntungan jika harga saham berubah dari waktu ke waktu. Setiap bidang investasi memiliki resiko, begitu juga saat membeli saham. Salah satu risiko yang dihadapi saat membeli saham adalah resiko turunnya harga, yang juga dikenal sebagai *capital loss,* dan resiko terjadinya likuidasi terhadap perusahaan yang mengeluarkan saham. Meskipun saham dapat menghasilkan keuntungan besar, para investor juga mempunyai kemungkinan kehilangan sebagian atau seluruh investasi mereka jika harga saham turun. Tingkat risiko biasanya dikaitkan dengan pengembalian yang diharapkan, saham dengan risiko yang lebih tinggi memiliki potensi pengembalian yang lebih tinggi.

Kebanyakan para calon nasabah dan investor masih takut untuk investasi saham dikarenakan risiko-risiko yang ada dan kurangnya pengetahuan mereka tentang investasi saham. Calon nasabah dan investor juga masih takut tertipu terhadap pialang-pialang saham karena mereka belum mengetahui naik turunnya harga saham, sehingga apabila salah melangkah dalam melakukan transaksi investasi akan mengakibatkan kerugian bagi para calon nasabah dan investor.

Peramalan harga saham perbankan membantu investor untuk membuat keputusan investasi yang lebih baik. Dengan memahami pola tren potensial harga saham, investor dapat membuat keputusan pembelian atau penjualan berdasarkan informasi dari peramalan. Penelitian ini menggunakan data

harga saham pada *website* www.investing.com mulai 2 Januari sampai 23 September 2023. Penelitian ini menggunakan metode *Moving Average* dan *Exponential Smoothing* (*Adaptive Parameter*) untuk peramalannya. Dengan demikian, penelitian ini akan bertujuan untuk meramalkan harga saham dan membandingkan kedua metodenya.

II. DASAR TEORI

Pada penelitian ini metode yang diperlukan adalah *forecasting* atau yang biasa disebut dengan peramalan. Peramalan merupakan salah satu cara yang efektif dan efisien untuk melakukan perencanaan. Dari beberapa metode pada forecasting ada metode yang disebut metode *Moving Average* dan metode *Exponential Smoothing*. Metode *Moving Average* merupakan metode yang paling umum digunakan dalam peramalan harga saham, yang didasarkan pada rata-rata bergerak dari data historis. Sedangkan, metode *Exponential Smoothing* adalah metode peramalan yang berfokus pada perubahan terkini dalam data historis. Metode ini memberikan bobot lebih besar pada data terkini, karena itu, metode ini cocok untuk situasi di pola permintaan cenderung berubah-ubah.

A. Metode *Moving Average*

Menurut [4], [5], [6] salah satu metode peramalan bisnis yang sederhana adalah *Moving Average*. *Moving Average* menggunakan kumpulan data historis atau data masa lalu, untuk memperkirakan kondisi masa depan. Rumus *Moving Average* dinyatakan pada (1).

$$MA = \frac{A_1 + \cdots + A_n}{n} \tag{1}$$

B. Metode *Exponential Smoothing*

Menurut [7], [8], [9] *Exponential Smoothing* sederhana adalah teknik untuk menyelaraskan data deret waktu dengan menggunakan fungsi eksponensial untuk memberikan bobot yang kurang secara eksponensial seiring berjalannya waktu. Metode penghitungan ini menggunakan rata-rata tertimbang, yang bobotnya berkurang secara eksponensial seiring dengan jarak pengamatan sebelumnya. Pengamatan paling lama terkait dengan bobot terkecil. Dikarenakan pada penelitian ini menggunakan *Adaptive Parameter*, maka alpha yang digunakan ditentukan oleh beta pada penerapannya. Rumus *Exponential Smoothing* dinyatakan pada (2).

$$F_{t+1} = a * F_t + (1 - a) * F_{t-1} \tag{2}$$

Terdapat beberapa cara untuk mengukur keakuratan dari kedua metode *Moving Average* dan *Exponential Smoothing*, seperti MAE, MSE, dan MAPE [10], [11], [12]. Beberapa di antaranya dinyatakan pada (3), (4) dan (5).

- *MAE (Mean Absolute Error)*

$$MAE = \frac{\sum e_i}{n} \tag{3}$$

- *MSE (Mean Square Error)*

$$MSE = \frac{\sum e_i^2}{n} \tag{4}$$

- *MAPE (Mean Absolute Percentage Error)*

$$MAPE = \sum_{i=1}^{n} \frac{|PE_i|}{n} \tag{5}$$

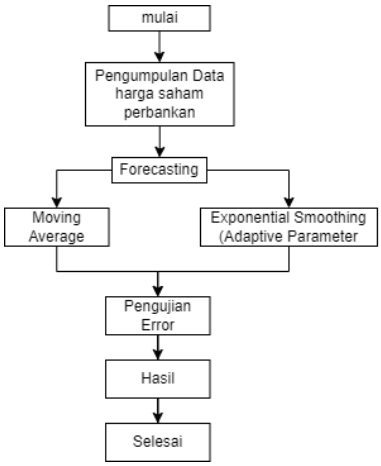
C. Microsoft Excel

Menurut [13], [14] Microsoft Excel termasuk dalam paket Microsoft Office merupakan perangkat lunak *spreadsheet* yang paling populer di dunia. Program Microsoft Excel ini dapat mengatur, menghitung, dan menganalisis data dalam bentuk tabel atau lembar kerja (*worksheet*) yang terdiri dari baris dan kolom. Pengguna Excel memungkinkan untuk melakukan perhitungan yang sistemastis, membuat grafik, dan menjalankan tugas yang terkait dengan data.

III. METODOLOGI

A. Desain Penelitian

Alur desain penelitian dalam melakukan penelitian ini dapat dilihat pada Gambar 1.



Gambar 1. Desain penelitian

Pertama, pengumpulan data sekunder harga saham perbankan bertujuan untuk mendapatkan data harga saham perbankan pada tanggal 2 Januari sampai 22 September tahun 2023. Pada penelitian ini pengumpulan data dilakukan dengan cara langsung mengambil data pada *website* www.investing.com. Selanjutnya untuk data saham perbankan disalin ke dalam aplikasi Excel.

Kedua, *forecasting* dengan metode *Moving Average* dan *Exponential Smoothing* (*Adaptive Parameter*) bertujuan untuk mengetahui hasil peramalan dari kedua metode tersebut pada harga saham. Cara untuk melakukan peramalan harga saham perusahaan perbankan dalam penelitian ini yaitu data historis pada harga saham perbankan yang sudah dicopy selanjutnya dipindahkan ke Excel kemudian langsung memasukkan rumus-rumus pada kedua metode tersebut.

Ketiga, pengujian eror bertujuan untuk mengetahui akurasi peramalan yang dilakukan. Rumus-rumus pengujian eror di antaranya adalah MAE, MSE, dan MAPE. Terakhir, hasil peramalan bertujuan untuk mengetahui perbedaan hasil dari peramalan kedua metode yaitu *Moving Average* dan *Exponential Smoothing* (*Adaptive Parameter*).

B. Lokasi Penelitian

Penelitian dilakukan di laboratorium pribadi peneliti atau di fasilitas laboratorium komputer kampus, dengan pemrosesan data menggunakan perangkat lunak Excel. Data yang digunakan merupakan data sekunder dari *website* https://id.investing.com/. Data diambil mulai 2 Januari sampai 22 September 2023.

C. Teknik Pengumpulan Data

Data diperoleh dengan menggunakan metode dokumentasi, yaitu mengunduh dataset dari *website* https://id.investing.com/ yang diambil mulai 2 Januari sampai 22 September 2023.

D. Teknik Analisis Data

Analisis data pada penelitian ini menggunakan analisis data deskriptif kuantitatif. Teknik analisis deskriptif kuantitatif adalah teknik yang digunakan untuk pengujian, pengukuran, dan hipotesis berdasarkan perhitungan matematika dan statistik. Analisis statistik yang digunakan untuk menganalisis data dengan menggambarkan atau menjelaskan data yang dikumpulkan seperti apa adanya [15].

IV. HASIL DAN PEMBAHASAN

A. Pembahasan

Data harga saham perbankan ini akan dilakukan proses peramalan atau *forecasting* menggunakan metode *Moving Average* dan *Exponential Smoothing* (*Adaptive Parameter*). Pada proses peramalan tersebut akan diperlukan beberapa tahapan berupa proses *forecasting*, peramalan periode selanjutnya, dan perhitungan eror.

B. Pemilihan Metode Terbaik

Peramalan harga saham merupakan alat penting bagi investor untuk membuat keputusan investasi yang lebih terinformasi dan meminimalkan risiko kerugian. Dalam konteks ini, metode *Moving Average* (MA) dan *Exponential Smoothing* (ES) dengan parameter adaptif sering digunakan untuk memprediksi harga saham berdasarkan data historis. Bagian berikut akan membahas kinerja kedua metode tersebut dalam meramalkan harga saham perbankan, membandingkan akurasi, kelemahan, serta faktor-faktor yang memengaruhi efektivitasnya, untuk menentukan metode terbaik yang dapat diandalkan oleh investor.

1. Pemilihan Metode Terbaik *BBRI*

Dalam analisis peramalan saham BBRI, ada dua metode yang digunakan. Masing-masing adalah *Moving Average* (MA) dan *Exponential Smoothing* (ES) dengan parameter adaptif. Tabel 1 menyajikan hasil evaluasi kinerja model MA. Tabel 2 menunjukkan hasil dari metode ES. Beberapa metrik evaluasi yang digunakan mencakup ME (*Mean Error*), MAE (*Mean Absolute Error*), MSE (*Mean Squared Error*), dan MAPE (*Mean Absolute Percentage Error*). Perbandingan hasil ini memberikan gambaran mengenai tingkat akurasi dan kesalahan masing-masing metode dalam meramalkan pergerakan harga saham BBRI.

a. *Moving Average*

Tabel 1. MA BBRI

Metrik Evaluasi	*Moving Average*
ME	4
MAE	59,77
MSE	6018,84
MAPE	1,1708%

Tabel 1 menunjukkan hasil evaluasi model *Moving Average* (MA) dalam meramalkan harga saham BBRI. Nilai ME (*Mean Error*) sebesar 4 mengindikasikan bahwa model cenderung sedikit *overestimate*. MAE (*Mean Absolute Error*) sebesar 59,77 menunjukkan rata-rata besar kesalahan absolut dalam satuan harga saham. MSE (*Mean Squared Error*) yang tinggi, yaitu 6018,84, mengindikasikan adanya beberapa kesalahan prediksi yang cukup besar. Sementara itu, MAPE (*Mean Absolute Percentage Error*) sebesar 1,1708% menandakan bahwa tingkat kesalahan persentase rata-rata masih tergolong rendah, yang berarti model MA cukup akurat dalam konteks ini.

b. *Exponential Smoothing* (*Adaptive Parameter*)

Tabel 2. ES BBRI

Metrik Evaluasi	*Exponential Smoothing* (*Adaptive Parameter*)
MASE	50,33875
ME	1,901408
MSE	1,563237
MAPE	1,00816%

Tabel 2 menyajikan hasil evaluasi metode *Exponential Smoothing* (dengan parameter adaptif) dalam memprediksi harga saham BBRI. Nilai ME (*Mean Error*) sebesar 1,901408 menunjukkan bahwa model memiliki bias sangat kecil dalam prediksi. MSE (*Mean Squared Error*) sebesar 1,563237 menunjukkan kesalahan kuadrat rata-rata yang

sangat kecil, menandakan bahwa model cukup presisi. MAPE (*Mean Absolute Percentage Error*) sebesar 1,00816% mengindikasikan bahwa model memiliki tingkat kesalahan persentase rata-rata yang rendah, lebih baik dibandingkan metode *Moving Average*. MASE (*Mean Absolute Scaled Error*) sebesar 50,34, meskipun tinggi, bisa mencerminkan sensitivitas terhadap skala data, sehingga interpretasinya perlu dilihat dalam konteks pembandingnya.

2. Pemilihan Metode Terbaik BBRI Syariah

Untuk mengevaluasi akurasi model peramalan harga saham BRIS, digunakan dua metode. Masing-masing adalah *Moving Average* (MA) dan *Exponential Smoothing* (ES) dengan parameter adaptif. Tabel 3 menyajikan hasil evaluasi model MA, sementara Tabel 4 menampilkan kinerja metode ES. Metrik evaluasi yang digunakan adalah ME, MAE, MSE, dan MAPE. Metrik evaluasi ini memberikan gambaran sejauh mana tiap-tiap model mampu meminimalkan kesalahan prediksi. Hasil perbandingan kedua metode ini dapat menentukan pendekatan mana yang lebih efektif dalam meramalkan pergerakan harga saham BRIS secara akurat dan efisien.

a. *Moving Average*

Tabel 3. MA BRIS

Metrik Evaluasi	*Moving Average*
ME	2
MAE	29,34
MSE	1938,57
MAPE	1,8201%

Tabel 3 menunjukkan kinerja model *Moving Average* (MA) dalam meramalkan harga saham BRIS. Nilai ME (*Mean Error*) sebesar 2 menunjukkan bahwa model cenderung sedikit *overestimate*, tetapi masih dalam batas yang wajar. MAE (*Mean Absolute Error*) sebesar 29,34 menandakan rata-rata kesalahan absolut dalam prediksi harga. MSE (*Mean Squared Error*) sebesar 193,57 cukup tinggi, menunjukkan adanya beberapa prediksi yang menyimpang cukup jauh dari nilai aktual. Sementara itu, MAPE (*Mean Absolute Percentage Error*) sebesar 1,8201% menunjukkan bahwa model MA memiliki tingkat akurasi yang cukup baik dalam konteks prediksi persentase.

b. *Exponential Smoothing* (*Adaptive Parameter*)

Tabel 4. ES BRIS

Metrik Evaluasi	*Exponential Smoothing* (*Adaptive Parameter*)
MASE	25,81101
ME	1,549296
MSE	1,354628
MAPE	1,6478%

Tabel 4 menyajikan hasil evaluasi metode *Exponential Smoothing* (dengan parameter adaptif) untuk peramalan harga saham BRIS. Nilai ME (*Mean Error*) sebesar 1,549296 menunjukkan bahwa model memiliki bias yang sangat kecil dalam prediksi. MSE (*Mean Squared Error*) sebesar 1,354628 menandakan kesalahan prediksi kuadrat rata-rata yang sangat rendah, mengindikasikan presisi yang baik. MAPE (*Mean Absolute Percentage Error*) sebesar 1,6478% mengindikasikan bahwa model cukup akurat dalam prediksi relatif terhadap nilai aktual. Sementara itu, nilai MASE (Mean Absolute Scaled Error) sebesar 25,81101 memberikan informasi tambahan mengenai skala kesalahan absolut terhadap benchmark model.

3. Pemilihan Metode Terbaik BBCA

Dalam upaya meramalkan pergerakan harga saham BBCA, digunakan dua metode peramalan yang umum. Masing-masing adalah *Moving Average* (MA) dan *Exponential Smoothing* (ES) dengan parameter adaptif. Tabel 5 menyajikan kinerja metode MA. Tabel 6 menampilkan hasil evaluasi metode ES. Metrik evaluasi seperti ME, MAE, MSE, dan MAPE digunakan untuk menilai tingkat kesalahan dan akurasi prediksi tiap-tiap model. Melalui perbandingan ini, dapat dilihat sejauh mana kedua pendekatan ini efektif dalam memberikan hasil peramalan yang andal untuk saham BBCA.

a. *Moving Average*

Tabel 5. MA BBCA

Metrik Evaluasi	*Moving Average*
ME	4
MAE	86,81
MSE	12422,99
MAPE	0, 9819%

Tabel 5 menampilkan hasil evaluasi model *Moving Average* (MA) untuk peramalan harga saham BBCA. Nilai ME (*Mean Error*) sebesar 4 menunjukkan adanya kecenderungan sedikit *overestimate* dari model. MAE (*Mean Absolute Error*) sebesar 86,81 menunjukkan bahwa rata-rata kesalahan absolut dalam prediksi cukup besar dalam satuan harga. MSE (*Mean Squared Error*) sebesar 12.422,99 menunjukkan adanya kesalahan kuadrat yang signifikan, kemungkinan karena beberapa prediksi menyimpang jauh. Namun, MAPE (*Mean Absolute Percentage Error*) yang hanya sebesar 0,9819% menunjukkan bahwa secara persentase, kesalahan prediksi relatif rendah, sehingga model tetap dapat dianggap cukup akurat dari perspektif proporsional.

b. *Exponential Smoothing* (*Adaptive Parameter*)

Tabel 6. ES BBCA

Metrik Evaluasi	*Exponential Smoothing* (*Adaptive Parameter*)
MASE	70,57284
ME	1,525822
MSE	1,705113
MAPE	0,8215%

Tabel 6 menampilkan hasil evaluasi model *Exponential Smoothing* (ES) dengan parameter adaptif untuk saham BBCA. Model *Exponential Smoothing* untuk saham BBCA memiliki akurasi persentase yang sangat baik berdasarkan MAPE (0,8215%) dan MSE yang relatif kecil (1,705113). Namun, nilai MASE yang tinggi (70,57284) menunjukkan bahwa model ini kurang efektif dibandingkan metode sederhana, dan adanya bias positif kecil (ME 1,525822) perlu diperhatikan. Penyesuaian parameter atau pendekatan lain mungkin diperlukan untuk meningkatkan kinerja prediksi secara keseluruhan.

4. Pemilihan Metode Terbaik BMRI

Untuk menganalisis akurasi model peramalan harga saham BMRI, digunakan dua metode statistik yang umum. Masing-masing adalah *Moving Average* (MA) dan *Exponential Smoothing* (ES) dengan parameter adaptif. Tabel 7 memperlihatkan kinerja model MA, sementara Tabel 8 menunjukkan hasil dari model ES. Evaluasi dilakukan menggunakan metrik seperti ME, MAE, MSE, dan MAPE, yang memberikan gambaran seberapa besar kesalahan prediksi masing-masing metode. Perbandingan ini membantu menentukan pendekatan yang lebih tepat dan efisien dalam meramalkan pergerakan harga saham BMRI.

a. *Moving Average*

Tabel 7. MA BMRI

Metrik Evaluasi	*Moving Average*
ME	8
MAE	67,92
MSE	8253,32
MAPE	1,2676%

Tabel 7 menyajikan hasil evaluasi kinerja metode *Moving Average* (MA) untuk peramalan data BMRI. Berdasarkan metrik yang ditampilkan, metode ini memiliki *Mean Error* (ME) sebesar 8, yang menunjukkan adanya sedikit bias positif dalam peramalan. *Mean Absolute Error* (MAE) sebesar 67,92 mencerminkan rata-rata kesalahan absolut yang relatif moderat. Sementara itu, *Mean Squared Error* (MSE) yang tinggi, yaitu 8253,32, mengindikasikan adanya variasi kesalahan yang cukup besar, kemungkinan dipengaruhi oleh beberapa prediksi yang meleset jauh. Namun, *Mean Absolute Percentage Error* (MAPE) sebesar 1,2676% menunjukkan tingkat kesalahan relatif yang sangat rendah, menandakan bahwa metode MA cukup akurat secara persentase untuk data BMRI. Secara keseluruhan, metode MA menunjukkan kinerja yang baik dalam hal akurasi relatif, meskipun perlu perhatian pada variasi kesalahan yang besar.

b. *Exponential Smoothing* (*Adaptive Parameter*)

Tabel 8. ES BMRI

Metrik Evaluasi	*Exponential Smoothing* (*Adaptive Parameter*)
MASE	59,9467
ME	4,39906
MSE	4,36326
MAPE	1,1374%

Tabel 8 menampilkan hasil evaluasi model *Exponential Smoothing* (ES) dengan parameter adaptif untuk saham BMRI. Model *Exponential Smoothing* dengan parameter adaptif untuk saham BMRI memiliki akurasi yang baik berdasarkan MAPE yang rendah (1,1374%). Namun, nilai MASE yang tinggi (59,9467) dan adanya bias positif (ME 4,39906) menunjukkan bahwa model ini belum optimal dan mungkin memerlukan penyesuaian parameter atau pendekatan lain untuk meningkatkan kinerja prediksi, terutama dalam mengurangi kesalahan absolut dan bias.

5. Pemilihan Metode Terbaik BBNI

Dalam peramalan harga saham BBNI, digunakan dua metode yang sering diaplikasikan dalam analisis time series. Masing-masing adalah *Moving Average* (MA) dan *Exponential Smoothing* (ES) dengan parameter adaptif. Tabel 9 menampilkan kinerja metode MA, sementara Tabel 10 menunjukkan hasil evaluasi dari model ES. Metrik seperti ME, MAE, MSE, dan MAPE digunakan untuk mengukur sejauh mana kesalahan prediksi dari masing-masing metode. Melalui perbandingan ini, kita dapat menilai metode mana yang lebih efektif dan stabil dalam memprediksi pergerakan harga saham BBNI.

a. *Moving Average*

Tabel 9. MA BBNI

Metrik Evaluasi	*Moving Average*
ME	-39
MAE	150,97
MSE	219835,77
MAPE	2,1442%

Tabel 9 menampilkan hasil evaluasi model *Moving Average* (MA) untuk saham BBNI. Model *Moving Average* untuk saham BBNI menunjukkan kinerja yang kurang optimal, ditunjukkan oleh nilai MSE yang sangat tinggi (219835,77) dan MAE yang cukup besar (150,97), serta adanya bias negatif (ME -39). Meskipun MAPE relatif rendah (2,1442%), model ini tampaknya kurang mampu menangani fluktuasi besar dalam data saham BBNI,

sehingga mungkin perlu penyesuaian periode atau metode lain untuk meningkatkan akurasi prediksi.

b. *Exponential Smoothing* (*Adaptive Parameter*)

Tabel 10. ES BBNI

Metrik Evaluasi	*Exponential Smoothing* (*Adaptive Parameter*)
MASE	104,1996
ME	-19,4836
MSE	-19,7883
MAPE	1,4202%

Tabel 10 menampilkan hasil evaluasi model *Exponential Smoothing* (ES) dengan parameter adaptif untuk saham BBNI. Model *Exponential Smoothing* dengan parameter adaptif untuk saham BBNI menunjukkan kinerja yang buruk berdasarkan MASE yang sangat tinggi (104,1996), mengindikasikan ketidakmampuan model dalam memprediksi dengan akurat dibandingkan metode sederhana. Bias negatif (ME -19,4836) dan nilai MAPE yang relatif rendah (1,4202%) menunjukkan akurasi persentase yang cukup baik, tetapi nilai MSE yang negatif (-19,7883) menunjukkan adanya masalah data atau perhitungan. Model ini memerlukan perbaikan atau verifikasi data untuk meningkatkan keandalan prediksi.

6. Pemilihan Metode Terbaik BBNI Syariah

Tabel 11 dan Tabel 12 menyajikan hasil evaluasi kinerja dua metode peramalan. Masing-masing adalah *Moving Average* (MA) dan *Exponential Smoothing* (ES) dengan parameter adaptif, untuk data BNIS. Tabel 11 menunjukkan metrik evaluasi MA, meliputi *Mean Error* (ME), *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE), dan *Mean Absolute Percentage Error* (MAPE). Sementara itu, Tabel 12 memaparkan metrik evaluasi ES, termasuk *Mean Absolute Scaled Error* (MASE), ME, MSE, dan MAPE. Analisis perbandingan metrik ini dapat memberikan wawasan tentang akurasi dan efektivitas masing-masing metode dalam meramalkan data BNIS.

a. *Moving Average*

Tabel 11. MA BNIS

Metrik Evaluasi	*Moving Average*
ME	-4
MAE	28,16
MSE	1403,04
MAPE	2,2478%

Tabel 11 menampilkan hasil evaluasi model *Moving Average* (MA) untuk saham BNIS. Model *Moving Average* untuk saham BNIS menunjukkan kinerja yang cukup baik dengan MAPE yang relatif rendah (2,2478%) dan bias kecil (ME -4). Namun, nilai MAE (28,16) dan MSE (1403,04) mengindikasikan bahwa model masih menghasilkan beberapa kesalahan prediksi yang signifikan. Penyesuaian periode *Moving Average* atau pendekatan lain mungkin diperlukan untuk meningkatkan akurasi prediksi.

b. *Exponential Smoothing* (*Adaptive Parameter*)

Tabel 12. ES BNIS

Metrik Evaluasi	*Exponential Smoothing* (*Adaptive Parameter*)
MASE	23,06761
ME	-1,78404
MSE	-1,73984
MAPE	1,8655%

Tabel 12 menampilkan hasil evaluasi model *Exponential Smoothing* (ES) dengan parameter adaptif untuk saham BNIS. Model *Exponential Smoothing* dengan parameter adaptif untuk saham BNIS memiliki akurasi persentase yang cukup baik berdasarkan MAPE (1,8655%) dan bias yang sangat kecil (ME -1,78404). Namun, nilai MASE yang tinggi (23,06761) menunjukkan kinerja model yang kurang efektif dibandingkan metode sederhana, dan nilai MSE yang negatif (-1,73984) mengindikasikan adanya masalah data atau perhitungan. Model ini memerlukan verifikasi data atau penyesuaian untuk meningkatkan keandalan prediksi.

7. Pemilihan Metode Terbaik BBTN

Tabel 13 dan Tabel 14 menyajikan hasil evaluasi kinerja dua metode peramalan. Masing-masing adalah *Moving Average*(MA) dan *Exponential Smoothing* (ES) dengan parameter adaptif, untuk data BBTN. Tabel 13 menampilkan metrik evaluasi MA, yang mencakup *Mean Error* (ME), *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE), dan *Mean Absolute Percentage Error* (MAPE). Sementara itu, Tabel 14 memaparkan metrik evaluasi ES, termasuk Mean Absolute Scaled Error (MASE), ME, MSE, dan MAPE. Perbandingan metrik-metrik ini memungkinkan analisis terhadap tingkat akurasi dan keandalan kedua metode dalam meramalkan data BBTN.

a. *Moving Average*

Tabel 13. MA BBNI

Metrik Evaluasi	*Moving Average*
ME	-1
MAE	27,09
MSE	9232,93
MAPE	,.9682%

Tabel 13 menampilkan hasil evaluasi model *Moving Average* (MA) untuk saham BBTN. Model *Moving Average* untuk saham BBTN menunjukkan kinerja yang kurang optimal, dengan MAPE yang relatif tinggi (5,9682%) dan MSE yang besar (9232,93), meskipun bias sangat kecil (ME -1). Nilai MAE (27,09) menunjukkan adanya deviasi prediksi yang signifikan. Penyesuaian periode *Moving Average* atau penggunaan metode lain mungkin diperlukan untuk meningkatkan akurasi prediksi saham BBTN.

b. *Exponential Smoothing* (*Adaptive Parameter*)

Tabel 14. ES BBTN

Metrik Evaluasi	*Exponential Smoothing* (*Adaptive Parameter*)
MASE	23,72416
ME	-0,35211
MSE	-0,39327
MAPE	5,6775%

Tabel 14 menampilkan hasil evaluasi model *Exponential Smoothing* (ES) dengan parameter adaptif untuk saham BBTN. Model *Exponential Smoothing* dengan parameter adaptif untuk saham BBTN menunjukkan kinerja yang kurang baik, dengan MASE yang tinggi (23,72416) dan MAPE yang relatif besar (5,6775%), meskipun bias sangat kecil (ME -0,35211). Nilai MSE yang negatif (-0,39327) mengindikasikan adanya masalah data atau perhitungan. Model ini memerlukan verifikasi data atau penyesuaian parameter untuk meningkatkan akurasi prediksi.

V. SIMPULAN

Berdasarkan pembahasan Tabel 1 hingga 14, model *Moving Average* (MA) dan *Exponential Smoothing* (ES) dengan parameter adaptif untuk peramalan harga saham BBRI, BRIS, BBCA, BMRI, BBNI, BNIS, dan BBTN menunjukkan kinerja yang bervariasi. Secara umum, kedua model memiliki akurasi persentase yang cukup baik dengan nilai MAPE rendah (0,8215% hingga 5,9682%), terutama pada saham BBCA (MAPE 0,8215% untuk ES) dan BBRI (MAPE 1,00816% untuk ES). Namun, nilai MASE yang tinggi (23,06761 hingga 104,1996) pada semua saham menunjukkan bahwa kedua model kurang efektif dibandingkan metode naif. Bias prediksi umumnya kecil, baik positif (misalnya, ME 4 untuk MA BBRI) maupun negatif (misalnya, ME -39 untuk MA BBNI). Nilai MSE yang tinggi pada beberapa kasus (misalnya, 219835,77 untuk MA BBNI) dan anomali MSE negatif pada model ES (BBNI, BNIS, BBTN) mengindikasikan masalah data atau perhitungan. Secara keseluruhan, model ES cenderung lebih presisi (MSE rendah pada BBRI, BRIS, BBCA) dibandingkan MA, tetapi keduanya memerlukan penyesuaian parameter, verifikasi data, atau pendekatan alternatif untuk meningkatkan akurasi dan keandalan prediksi.

Untuk meningkatkan kinerja peramalan harga saham BBRI, BRIS, BBCA, BMRI, BBNI, BNIS, dan BBTN, disarankan untuk melakukan verifikasi dan koreksi data guna mengatasi anomali seperti nilai MSE negatif pada model *Exponential Smoothing* (ES). Selain itu, penyesuaian parameter pada model ES dan *Moving Average* (MA), seperti optimasi periode atau bobot, dapat membantu mengurangi nilai MASE yang tinggi dan meningkatkan efektivitas dibandingkan metode naif. Mengingat ES menunjukkan presisi lebih baik pada beberapa saham (BBRI, BRIS, BBCA), pendekatan ini dapat diprioritaskan dengan tuning parameter yang lebih cermat. Alternatifnya, eksplorasi metode lain seperti ARIMA atau *machine learning* (misalnya, LSTM) dapat dipertimbangkan untuk menangani fluktuasi data yang besar dan meningkatkan akurasi prediksi secara keseluruhan.

REFERENSI

[1] Republik Indonesia, *Undang-Undang Republik Indonesia Nomor 10 Tahun 1998*. Indonesia, 1998.

[2] A. P. Putri and M. Mesrawati, "Pengaruh Analisis Teknikal Terhadap Trend Pergerakan Harga Saham Perusahaan Subsektor Hotel Dan Restoran," *EKUITAS (Jurnal Ekonomi Dan Keuangan)*, vol. 3, no. 3, pp. 324–343, 2019.

[3] N. N. M. Cahyani and L. P. Mahyuni, "Akurasi *Moving Average* Dalam Prediksi Saham LQ45 di Bursa Efek Indonesia," *Jurnal Manajemen*, vol. 9, no. 7, pp. 2769–2789, 2020.

[4] S. Sismi and M. Y. Darsyah, "Perbandingan Prediksi Harga Saham PT. BRI, Tbk Dengan METODE ARIMA Dan MOVING AVERAGE," in *Prosiding Seminar Nasional Mahasiswa Unimus*, 2018.

[5] H. H. Van Rossum, "Moving average quality control: principles, practical application and future perspectives," *Clinical Chemistry and Laboratory Medicine (CCLM)*, vol. 57, no. 6, pp. 773–782, May 2019, doi: 10.1515/cclm-2018-0795.

[6] C. Chiarella, X.-Z. He, and C. Hommes, "A dynamic analysis of moving average rules," *Journal of Economic Dynamics and Control*, vol. 30, no. 9–10, pp. 1729–1753, Sep. 2006, doi: 10.1016/j.jedc.2005.08.014.

[7] H. Sinaga and N. Irawati, "A medical disposable supply demand forecasting by *Moving Average* and *Exponential Smoothing* method," in *Proceedings of the 2nd Workshop on Multidisciplinary and Applications (WMA) 2018, 24-25 January 2018, Padang, Indonesia*, 2020.

[8] E. S. Gardner, "Exponential smoothing: The state of the art," *Journal of Forecasting*, vol. 4, no. 1, pp. 1–28, Jan. 1985, doi: 10.1002/for.3980040103.

[9] B. Billah, M. L. King, R. D. Snyder, and A. B. Koehler, "Exponential smoothing model selection for forecasting," *International Journal of Forecasting*, vol. 22, no. 2, pp. 239–247, Apr. 2006, doi: 10.1016/j.ijforecast.2005.08.002.

[10] A. M. Khan and M. Osińska, "Comparing forecasting accuracy of selected grey and time series models based on energy consumption in Brazil and India," *Expert Systems with Applications*, vol. 212, p. 118840, Feb. 2023, doi: 10.1016/j.eswa.2022.118840.

[11] S. Kim and H. Kim, "A new metric of absolute percentage error for intermittent demand forecasts," *International Journal of Forecasting*, vol. 32, no. 3, pp. 669–679, Jul. 2016, doi: 10.1016/j.ijforecast.2015.12.003.

[12] M. R. Nieto, R. B. C. Benitez, and J. N. Martinez, "Comparing models to forecast cargo volume at port terminals," *JART*, vol. 19, no. 3, pp. 238–249, Jun. 2021, doi: 10.22201/icat.24486736e.2021.19.3.1695.

[13] M. Alexander, R. Kusleika, and J. Walkenbach, *Excel 2019 Bible*. John Wiley & Sons, 2018.

[14] D. Novita, F. P. Sihotang, and S. Khairani, "Pelatihan Penggunaan Microsoft Excel Untuk Mengolah Data Bagi Siswa/i SMK Bina Cipta Palembang," *fordicate*, vol. 2, no. 2, pp. 109–118, Apr. 2023, doi: 10.35957/fordicate.v2i2.4759.

[15] D. Sugiyono, "Metode penelitian pendidikan pendekatan kuantitatif, kualitatif dan R&D," 2013.

Perancangan dan Pengujian Kinerja Sistem Informasi Zakat Fitrah Berbasis IoT

Rizal Aziz Pradana[1], Unan Yusmaniar Oktiawati[1,*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
rizalazizpradana1@mail.ugm.ac.id
[*]Korespondensi: unan_yusmaniar@ugm.ac.id;

Abstract – In implementing the management of zakat fitrah, especially at the Nurul Islam Sabrang Mosque, calculations are still being carried out manually. The zakat fitrah committee must weigh the weight of the zakat and then add up the total weight manually every time zakat is received. Then, the dissemination of information related to the management of zakat fitrah is still done by word of mouth, so it is less efficient. Therefore, this research makes an Internet of Things-based zakat fitrah information system to overcome these problems.. This system enables the automatic calculation of the weight of zakat fitrah through a system connected to a load cell sensor, and information related to zakat fitrah can be easily obtained in real time through a website. Then, several tests related to functionality were carried out to determine the feasibility of the system in the form of black box testing, scale accuracy, and QoS on the performance of the HTTP protocol when sending data from the sensor node to the server. The test results demonstrate that the system is feasible to use, achieving high accuracy, satisfactory QoS, and good HTTP protocol performance in terms of parameters such as delay, packet loss, and packet delivery when sending data from node sensor to server.

Keywords – Zakat fitrah, Internet of Things, Load Cell, QoS, HTTP

Intisari – Dalam pelaksanaan pengelolaan zakat fitrah, khususnya di Masjid Nurul Islam Sabrang saat ini masih dilakukan perhitungan secara manual. Panitia zakat fitrah harus menimbang berat zakat kemudian menjumlahkan total berat secara manual setiap kali ada zakat yang masuk. Kemudian, dalam penyebaran informasi terkait pengelolaan zakat fitrah masih dilakukan dari mulut ke mulut sehingga kurang efisien. Oleh karena itu, penelitian ini membuat sistem informasi zakat fitrah berbasis *Internet of Things* untuk mengatasi permasalahan tersebut. Sistem ini memungkinkan perhitungan berat zakat fitrah akan dilakukan otomatis oleh sistem yang terhubung dengan sensor *load cell* dan informasi terkait zakat fitrah juga akan mudah didapatkan melalui *website* secara *real time*. Kemudian, dilakukan beberapa pengujian terkait fungsionalitas untuk mengetahui kelayakan sistem berupa *black box testing*, akurasi timbangan, dan QoS terhadap performa protokol HTTP pada saat pengiriman data dari *node* sensor ke *server*. Hasil pengujian tersebut menunjukkan bahwa sistem layak digunakan dengan akurasi termasuk tinggi dan QoS performa protokol HTTP dengan parameter *delay*, *packet loss*, dan *packet delivery* saat pengiriman data dari *node* sensor ke *server* berkategori memuaskan.

Kata kunci – Zakat Fitrah, *Internet of Things*, *Load Cell*, QoS, HTTP

I. PENDAHULUAN

Indonesia merupakan negara dengan mayoritas penduduk muslim terbanyak di dunia. Dari 276 juta penduduk Indonesia tahun 2023 lebih dari 87% di antaranya adalah muslim atau beragama Islam [1]. Sebagai seorang muslim, membayar zakat merupakan salah satu rukun Islam yang harus dipenuhi untuk menyempurnakan ibadah. Zakat merupakan harta yang wajib diberikan oleh seorang muslim atau badan usaha yang dimiliki oleh muslim kepada orang yang berhak menerimanya sesuai dengan syariat Islam [2]. Salah satu jenis zakat tersebut adalah zakat fitrah. Zakat fitrah adalah zakat yang wajib dikeluarkan oleh seorang muslim untuk menyucikan ibadah mereka [3]. Besarnya pembayaran zakat fitrah di Indonesia berkisar 2,5 kg atau 2,7 kg beras dan ditunaikan saat bulan ramadan sebelum fajar Idul Fitri setiap tahunnya.

Saat ini pelaksanaan pengelolaan zakat fitrah, khususnya di Masjid Nurul Islam Sabrang Yogyakarta, masih melakukan perhitungan secara manual. Panitia zakat fitrah harus menimbang berat zakat, lalu menjumlahkan total berat secara manual setiap kali ada zakat yang masuk. Selain memakan waktu, hal tersebut juga memungkinkan terjadi kesalahan perhitungan. Kemudian, dalam penyebaran informasi terkait pengelolaan zakat fitrah masih dilakukan dari mulut ke mulut. Ketika masyarakat ingin mengetahui informasi terkait zakat fitrah, maka harus datang ke Masjid dan bertanya langsung ke panitia zakat fitrah sehingga kurang efisien.

Berdasarkan uraian di atas, penelitian ini bermaksud untuk mengimplementasikan sistem informasi zakat fitrah berbasis *Internet of Things* (IoT). IoT merupakan teknologi yang memungkinkan pengguna untuk mengelola dan mengoptimalkan perangkat keras melalui jaringan internet [4]. Dengan adanya teknologi ini, perhitungan berat zakat fitrah akan dilakukan otomatis oleh sistem yang terhubung dengan *node* sensor dan informasi terkait zakat fitrah juga akan mudah didapatkan melalui *web app* secara *real time*. Selain itu, peneliti juga akan menganalisis kinerja pengiriman data dari *node* sensor menuju ke *server* melalui protokol HTTP. Hasil dari penelitian akan menghasilkan sebuah sistem informasi zakat fitrah berbasis IoT, serta dapat mengetahui nilai dari *Quality of Service* saat pengiriman data dari *node* sensor ke *server*.

II. DASAR TEORI

A. Sistem Informasi

Sistem informasi merupakan rangkaian terintegrasi dari komponen, proses, dan teknologi yang digunakan untuk mengumpulkan, mengelola, memproses, dan menyebarkan informasi. Sistem informasi terdiri dari beberapa elemen, seperti perangkat keras, perangkat lunak, dan jaringan komunikasi untuk menghubungkan berbagai komponen dengan pengguna. Di era sekarang, sistem informasi telah digunakan di segala bidang termasuk penggunaan untuk pengelolaan zakat [5].

B. Zakat Fitrah

Menurut [3] terdapat dua pengertian mengenai zakat fitrah. Pertama, zakat fitrah merupakan zakat yang dikeluarkan untuk menyucikan orang berpuasa baik dari ucapan atau perilaku yang tidak bermanfaat. Kedua, zakat fitrah merupakan zakat karena ciptaan yang berarti zakat fitrah ini diwajibkan kepada setiap umat islam, sehingga disebut dengan zakat badan atau pribadi. Adapun jenis zakat yang harus dibayarkan adalah bahan makanan pokok dengan ukuran satu *sha'*. Di Indonesia sendiri, besarnya pembayaran zakat fitrah adalah 2,5 kg atau 2,7 kg beras. Kemudian, untuk waktu pembayarannya adalah saat bulan Ramadan dengan batasan sebelum fajar Idul Fitri setiap tahunnya. Untuk sistem ini, digunakan berat 2,7 kg per zakat [6].

C. *Internet of Things*

Internet of Things (IoT) merupakan sistem yang terdiri dari sensor, aktuator, dan *smart object* yang bertujuan untuk menghubungkan semua hal, termasuk benda di keseharian dan industri, dibuat memiliki *intelligent*, *programmable*, dan mampu berinteraksi dengan manusia dan sesamanya [7], [8]. Konsep *Internet of Things* bertujuan untuk meningkatkan manfaat dari koneksi internet yang terus terhubung [9]. Oleh karena itu, jaringan internet merupakan faktor penting untuk kelancaran perangkat IoT yang menjadi penghubung antara perangkat dan sistem, sedangkan manusia menjadi pemantau dari setiap perilaku sistem saat bekerja [10]. IoT juga digunakan untuk mendukung sistem informasi zakat dan transaksi jual beli salak [11].

D. *Node* Sensor

Node sensor adalah kombinasi antara perangkat mikrokontroler dan sensor. *Node* sensor berperan sebagai pengambil dan pengolah data. *Node* sensor ini akan membaca data dari sensor-sensor tersebut dan mengirimkannya ke sistem lain seperti komputer untuk dianalisis. Dengan menggunakan *node* sensor, memudahkan untuk memantau lingkungan sekitar dan mendapatkan informasi yang diperlukan untuk pengambilan keputusan.

E. Pengujian *Black Box*

Pengujian *black box* adalah metode pengujian sistem yang melibatkan pengujian fungsionalitas dan perilaku sistem tanpa memperhatikan struktur atau logika *internal* dari kode program yang diuji. Dalam pengujian *black box*, pengujian dilakukan berdasarkan spesifikasi kebutuhan dan fungsionalitas yang telah ditentukan, tanpa pengetahuan detail tentang bagaimana perangkat lunak diimplementasikan. Pengujian ini bertujuan untuk menguji sistem secara keseluruhan dari perspektif pengguna. Pengujian ini berfokus pada *input* dan *output* yang dihasilkan oleh sistem, serta interaksi antara sistem dengan pengguna atau komponen *eksternal*. Tujuan utama dari pengujian *black box* adalah untuk memastikan bahwa sistem memberikan hasil yang diharapkan sesuai dengan spesifikasi dan kebutuhan yang telah ditetapkan.

F. Skala Likert

Skala Likert merupakan metode yang penelitian untuk mengukur pendapat, sikap, atau persepsi seseorang terhadap suatu topik. Skala ini menggunakan pernyataan atau pertanyaan dengan pilihan jawaban berupa tingkat persetujuan. Jumlah minimal kategori yang diperlukan dalam skala Likert adalah dua, di mana responden dapat memilih tingkat persetujuan mereka yang paling sesuai. Jawaban responden tersebut dapat dianalisis datanya untuk mendapatkan pemahaman tentang preferensi, sikap, atau kepuasan orang-orang terhadap suatu topik. Hasil dari skala Likert dapat membantu dalam pengambilan keputusan atau perbaikan suatu sistem.

G. Akurasi

Akurasi merupakan ukuran seberapa jauh hasil pengukuran mendekati nilai yang sebenarnya, dalam hal ini adalah timbangan. Nilai akurasi dinyatakan dalam bentuk persentase kesalahan relatif atau selisih perbedaan nilai dengan standar yang diketahui. Semakin rendah persentase kesalahan relatif atau perbedaannya, semakin tinggi akurasinya. Akurasi timbangan dalam penimbangan zakat fitrah sangat penting karena dapat mempengaruhi proses perhitungan ulang zakat. Oleh karena itu, penggunaan timbangan yang akurat menjadi faktor kunci dalam pengelolaan zakat fitrah.

H. *Quality of Services*

Quality of Services atau QoS merupakan sebuah metode untuk mengukur keandalan jaringan [12]. Dalam pengukuran QoS dalam suatu jaringan terdapat beberapa parameter, yaitu *delay, packet loss,* dan *packet delivery. Delay* merupakan waktu keterlambatan saat pengiriman paket dari asal ke tujuan, sedangkan *packet loss* merupakan jumlah paket yang hilang selama proses pengiriman data ke tujuan. Jumlah paket yang hilang ini bisa lebih dari satu paket. Kemudian, *packet delivery* adalah rasio antara jumlah paket yang berhasil dikirim dengan total jumlah paket yang dikirim. Dari hasil indeks setiap parameter tersebut, nantinya akan dijumlah dan dicari rata-ratanya hingga diperoleh nilai indeks performa kualitas jaringan [13].

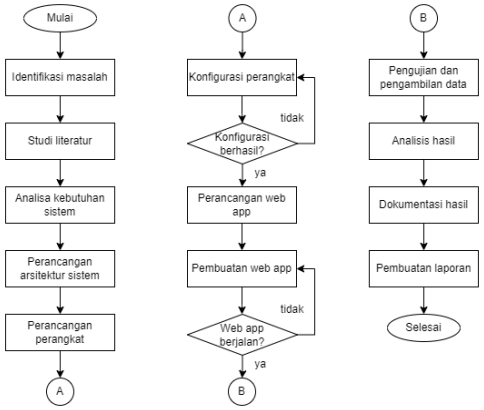
I. *Hypertext Transfer Protocol*

Hypertext Transfer Protocol (HTTP) merupakan salah satu protokol yang digunakan untuk komunikasi antar perangkat IoT dengan server atau *platform* lain. Berbagai jenis data dapat dikirimkan dengan HTTP melalui web, seperti dokumen, file, gambar, ataupun video. HTTP menggunakan model *client-server* pada *web* dan dibangun di atas *Transmission Control Protocol* [14]. Protokol tersebut memungkinkan perangkat IoT untuk mengirim permintaan dan menerima *respons* melalui internet. Melalui HTTP, perangkat IoT dapat berkomunikasi dengan aplikasi web atau layanan *cloud* yang terhubung [15]. HTTP menggunakan metode permintaan seperti *GET*, *POST*, *PUT*, dan *DELETE* untuk mengirimkan permintaan dari perangkat IoT ke server. Perangkat IoT dapat mengirimkan permintaan *GET* untuk mengambil data, *POST* untuk mengirimkan data baru, *PUT* untuk memperbarui data, atau *DELETE* untuk menghapus data dari server. Selain itu, HTTP juga mendefinisikan *header* dan *respons* yang digunakan untuk mengirimkan informasi tambahan terkait permintaan dan *respons*. *Header* dapat digunakan untuk mengidentifikasi perangkat IoT, menyampaikan informasi otorisasi, atau memberikan instruksi tambahan kepada *server*.

III. METODOLOGI

Penelitian diawali dengan melakukan identifikasi masalah untuk menentukan topik apa yang akan dibahas dalam penelitian. Setelah itu dilakukan studi literatur untuk mendapatkan referensi dan informasi yang dapat menunjang penelitian. Kemudian dilakukan analisis kebutuhan sistem, seperti *software*, *hardware*, dan teknologi yang akan digunakan dalam penelitian. Berikutnya dilakukan perancangan arsitektur sistem sehingga didapatkan alur kerja sistem dari tahap pengumpulan, pemrosesan, hingga penampilan data pada sistem.

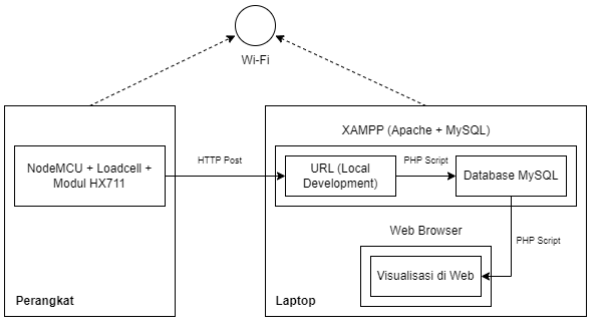
Masuk ke tahap teknis, penelitian dimulai dari perancangan perangkat dengan membuat rangkaian elektronik untuk menghubungkan seluruh komponen. Kemudian, dilakukan konfigurasi perangkat berupa sensor *load cell* dan NodeMCU ESP8266 dengan Arduino IDE. Konfigurasi berhasil dilakukan jika pembacaan data berhasil dilakukan dengan akurat dan berhasil terkoneksi dengan Wi-Fi. Apabila berhasil, selanjutnya dilakukan perancangan *web app* dengan membuat *use case diagram*, *activity diagram*, dan *wireframe*. Setelah itu, dilakukan pembuatan *web app* yang dibuat menggunakan *framework* Laravel dan berbasis PHP. Jika *web app* berhasil berjalan dan berfungsi dengan baik, maka pengujian dan pengambilan data dapat dilakukan. Beberapa pengujian dilakukan pada tahap ini, seperti pengujian *black box*, akurasi timbangan, dan QoS dengan parameter *delay*, *packet loss*, dan *packet delivery*. Setelah itu, dilakukan analisis hasil pengujian terkait dengan kelayakan sistem. Kemudian dilakukan dokumentasi hasil dan pembuatan laporan akhir. Diagram alir metode penelitian dapat dilihat pada Gambar 1.



Gambar 1. Diagram alir metode penelitian

A. Perancangan Sistem

Perancangan diawali dari sensor *load cell* yang terhubung dengan modul HX711 dan NodeMCU ESP8266. Sensor *load cell* membaca berat zakat fitrah dan mengirimkan data tersebut ke server melalui Wi-Fi dengan HTTP. Data yang dikirimkan tersebut berformat *plain text* atau JSON. Kemudian, data tersebut akan diproses dan disimpan di *database* MySQL sebagai pusat penyimpanan data. Data yang sudah tersimpan selanjutnya akan ditampilkan oleh *web server* di *komputer* yang terhubung ke *server*. *Server* yang digunakan merupakan sebuah *komputer* yang terinstal layanan *web server* Apache dan *database server* MySQL, yakni dengan XAMPP. Selanjutnya, data tersebut akan divisualisasikan dalam bentuk grafik dan tabel pada *website*. Analisis QoS dilakukan pada proses pengiriman data hingga data sampai ditampilkan di *website*. Rancangan arsitektur sistem dapat dilihat pada Gambar 2.

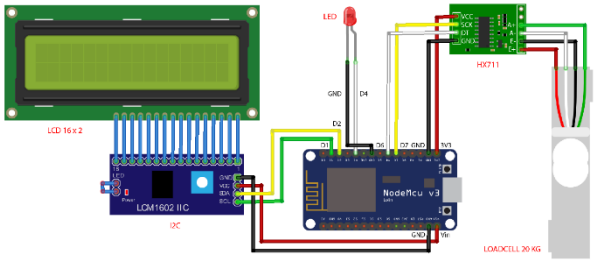


Gambar 2. Arsitektur sistem

B. Perancangan Perangkat

Perangkat yang digunakan pada penelitian ini terdiri dari NodeMCU ESP8266, sensor *load cell*, modul HX711, LCD, modul I2C, dan LED. NodeMCU digunakan sebagai perangkat pemroses program dan agar dapat terhubung langsung dengan Wi-Fi. Kemudian terdapat sensor *load cell* yang terhubung dengan modul HX711 digunakan untuk membaca berat zakat fitrah dan mengonversi ke dalam

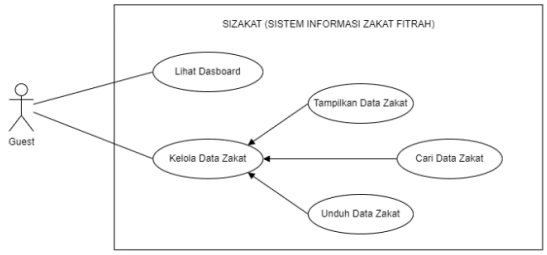
besaran tegangan. Terdapat juga LCD dan I2C yang saling terhubung yang digunakan sebagai penampil karakter berupa huruf sebagai informasi terkait hasil timbang. Selanjutnya ada LED yang berfungsi sebagai indikator koneksi Wi-Fi dari NodeMCU. Rancangan perangkat secara detail dapat dilihat pada Gambar 3.



Gambar 3. Rancangan perangkat

C. *Use Case Diagram*

Use case diagram digunakan untuk menggambarkan hubungan interaksi antara aktor dan sistem. Sistem yang dibuat menggunakan dua aktor, yaitu *guest* dan admin. *Use case diagram guest* dapat dilihat pada Gambar 4 dan *use case diagram* admin dapat dilihat pada Gambar 5.



Gambar 4. *Use case diagram guest*

Gambar 5. *Use case diagram* admin

D. Pengujian *Black box*

Pengujian *black box* bertujuan untuk menguji sistem dari perspektif pengguna dan memastikan bahwa sistem memberikan hasil yang diharapkan sesuai dengan kebutuhan yang telah ditetapkan. Pengujian ini berfokus pada *input* dan *output* yang dihasilkan oleh sistem, serta interaksi antara sistem dengan pengguna atau komponen eksternal. Pengambilan data dilakukan dengan media kuesioner menggunakan Google Form dengan responden yang dituju adalah panitia zakat fitrah dan jamaah Masjid Nurul Islam Sabrang. Terdapat 8 pertanyaan yang dibuat seperti pada Tabel 1.

Tabel 1. Daftar pernyataan kuesioner

No	Pernyataan	Role
1	Halaman *dashboard guest* menampilkan informasi berupa total berat zakat, jumlah muzakki, dan grafik perolehan zakat per hari.	*Guest*
2	Fitur pencarian dan unduh data zakat pada halaman data zakat *guest* dapat dijalankan.	
3	Fungsi login ke halaman admin dapat dilakukan.	
4	Halaman *dashboard* admin menampilkan informasi berupa total berat zakat, jumlah muzakki, dan grafik perolehan zakat per hari.	
5	Halaman timbang admin menampilkan informasi terkait berat zakat dan jumlah orang secara *real time*, serta fitur *input* nama muzakki dapat berfungsi.	Admin
6	Fitur pencarian, edit, unduh, dan hapus data zakat pada halaman data zakat dapat berfungsi.	
7	Fitur pencarian, tambah, edit, dan hapus data pengguna pada halaman pengguna dapat dijalankan.	
8	Fungsi *logout* sistem dapat dilakukan.	

Data hasil kuesioner tersebut, berikutnya akan diolah dan disajikan dalam bentuk tabel menggunakan skala Likert yang kemudian dilakukan analisis untuk mendapatkan kesimpulan. Adapun untuk perhitungan yang dilakukan akan menggunakan skala interval untuk menunjukkan hasil kuantitatif. Setiap jawaban akan diberikan nilai 3 untuk S (Setuju), nilai 2 untuk Kurang Setuju (KS) dan nilai 1 untuk Tidak Setuju (TS).

Kemudian dilakukan perhitungan nilai persentase menggunakan (1) dan disimpulkan berdasarkan interpretasi nilai sesuai dengan Tabel 2.

$$Nilai\ persentase\ = \frac{Total\ nilai}{Nilai\ tertinggi}\ x\ 100\% \quad (1)$$

Tabel 2. Interpretasi Nilai

Nilai	Keterangan
0% - 33,3%	Tidak Setuju
33,3% - 66,7%	Kurang Setuju
66,6% - 100%	Setuju

E. Pengujian Akurasi Timbangan

Pengujian akurasi timbangan dilakukan dengan cara membandingkan nilai timbangan dari sistem dengan data referensi yang sudah diketahui berat aslinya. Dimana data referensi atau nilai sebenarnya yang digunakan adalah 2.7 kg beras. Skenario pengujian dilakukan dengan melakukan percobaan penimbangan berat beras zakat fitrah sebanyak 10 kali percobaan. Dari hasil pengukuran tersebut, nantinya akan dilakukan analisis untuk mencari nilai rata – rata dan persentase *error*-nya terlebih dahulu. Rumus untuk mencari nilai persentase *error* dapat dilihat pada (2).

$$\%Error\ = \left[\frac{X - Xi}{X}\right] x\ 100\% \quad (2)$$

Persamaan (2) didefinisikan oleh beberapa nilai. X merupakan nilai sebenarnya dan Xi adalah nilai pengukuran. Jika nilai persentase *error* sudah didapatkan, perhitungan untuk mencari nilai akurasi dapat dilakukan dengan mengurangi 100% dengan nilai dari persentase *error*.

F. *Quality of Services* (QoS)

Pengujian ini dilakukan untuk melihat kinerja saat proses pengiriman data dari *node* sensor ke server menggunakan parameter QoS yakni *delay*, *packet loss*, dan *packet delivery*. Skenario pengujian dilakukan dengan melakukan pengiriman data secara terus menerus dari *node* sensor ke *server* melalui protokol HTTP dengan sepuluh kali percobaan masing-masing selama satu menit. Pengujian dilakukan pada waktu sore hari pukul 03.00 WIB dengan jarak perangkat adalah 150 cm tanpa penghalang. Pengujian menggunakan jaringan dari *mobile hotspot* dengan *provider* Indosat. Seluruh pengujian menggunakan aplikasi Wireshark untuk membantu merekam lalu lintas data. Hierarki pengujian QoS dapat dilihat pada Gambar 6.

Gambar 6. Hierarki pengujian QoS

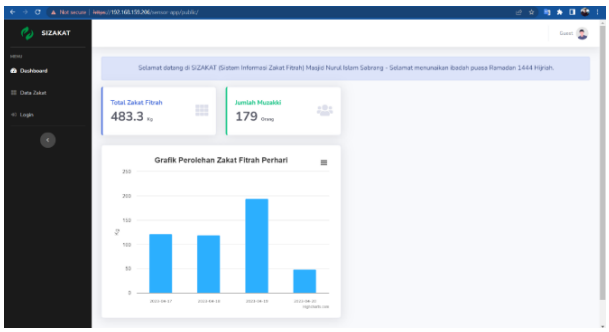
Data yang rekam tersebut nantinya akan dilakukan filter sesuai dengan kebutuhan perhitungan parameter yang dicari, yaitu *delay*, *packet loss*, dan *packet delivery*. Kemudian jika nilai dari setiap parameter QoS sudah didapatkan, maka akan dilakukan pengelompokan sesuai dengan kategori QoS versi TIPHON untuk mengetahui hasilnya.

IV. HASIL DAN PEMBAHASAN

A. *Black box*

1. Halaman *dashboard guest*

Pada bagian ini responden diminta untuk mengakses *web* dan masuk menu *dashboard* untuk melihat apakah jumlah zakat fitrah, jumlah muzakki, dan grafik perolehan zakat fitrah per hari berhasil ditampilkan dan sesuai dengan data sebenarnya. Halaman *dashboard guest* dapat dilihat pada Gambar 7.



Gambar 7. Halaman *dashboard guest*

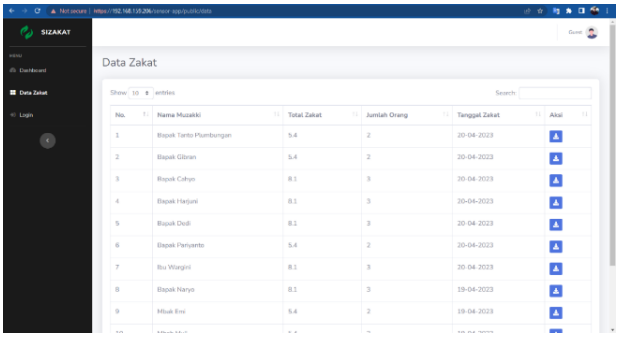
Adapun skenario dan hasil pengujian menu *dashboard guest* dengan metode *black box testing* ditunjukkan pada tabel 3.

Tabel 3. Hasil pengujian halaman *dashboard guest* dengan *black box testing*

Butir Uji	Hasil yang Diharapkan	Hasil Sebenarnya	Kesimpulan
Mengakses menu *dashboard guest* dan melihat hasil yang ditampilkan	Dapat menampilkan data jumlah zakat fitrah, jumlah muzakki, dan grafik perolehan zakat fitrah per hari dan sesuai dengan data sebenarnya	Menampilkan data jumlah zakat fitrah, jumlah muzakki, dan grafik perolehan zakat fitrah per hari dan sesuai dengan data sebenarnya	Berhasil

2. Halaman data zakat *guest*

Responden diminta untuk mengakses menu data zakat *guest* dan melakukan pencarian data secara acak serta mengunduh data tersebut. Halaman data zakat *guest* dapat dilihat pada Gambar 8.



Gambar 8. Halaman data zakat *guest*

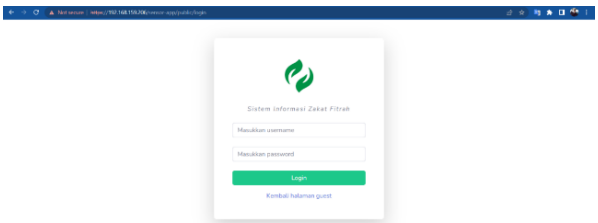
Adapun skenario dan hasil pengujian menu data zakat *guest* dengan metode *black box testing* ditunjukkan pada Tabel 4.

Tabel 4. Hasil pengujian halaman data zakat *guest* dengan *black box testing*

Butir Uji	Hasil yang Diharapkan	Hasil Sebenarnya	Kesimpulan
Mengakses menu data zakat *guest*, mencari data acak, mengunduh data	Menu data zakat *guest* dapat diakses dan fitur pencarian serta unduh data dapat berfungsi.	Menu data zakat *guest* dapat diakses dan fitur pencarian serta unduh data dapat berfungsi.	Berhasil

3. Halaman *login* admin

Pada bagian ini responden diminta untuk mengisi *username* dan *password* untuk *login* ke dalam sistem. Halaman *login* admin dapat dilihat pada Gambar 9.



Gambar 9. Halaman *login* admin

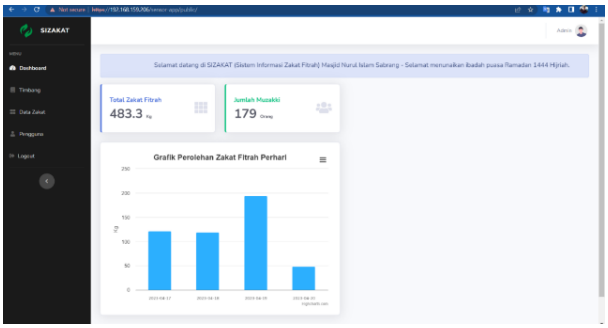
Adapun skenario dan hasil pengujian menu *login* dengan metode *black box testing* ditunjukkan pada Tabel 5.

Tabel 5. Hasil pengujian halaman *login* admin dengan *black box testing*

Butir Uji	Hasil yang Diharapkan	Hasil Sebenarnya	Kesimpulan
Mengisi *username* dan *password* pada halaman *login*	Dapat melakukan *login* ke dalam sistem	Responden dapat melakukan *login* ke dalam sistem	Berhasil

4. Halaman *dashboard admin*

Pada bagian ini responden diminta untuk masuk ke menu *dashboard* admin untuk melihat apakah jumlah zakat fitrah, jumlah muzakki, dan grafik perolehan zakat fitrah per hari berhasil ditampilkan dan sesuai dengan data sebenarnya. Halaman *dashboard* admin dapat dilihat pada Gambar 10.



Gambar 10. Halaman *dashboard* admin

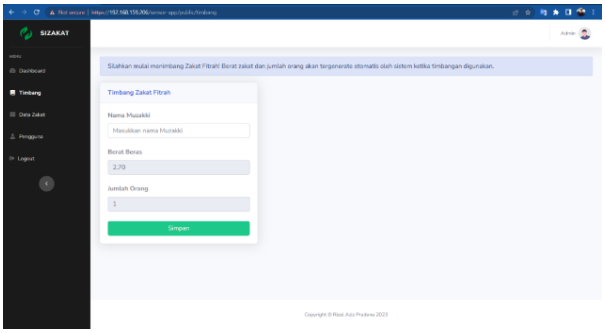
Adapun skenario dan hasil pengujian menu *dashboard* admin dengan metode *black box testing* ditunjukkan pada Tabel 6.

Tabel 6. Hasil pengujian halaman *dashboard* admin dengan *black box testing*

Butir Uji	Hasil yang Diharapkan	Hasil Sebenarnya	Kesimpulan
Mengakses menu dashboard admin	Menampilkan data jumlah zakat fitrah, jumlah muzakki, dan grafik perolehan zakat fitrah per hari	Menampilkan data jumlah zakat fitrah, jumlah muzakki, dan grafik perolehan zakat fitrah per hari	Berhasil

5. Halaman *timbang admin*

Responden diminta mengakses menu timbang untuk mengecek berat zakat dan jumlah orang yang tertampil sesuai dengan berat timbangan dibagi 2,7 kg serta memasukkan data zakat berupa nama muzakki. Halaman timbang admin dapat dilihat pada Gambar 11.



Gambar 11. Halaman timbang admin

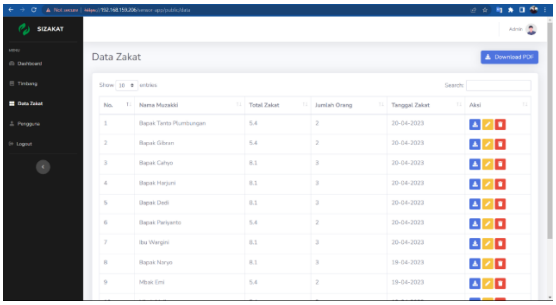
Adapun skenario dan hasil pengujian menu timbang admin dengan metode *black box testing* ditunjukkan pada Tabel 7.

Tabel 7. Hasil pengujian halaman timbang admin dengan *black box testing*

Butir Uji	Hasil yang Diharapkan	Hasil Sebenarnya	Kesimpulan
Mengakses menu timbang admin, mengecek berat zakat dan jumlah orang yang tertampil serta memasukkan data zakat	Berat yang tertampil pada web sama dengan berat timbangan serta dapat memasukkan data zakat berupa nama muzakki pada sistem	Berat yang tertampil pada web sama dengan berat timbangan serta dapat memasukkan data zakat berupa nama muzakki pada sistem	Berhasil

6. Halaman data zakat admin

Responden diminta untuk mengakses menu data zakat dan melakukan pencarian data secara acak. Setelah itu responden dapat mengubah, menghapus serta mengunduh data tersebut. Halaman data zakat admin dapat dilihat pada Gambar 12 berikut.



Gambar 12. Halaman data zakat admin

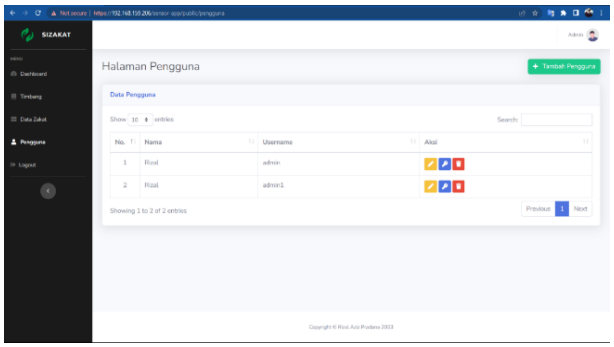
Adapun skenario dan hasil pengujian menu data zakat admin dengan metode *black box testing* ditunjukkan pada Tabel 8.

Tabel 8. Hasil pengujian halaman data zakat admin dengan *black box testing*

Butir Uji	Hasil yang Diharapkan	Hasil Sebenarnya	Kesimpulan
Mengakses menu data zakat admin, mencari data acak, mengubah data, menghapus data, dan mengunduh data	Menu data zakat admin dapat diakses dan fitur pencarian, ubah data, hapus data, serta unduh data dapat berfungsi.	Menu data zakat admin dapat diakses dan fitur pencarian, ubah data, hapus data, serta unduh data dapat berfungsi.	Berhasil

7. Halaman pengguna admin

Responden diminta untuk mengakses menu pengguna dan melakukan pencarian data secara acak. Kemudian responden mencoba untuk menambah, mengubah, serta menghapus data tersebut. Halaman pengguna admin dapat dilihat pada Gambar 13.



Gambar 13. Halaman pengguna admin

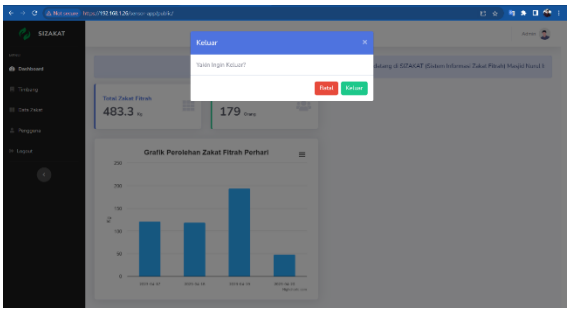
Adapun skenario dan hasil pengujian menu pengguna admin dengan metode *black box testing* ditunjukkan pada Tabel 9.

Tabel 9. Hasil pengujian halaman pengguna admin dengan *black box testing*

Butir Uji	Hasil yang Diharapkan	Hasil Sebenarnya	Kesimpulan
Mengakses menu pengguna admin, mencari data acak, mengubah, dan menghapus data	Menu pengguna admin dapat diakses dan fitur pencarian, tambah, ubah, serta hapus data dapat berfungsi.	Menu pengguna admin dapat diakses dan fitur pencarian, tambah, ubah, serta hapus data dapat berfungsi.	Berhasil

8. Halaman *logout* admin

Responden diminta untuk mengakses menu *logout* untuk keluar dari sistem. Halaman *logout* admin dapat dilihat pada Gambar 14.



Gambar 14. Halaman *logout*

Adapun skenario dan hasil pengujian menu *logout guest* dengan metode *black box testing* ditunjukkan pada Tabel 10.

Tabel 10. Hasil pengujian halaman *logout* admin dengan *black box testing*

Butir Uji	Hasil yang Diharapkan	Hasil Sebenarnya	Kesimpulan
Menekan button *logout*	Keluar dari sistem	Keluar dari sistem	Berhasil

B. Akurasi Timbangan

Pengujian akurasi dilakukan dengan cara membandingkan nilai timbangan dari sistem dengan data referensi yang sudah diketahui berat aslinya. Terdapat sepuluh kali percobaan dengan data referensi yang digunakan adalah 2.7 kg. Pengukuran akurasi timbangan dapat dilihat pada Tabel 11.

Tabel 11. Pengukuran akurasi timbangan

Pengukuran ke-n	Hasil (Kg)
1	2,70
2	2,69
3	2,69
4	2,70
5	2,68
6	2,70
7	2,70
8	2,71
9	2,69
10	2,70

Berdasarkan dari hasil pengukuran akurasi timbangan di atas, diketahui bahwa nilai terkecil sebesar 2.68 kg dan nilai terbesar adalah 2.71 kg. Apabila diambil rata-rata secara keseluruhan, maka didapatkan hasil sebesar 2.696 kg. Dari rata-rata tersebut dilakukan perhitungan untuk mencari persentase *error* dengan mengurangi data referensi dengan data pengukuran, kemudian membagi hasilnya dengan data referensi dan dikalikan dengan 100%, sehingga didapatkan nilai persentase *error* sebesar 0.15%. Setelah diketahui persentase *error*-nya, maka dapat dicari akurasinya dengan mengurangi 100% dengan persentase *error*, sehingga didapatkan hasil akurasi sebesar 99.85%. Hasil tersebut termasuk sangat baik karena mendekati nilai 100%.

C. *Quality of Services*

1. *Delay*

Pengujian *delay* dilakukan dengan melihat selisih waktu kirim dan waktu terima data dari *node* sensor ke *server*. Pengujian dilakukan sebanyak sepuluh kali percobaan dengan waktu satu menit untuk setiap percobaan. Adapun hasil dari pengujian *delay* ditampilkan pada Tabel 12.

Tabel 12. Hasil pengujian *delay*

Percobaan ke-n	*Delay* (ms)	Waktu Pengujian
1	156,07	1 menit
2	124,09	
3	169,84	
4	156,73	
5	148,77	
6	152,28	
7	145,54	
8	155,75	
9	120,04	
10	171,83	

Berdasarkan dari hasil pengukuran *delay* yang telah dilakukan, dapat diketahui bahwa nilai *delay* terkecil adalah 120,04 ms, sedangkan nilai terbesar adalah 171,83 ms. Dari

total 10 percobaan yang telah dilakukan, didapatkan rata-rata *delay* dari seluruh percobaan sebesar 150,10 ms. Sehingga dalam pengujian pengiriman data dari *node* sensor ke *server* memiliki *delay* dengan kategori bagus berdasarkan standar TIPHON. Dari hasil pengujian tersebut terdapat beberapa faktor yang mempengaruhi *delay*, yaitu kestabilan koneksi internet dan besaran *bandwidth* yang digunakan.

2. *Packet Loss*

Pengujian *packet loss* dilakukan untuk meng*et*ahui apakah terdapat packet yang hilang saat proses pengiriman data dari *node* sensor ke server atau tidak. Sebanyak sepuluh kali percobaan telah dilakukan dengan waktu satu menit untuk setiap percobaan. Dari semua percobaan tersebut, tidak ditemukan *packet loss* saat pengiriman data. Adapun hasil dari pengujian *packet loss* ditampilkan pada Tabel 13.

Tabel 13. Hasil pengujian *packet loss*

Percobaan ke-n	*Packet loss*	Waktu Pengujian
1	0%	1 menit
2	0%	
3	0%	
4	0%	
5	0%	
6	0%	
7	0%	
8	0%	
9	0%	
10	0%	

Berdasarkan dari hasil pengujian *packet loss* yang telah dilakukan, dapat diketahui bahwa rata-rata nilainya adalah 0% atau tidak ada *packet* yang *loss*. Sehingga berdasarkan standar TIPHON dalam pengujian pengiriman data dari *node* sensor ke server memiliki *packet loss* dengan kategori sangat bagus.

3. *Packet Delivery*

Pengujian ini dilakukan dengan mengukur jumlah paket yang berhasil dikirimkan dari *node* sensor ke *server* dibandingkan dengan jumlah total paket yang dikirimkan. Dari sebanyak sepuluh kali percobaan dengan waktu 1 menit untuk setiap percobaan, diketahui bahwa semua paket yang diterima server berjumlah sama dengan jumlah paket yang dikirim dari *node* sensor. Adapun hasil dari pengujian *packet delivery* ditampilkan pada Tabel 14.

Dari hasil pengujian tersebut dapat diketahui bahwa dari total 10 percobaan, mendapatkan rata -rata nilai *packet delivery* sebesar 100%. Sehingga apabila dikelompokkan berdasarkan kategori *packet delivery* versi TIPHON, maka termasuk ke sangat bagus. Hasil tersebut sudah sesuai apabila dibandingkan dengan hasil *packet loss* sebesar 0% karena semakin tinggi *packet delivery*, maka semakin rendah *packet loss*.

Tabel 14. Hasil pengujian *packet delivery*

Percobaan ke-n	*Packet delivery*	Waktu Pengujian
1	100%	1 menit
2	100%	
3	100%	
4	100%	
5	100%	
6	100%	
7	100%	
8	100%	
9	100%	
10	100%	

Dari pengujian parameter QoS di atas didapatkan hasil bahwa *delay* masuk ke indeks 3 = bagus, *packet loss* masuk ke indeks 4 = sangat bagus, dan *packet delivery* masuk ke indeks 4 = sangat bagus. Apabila dikalkulasikan nilai indeks dari setiap parameter QoS tersebut dan didasarkan pada standar TIPHON, maka pengujian pengiriman data dari *node* sensor ke *server* masuk ke dalam kategori memuaskan.

V. SIMPULAN

Sistem informasi zakat fitrah berbasis IoT berhasil dibangun dan berfungsi dengan baik. Sistem juga memiliki akurasi timbangan yang tinggi mendekati nilai yang sebenarnya sebesar 99,85% dengan tingkat kesalahan yang kecil sebesar 0,15%. Kemudian, pengujian QoS saat pengiriman data dari *node* sensor ke *server* melalui protokol HTTP memiliki kategori memuaskan dengan rincian delay berkategori bagus dengan rata-rata *delay* sebesar 150,10 ms, packet loss berkategori sangat bagus sebesar 0% begitu juga dengan packet delivery berkategori bagus sebesar 100%.

REFERENSI

[1] Word Population Review, "Indonesia Population 2023 (Live)," Word Population Review, 2023. [Online]. Available: https://worldpopulation review.com/countries/indonesia-population. [Diakses 17 Februari 2023].

[2] Menteri Agama Republik Indonesia, Peraturan Menteri Agama Nomor 52 Tahun 2014 Tentang Ketentuan Umum, Jakarta, 2014.

[3] J. Zulhendra, "Tinjauan Hukum Islam Terhadap Zakat Fitrah Dalam Bentuk Uang," *Normative Jurnal Ilmiah Hukum*, vol. 5, pp. 94-105, 2017.

[4] F. Adani dan S. Salsabil, "Internet of Things: Sejarah teknologi dan penerapannya," *Jurnal Online Sekolah Tinggi Teknologi Mandala*, vol. 14, pp. 92-99, 2019.

[5] M. A. Hakam, A. Triayudi dan N. Hayati, "Implementasi Metode Agile pada Sistem Manajemen Zakat Berbasis Website dengan Framework Laravel," *Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)*, vol. 6, pp. 111-116, 2022.

[6] M. DAAIM, "Model Pengelolaan Zakat NU CARE-LAZISNU Tingkat Provinsi Jawa Tengah," *Doctoral dissertation*, IAIN KUDUS, 2020.

[7] IEEE Standards Association, "Internet of Things (IoT) Ecosystem Study," standards.ieee.org, pp. 01-03, 2015.

[8] K. A. Abdurahman, R. Munadi dan A. Indra Irawan, "Perancangan dan Implementasi Hidroponik Berbasis Internet of Things (IoT)

Menggunakan Protokol HTTP," *eProceedings of Engineering*, p. 7(2), 2020.

[9] B. A. Aji, "Pengembangan Website Sistem Pembelajaran Smart Home dengan Konsep Internet of Things Berbasis NodeMCU," Repository UGM, 2021.

[10] N. H. L. Dewi, "Prototype Smart Home dengan Modul Nodemcu ESP8266 Berbasis Internet of Things (IoT)," *Diss*. Universitas Islam Majapahit Mojokerto, 2019.

[11] O. V. Putra, F. R. Pradana, dan M. F. Alfarizqi, "Pengembangan Aplikasi Iot Manajemen Zakat Transaksi Penjualan Dan Pembelian Buah Salak Berbasis Web Menggunakan Metode Prototype," *PROSIDING SNAST*, pp. C 89-98, 2021.

[12] M. F. A. Safii, S. Raharjo, dan U. Lestari, "Analisis Quality of Service protokol MQTT dan HTTP pada penerapan sistem monitoring suhu berbasis NodeMCU (studi kasus ruang server kampus 3 IST Akprind Yogyakarta)," *Jurnal Jarkom*, pp. 11-19, 2019.

[13] TIPHON, "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) ; General aspects of Quality of Service (QoS).," etsi.org, 1999.

[14] N. Nikolov, "Research of MQTT, CoAP, HTTP and XMPP IoT," Proc. XXIX International Scientific Conference Electronics - ET2020, 2020.

[15] R. SUSANA, A. NUGRAHA dan D. NATALIANA, "Perancangan dan Realisasi Web-Based Data Logging System menggunakan ATmega16 melalui Hypertext Transfer Protocol (HTTP)," *ELKOMIKA: Jurnal Teknik Energi* Elektrik, Teknik Telekomunikasi, & Teknik Elektronika, vol. 3, pp. 01-15, 2015.

Analisis Kinerja Algoritma *Weighted Round Robin* dan *Weighted Least Connection* pada Sistem *Load Balancing Web Server* dengan HAProxy Terintegrasi *Cacti Monitoring*

Teuku Muhammad Fathin Rifat[1], Unan Yusmaniar Oktiawati[1,*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
tmfathin02@mail.ugm.ac.id
[*]Korespondensi: unan_yusmaniar@ugm.ac.id;

Abstract – The rapid advancement of technology over time has led to the high use of the internet in everyday life. The need for the internet affects website visitors who are increasing and makes the traffic load increase on the server. The more the amount of traffic to the server can cause the server to go down, if used with excessive amounts (overload). Therefore, it is necessary to improve the quality of the network that is directly related to the server as the role of a network traffic. So using a load balancing system is a solution to overcome the occurrence of server down. In the application process in this research, to overcome this problem, Nginx, HAproxy and a virtual machine system using virtualbox tools are needed for this. The Load Balancing method uses two servers and three servers, where this method will divert the traffic load from a full web server to another web server according to the number of web servers used. And will be monitored using Cacti by displaying graphical data from load balancing performance on the web server. The results show that the implementation of HAProxy on a web server load balancing system that uses the Weighted Round Robin and Weighted Least Connection algorithms can overcome problems caused by excessive traffic loads. The Weighted Least Connection algorithm provides superior values on the parameters of throughput, response time, request error, cpu usage, and memory usage compared to the Weighted Round Robin algorithm on the web server load balancing system.

Keywords – Web Server , Load Balancing, HAProxy, Cacti Monitoring

Intisari – Kemajuan teknologi yang semakin pesat dari waktu ke waktu menyebabkan tingginya penggunaan internet dalam kehidupan sehari-hari. Kebutuhan internet mempengaruhi pengunjung *website* yang semakin meningkat dan membuat beban *traffic* meningkat pada *server*. Semakin banyak jumlah *traffic* menuju *server* dapat menyebabkan *server* menjadi *down* jika digunakan dengan jumlah berlebih (*overload*). Oleh karena itu, dibutuhkan suatu peningkatan kualitas jaringan yang berhubungan langsung dengan *server* sebagai peran dari suatu lalu lintas jaringan. Penggunaan sistem *load balancing* merupakan solusi untuk mengatasi terjadinya *server down*. Penelitian ini memerlukan Nginx, HAproxy, dan sistem *virtual machine* menggunakan tools Virtualbox. Metode *load balancing* menggunakan dua *server* dan tiga *server*. Kedua metode ini akan mengalihkan beban *traffic* dari *web server* yang sudah penuh kepada *web server* lainnya sesuai dengan jumlah *web server* yang digunakan. Kemudian, selanjutnya akan dimonitoring menggunakan Cacti dengan menampilkan data grafik dari kinerja *load balancing* pada *web server*. Hasil menunjukkan dengan implementasi HAProxy pada sistem *load balancing web server* yang menggunakan algoritma *Weighted Round Robin* dan *Weighted Least Connection* dapat mengatasi permasalahan yang diakibatkan oleh beban *traffic* yang berlebihan. Algoritma *Weighted Least Connection* memberikan nilai yang lebih unggul pada parameter *throughput, response time, error request,* CPU *usage,* dan *memory usage* dibandingkan algoritma *Weighted Round Robin* pada sistem *load balancing web server*.

Kata kunci – *Web Server*, *Load Balancing*, HAProxy, *Cacti Monitoring*

I. PENDAHULUAN

Seiring berkembangnya teknologi komunikasi dan informasi, terutama dalam teknologi jaringan sangat berkaitan erat dengan teknologi internet yang menjadi salah satu sumber penyedia informasi terpopuler di era modern ini. Hampir seluruh sektor yang berada di sekitar telah memanfaatkan teknologi informasi yang menggunakan internet dalam membantu pekerjaan sehari-hari ataupun sekadar untuk hiburan semata. Perkembangan pengguna internet berpengaruh terhadap peningkatan data yang ditransmisikan. Paradigma di zaman sekarang telah berubah, salah satunya pada sektor hiburan dan informasi. Pada zaman dulu, sarana informasi berbentuk fisik memegang peranan penting dalam mendapatkan informasi terbaru. Saat ini hampir seluruh informasi yang dibutuhkan oleh khalayak umum baik informasi terdahulu hingga terkini, telah dibuat dan ditransmisikan ke dalam bentuk informasi digital yang dapat diakses kapan saja dan di mana saja dengan mudah. Tidak hanya pada sektor hiburan dan informasi, sektor perekonomian juga sangat memerlukan teknologi dalam mendongrak penjualan dan pendapatan. Terlebih dalam beberapa tahun belakangan, dunia dilanda wabah virus corona yang menyebabkan seluruh aktivitas di luar ruangan dibatasi. Salah satu yang memperoleh dampaknya adalah aktivitas berdagang. Hal tersebut menyebabkan para pedagang mengalami kerugian. Dengan demikian, untuk memenuhi target penjualan dan mendapatkan keuntungan, para pedagang beralih menggunakan teknologi untuk melanjutkan proses berdagang baik dengan membangun *website e-commerce* pribadi atau bergabung dengan *startup e-commerce*. Berdasarkan data pada Statista Market Insight, jumlah pengguna *e-commerce* baik secara *website* maupun aplikasi akan diproyeksikan mencapai 196,477 juta pengguna hingga akhir tahun 2023 [1].

Dalam mendukung pemenuhan kebutuhan untuk mendapatkan informasi terkini dan proses jual beli secara *online* diperlukan akses yang cepat dan stabil, salah satu aspek yang mempengaruhi kecepatan akses dan kestabilan sebuah alamat *website*, yaitu *server* penyedia layanan. Perangkat *server* atau disebut juga sebagai penyedia layanan merupakan sistem komputer yang memiliki layanan khusus berupa pengelolaan dan penyimpanan data. Perangkat *server* memiliki peran penting dalam menyediakan layanan akses yang lebih cepat untuk proses mengirim atau menerima data yang tersedia pada perangkat *server*. Apabila klien semakin banyak maka beban kerja kerja *server* semakin berat, sehingga diperlukan spesifikasi *server* yang bagus dan dapat melayani berbagai permintaan klien. Instansi perusahaan dan pengusaha individu yang menggunakan sistem informasi berbasis *website* dalam mengoperasikan segala kegiatan bisnisnya, membutuhkan *server* dengan spesifikasi yang cukup mumpuni, khususnya ketika banyak klien yang mengakses. Perangkat *server* dengan spesifikasi tinggi sangat diperlukan dalam membangun *webserver*, sehingga dapat memberikan layanan yang cepat dan stabil kepada klien ketika terdapat pengakses berjumlah banyak. Namun, biaya yang diperlukan dalam memenuhi kebutuhan tersebut tergolong tidak murah. Perangkat *server* tunggal yang memiliki spesifikasi cukup tinggi sangat rawan mengalami *trouble* dan *down* ketika mengalami lonjatan *request* sangat besar sehingga menyebabkan *website* tidak dapat diakses oleh pengunjung.

Penggabungan beberapa *server* menjadi satu kesatuan yang bekerja secara bersamaan (pemerataan beban *server*) merupakan salah satu solusi dalam mengatasi permasalahan diatas. Salah satu metode yang dapat digunakan yaitu *load balancing*. *Load balancing* merupakan suatu teknik dalam mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara optimal, memaksimalkan *throughput*, memperkecil *response time*, dan menghindari terjadinya *overload* pada salah satu *server* [2], [3]. Tujuan dari *Load balancing web server,* yaitu meringankan beban yang ditanggung oleh setiap *server*, sehingga dapat meningkatkan kinerja *server* yang *high availability* dan terjaga meskipun salah satu *server* mengalami *down* dan tidak dapat melayani *request* dari klien, dikarenakan *server* lain secara otomatis akan menggantikannya. Salah satu aplikasi yang dapat digunakan untuk sistem *load balancing web server* adalah HAProxy. HAProxy atau *High Availibility Proxy* merupakan perangkat lunak *open-source* yang dikembangkan khusus untuk melakukan *load balancing* dan biasanya digunakan dalam penanganan lalu lintas HTTP dan TCP yang sangat cepat, seperti aplikasi web dengan tingkat lalu lintas tinggi dan telah digunakan oleh *website* terkenal, seperti Github, Stackoverfow, Reddit, Tumblr, Instagram, dan Twitter [4], [5]. Terdapat beberapa algoritma *load balancing* pada HAProxy, antara lain *Weighted Round Robin* dan *Weighted Least Connection*. Algoritma *Weighted Round Robin* merupakan perbaikan dari algoritma *Round Robin* yang mendistribusikan beban secara seimbang menuju seluruh *server* tanpa memperhatikan kapasitas proses yang berbeda. Masing-masing *server* diberi bobot dengan bilangan *interger* yang menunjukkan kapasitas proses, di mana bobot awal adalah 1. Algoritma *Weighted Least Connection* mengarahkan koneksi dari jaringan ke *server* yang mempunyai jumlah koneksi aktif paling sedikit. Algoritma ini merupakan algoritma dinamis karena menghitung jumlah koneksi yang aktif pada setiap *server* secara dinamis dan algoritma ini cukup baik untuk memperhalus distribusi koneksi saat *request* sangat beragam [6], [7].

Berdasarkan hal di atas, penelitian ini melakukan "Analisis Kinerja Algoritma *Weighted Round Robin* dan *Weighted Least Connection* pada Sistem *Load Balancing Web Server* dengan HAProxy Terintegrasi *Cacti Monitoring*". Penelitian ini menggunakan dua buah algoritma, yaitu *Weighted Round Robin* dan *Weighted Least Connection*. Tujuan penelitian ini untuk mengetahui kinerja dari sistem *load balancing* yang menggunakan dua buah algoritma tersebut, kemudian menemukan algoritma mana yang memberikan hasil terbaik dari segi nilai *throughput*, *response time*, dan *error request* serta memonitoring dari kinerja *load balancing* dengan *Cacti Monitoring*.

II. DASAR TEORI

Suatu penelitian diperlukan adanya hasil dari beberapa penelitian sebelumnya sebagai landasan teori dan acuan dalam penelitian ini. Penelitian [8] merancang sistem *load balancing* pada *web server* menggunakan metode *apache* yang diimplementasikan sebagai *load balancing* serta sebagai *web server* yang memiliki tujuan dalam meningkatkan kinerja *web server*. Pada sistem *load balancing*-nya, mengimplementasikan teknik *reverse proxy* dengan konfigurasi yang diberikan berupa *method by busyness* dan *method by request*. Skenario pengujian dibagi menjadi tiga skenario pengujian, yaitu tes terhadap koneksi antara *database server* dengan *web server,* tes sistem kerja *load balancer method by request,* tes sistem kerja *load balancer method by busyness*. Penelitian ini menyatakan bahwa *sistem load balancing* metode *apache* dengan *method by request* akan melanjutkan seluruh *request* yang masuk menuju ke beberapa *web server* secara berurutan, dan *load balancing* metode *apache* dengan *method by busyness* akan mengarahkan menuju *web server* yang memiliki waktu respon tinggi.

Selain dengan Apache*, load balancing* dapat diterapkan dengan menggunakan Nginx yang lebih ringan dan menjadi *server* khusus *reverse proxy* tanpa membebani *hardware* yang digunakan sebagai *load balancer* [8], [9]. Penelitian [10] menggunakan layanan Nginx sebagai *load balancing web server* yang dibangun pada *virtual machine* VirtualBox. Pengujian dilakukan dengan menggunakan serangan kiriman paket secara simultan pada waktu beberapa detik dengan *tools siege.* Hasil menunjukkan bahwa dengan menggunakan Nginx untuk *load balancing web server* dapat membantu *web server* mengalihkan *request* dari *client* ke beberapa *web*

server secara bergantian ketika terjadi *overload*, tetapi secara manual.

Penelitian [11] merancang implementasi *load balancing* dengan tambahan metode NTH. Penelitian ini memiliki infrastruktur menggunakan satu komputer *server*, Mikrotik RB941-2nD-TC, dua komputer *client*, satu *switch,* dua NIC dengan dukungan *software ubuntu server, web server, Openssh Server, Winbox,* dan *WinSCP.* Peneliti menggunakan metode *load balancing* NTH yang didukung dengan NIC sebagai jalur dalam pembagi beban yang ditanggung. Hasil dari penelitian berikut menunjukkan bahwa dengan menggunakan metode *load balancing NTH* berhasil yang ditandai dengan keseimbangan pada proses *download file* dan pada kecepatan *download 2 client* sama besar yakni 256 kBps sesuai dengan konfigurasi *bandwidth* yang telah dibuat yaitu sebesar 512 kBps.

Penelitian selanjutnya menggunakan metode *clustering* server untuk proses *load balancing*. Menerapkan *clustering server* pada sistem *load balancing* dapat meningkatkan kinerja seluruh *server* dan mendukung sinkronisasi file-file yang disimpan pada seluruh *server* [12]. *Server* yang digunakan sebagai *web server* dibuat menjadi sebuah *cluster* dengan menggunakan fungsi *failover cluster* dan *load balancing cluster* yang memungkinkan *server-server* tersebut berkolaborasi dan saling bekerja sama dalam menciptakan kinerja yang maksimal, sehingga jika salah satu *server* mengalami masalah, *server* lain dapat mengambil alih tugasnya dan menjaga kualitas layanan. Hasil dari penelitian tersebut menunjukkan bahwa dengan menggunakan *clustering server* performa yang dihasilkan untuk layanan sangat baik dan berjalan sesuai harapan, terutama pada sinkronisasi file. Dengan konfigurasi yang dirancang sedemikian rupa dan juga memanfaatkan *clustering server*, file-file dapat tersimpan otomatis pada ketiga *server* meskipun proses penyimpanan file hanya dilakukan pada salah satu *server*.

Penelitian [13] merancang sistem *load balancing* terhadap *web server* dengan menggunakan Nginx. Sistem yang dirancang menggunakan sebagai *web server* dan *load balancer* serta menerapkan algoritma *round robin*, algoritma *least connection*, dan algoritma *IP hash* pada sistem *load balancing.* Penelitian ini melakukan pengujian fungsionalitas untuk mengetahui kemampuan sistem yang dirancang dalam mengatasi masalah yang ada dan untuk mengetahui algoritma mana yang dapat memberikan hasil terbaik pada sistem *load balancing* di antara ketiga algoritma yang digunakan.

Penelitian [14] memiliki tujuan untuk menguji bagaimana *load balancing* dengan HAProxy yang menggunakan algoritma *round robin* dapat memberikan kinerja terbaik dalam mengatasi masalah ketika salah satu *server down* atau mengalami *trouble*. Penelitian ini menyatakan bahwa penerapan sistem *load balancing* menggunakan HAProxy dapat meningkatkan ketersediaan *server* (*uptime*) pada suatu *website* dan memberikan kualitas yang terjaga dan stabil ketika salah satu *server* mengalami *down*, sehingga *user* tidak merasakan terjadi *down time*.

Penelitian [15] mengimplementasikan dua buah metode *load balancing* pada pengujian terhadap sistem *e-learning* yang dibandingkan nilainya dari beberapa aspek pengujian, seperti nilai *throughput* dan *response time*. Hasil dari penelitian ini menunjukkan bahwa *load balancing* yang mengimplementasikan metode HAProxy memberikan kinerja yang sangat baik dibandingkan dengan *load balancing* yang mengimplementasikan metode Nginx dengan hasil pada nilai *response time* dan *throughput* HAProxy meraih hasil yang sangat baik sebesar 533,14 ms untuk *response time* dan 40,84 kB/sec untuk *throughput*.

Penelitian yang [16] memiliki tujuan untuk menguji bagaimana algoritma penjadwalan *Weighted Least Connection* yang telah diimprovisasi dapat menjawab kebutuhan *load balancing* pada *web cluster system* dengan lebih baik lagi dibanding dengan algoritma *Weighted Least Connection* sebelum diimprovisasi. Penelitian ini menggunakan dua buah protokol, yaitu *server selection* dan *overload* dengan tujuan membantu proses *load balancing* terhadap *web cluster system* yang menggunakan *Weighted Least Connection* sebagai algoritma pada sistem *load balancing*. Hasil penelitian ini menunjukkan bahwa hasil improvisasi dari penggunaan *Weighted Least Connection* yang menerapkan dua buah protocol baru mampu menghasilkan kinerja yang lebih baik pada sistem *load balancing* terhadap *web cluster.*

Penelitian [17] memiliki tujuan untuk menguji bagaimana kinerja algoritma *least connection* dan IP *hash* pada sistem *load balancing web server* yang melalui jaringan SDN. Kedua algoritma yang digunakan pada penelitian ini akan dievaluasi dengan beberapa parameter dasar untuk *web server*. Hasil penelitian ini menunjukkan bahwa algoritma IP *hash* menghasilkan kinerja yang baik, dengan hasil pada parameter *response time* sebesar 61 ms, hasil pada *throughput* sebesar 1,23 Mbps, dan hasil pada *resource memory* sebesar 189,2 MB.

Penelitian sebelumnya masih menggunakan Nginx, HAProxy, dan Apache sebagai metode *load balancing*. Selain dengan metode tersebut, *load balancing* juga dapat menggunakan metode Traefik and Envoy. Penelitian [18] menggunakan tiga buah metode, yaitu HAProxy, Nginx, dan Traefik and Envoy untuk proses *load balancing* pada *web server*. Hasil dari penelitian ini menunjukkan bahwa metode Traefik and Envoy memiliki kinerja yang cukup bagus, lebih bagus dari pada Nginx, tetapi tidak sebagus HAProxy, dengan data menunjukkan bahwa *traffic* berkinerja 24,1% lebih baik, *envoy* 26,9% lebih baik, dan HAProxy 36,0% lebih baik dibandingkan Nginx.

III. METODOLOGI

A. Alat

Alat yang digunakan pada penelitian ini berupa laptop dan *Virtual Machine* memiliki spesifikasi seperti pada Tabel 1 dan Tabel 2.

- Laptop

Tabel 1. Spesifikasi Laptop

Spesifikasi	Detail
Sistem Operasi	Windows 11 Home Single Language 64-bit
Processor	11th Gen i7-11800H 2.30GHz
Memory	16GB RAM
GPU	NVIDIA GeForce RTX 3050Ti
Penyimpanan Sistem	512GB SSD NVME M.2

- *Virtual Machine*

Tabel 2. Spesifikasi *Virtual Machine*

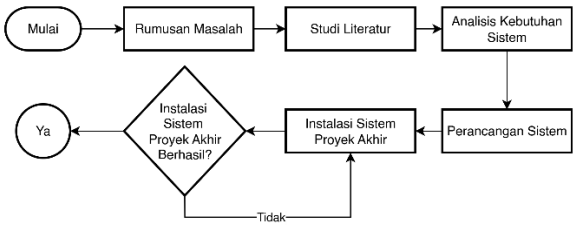
Virtual Machine	Spesifikasi	
Load Balancer	OS	Debian 11.6
	RAM	4 GB
	Disk	80 GB
	Alamat IP	
	Nama VM	Load Balancer
Web Server 1	OS	Debian 11.6
	RAM	2 GB
	Disk	50 GB
	Alamat IP	
	Nama VM	Nginx Web Server
Web Server 2	OS	Debian 11.6
	RAM	2 GB
	Disk	50 GB
	Alamat IP	
	Nama VM	Nginx Web Server
Web Server 3	OS	Debian 11.6
	RAM	2 GB
	Disk	50 GB
	Alamat IP	
	Nama VM	Nginx Web Server

B. Bahan

Dalam penelitian ini menggunakan beberapa bahan seperti *Virtual Machine* (VirtualBox), Windows OS, Web Browser.

C. Diagram Alir Penelitian

Secara ringkas alur penelitian secara singkat dapat dilihat pada Gambar 1.



Gambar 1. Diagram Alir Penelitian

D. Pengujian

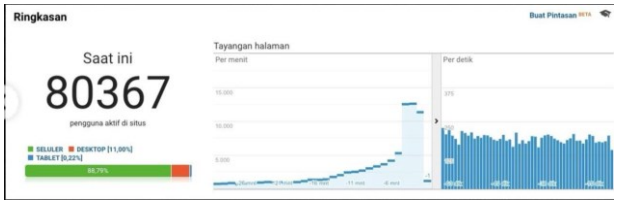
Pengujian pada penelitian berikut ini dilakukan dengan menguji sistem kinerja dari *load balancing* HAProxy terhadap *webserver* yang menggunakan dua buah algoritma, yaitu *Weighted Round Robin* dan *Weighted Least Connection*. Penggunaan dua buah algoritma memiliki tujuan untuk melihat algoritma mana yang dapat memberikan kualitas terbaik pada sistem *load balancing* terhadap *webserver*. Dalam pengujian berikut, menggunakan dua buah variasi jumlah *server* dengan pengujian pertama menggunakan tiga buah *server* dan pengujian kedua menggunakan dua buah *server.* Hal tersebut bertujuan agar dapat mengidentifikasi terhadap kemampuan sistem *load balancing* dalam menangani lalu lintas dengan baik ketika terdapat lebih banyak atau lebih sedikit sumber daya *server* yang tersedia serta memastikan bahwa sistem *load balancing* dan infrastruktur web dapat beroperasi secara optimal dalam berbagai kondisi yang mungkin terjadi di lingkungan produksi.

Pengujian berikut akan mengirimkan *request* atau permintaan dengan dua buah variasi yang terdiri dari dua buah jumlah *request*, pertama dengan jumlah terendah sebesar 7.500 *request* dan kedua dengan jumlah terbesar sebesar 10.000 *request*. Jumlah *request* ditetapkan sebanyak dua buah dengan tujuan untuk menguji performa dari infrastruktur dalam menghadapi lonjakan lalu lintas atau beban yang tidak terduga baik dengan jumlah *request* yang sangat sedikit maupun dengan jumlah *request* yang sangat banyak.

Selain itu, hal tersebut juga bertujuan untuk lebih memahami secara lebih dari kemampuan sistem, mengidentifikasi potensi *bottleneck* atau masalah kinerja, serta memastikan bahwa infrastruktur web dapat mengatasi beban lalu lintas yang bervariasi dengan baik dalam lingkungan produksi yang sebenarnya. Waktu yang digunakan pada proses pengiriman seluruh jumlah, jumlah *request* sebesar 7.500 dan jumlah *request* sebesar 10.000 menggunakan waktu selama 10s agar pemantauan yang dilakukan sesuai dengan keadaan yang terjadi.

Hal ini dikarenakan *use case* yang digunakan adalah *website* brand lokal ketika *launching* produk *limited edition* terbaru di mana *client* yang mengakses ketika ingin membeli produk tersebut berkisar 300 *client* per detik. Pengambilan jumlah *request* yang diujikan berdasarkan contoh pada suatu *website brand* pakaian lokal ketika merilis suatu produk

terbaru yang hanya memiliki stok terbatas, seperti yang tertera pada Gambar 2.



Gambar 2. Jumlah *Client Website Brand Heymale*

IV. HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan analisis kinerja dari dua buah algoritma *load balancing* yang digunakan. Dengan membandingkan dua buah algoritma memberikan gambaran kepada *user* yang menggunakan *load balancing* baik untuk kegunaan pribadi maupun untuk perusahaan. Agar sistem yang digunakan dapat berjalan dengan lancar dan mengatasi masalah, terutama mengenai masalah *server* yang *down* ketika *overload traffic*, pengujian performa dilakukan dengan beberapa konfigurasi baik pada sisi jumlah *server* yang hidup maupun pada sisi algoritma *load balancing* dengan *request* atau permintaan yang akan dikirimkan pada pengujian berjumlah 7.500 *request* dan 10.000 request dan berlaku pada setiap pengujian.

A. Analisis Hasil Pengujian WRR dan WLC dengan Tiga Buah *Webserver*

Pada pengujian ini dilakukan dengan tiga buah konfigurasi *weight* pada setiap *webserver* yang mana setiap *webserver* memiliki jumlah *weight* yang berbeda. Konfigurasi *weight* pertama memiliki perbandingan 3 : 1 : 2, dimana *webserver* pertama memiliki bobot *weight* yang lebih besar agar dapat menerima lebih banyak beban, sedangkan pada *webserver* kedua memiliki bobot *weight* yang lebih kecil agar beban yang diterima tidak telalu besar, dan pada *webserver* ketiga memiliki bobot *weight* yang tidak terlalu besar dan tidak terlalu kecil atau berada diantara beban *webserver* pertama dan *webserver* kedua sehingga kinerja beban yang diterima tidak terlalu berat dan tidak terlalu ringan. Pada konfigurasi *weight* kedua memiliki perbandingan 2 : 5 : 3, dimana *webserver* kedua memiliki bobot yang lebih besar, *webserver* pertama memiliki bobot yang lebih kecil, dan *webserver* ketiga memiliki jumlah bobot di antara *webserver* pertama dan *webserver* kedua. Pada konfigurasi *weight* ketiga memiliki perbandingan 2 : 1 : 3, dimana *webserver* ketiga memiliki bobot yang lebih besar dibandingkan dengan kedua *webserver* lainnya. Jumlah *request* yang akan dikirimkan menuju *load balancer* sesuai yang telah ditetapkan dengan dua buah *request*, yaitu 7.500 *request* dan 10.000 *request*. Hasil yang didapatkan ditampilkan dalam bentuk tabel.

1. Analisis Hasil Nilai Pengujian *Throughput*

Pengujian nilai *throughput* pada penelitian berikut dilakukan dengan menjalankan beberapa penelitian yang terdiri dari penelitian menggunakan *Weighted Round Robin* dan *Weighted Least Connection*. Hasil Data didapatkan dari proses pengujian dengan tools JMeter. Untuk nilai hasil pengujian *throughput* pada penelitian berikut dapat terlihat pada Tabel 3.

Tabel 3. *Throughput* Tiga *Web Server*

Banyak *request*	*Throughput*					
	Weighted Round Robin			*Weighted Least Connection*		
	3:1:2	**2:5:3**	**2:1:3**	**3:1:2**	**2:5:3**	**2:1:3**
7.500	1059,8	997,9	998,5	999,5	1.000	999,1
10.000	997,7	870,1	999,1	1018,8	995,6	1.000

Berdasarkan tabel hasil pengujian nilai *throughput* menunjukkan bahwa nilai *throughput* yang didapatkan dari penggunaan *weighted least connection* memiliki kinerja yang lebih baik dibandingkan *throughput* yang dihasilkan oleh *weighted round robin* pada tiga buah pengujian dengan nilai yang didapatkan oleh WLC sebesar 1018,8 Kbits/s, 1.000 Kbits/sec, dan 995,6 Kbits/s. Namun, pada salah satu pengujian algoritma WRR mendapatkan hasil lebih baik dibanding WLC dengan nilai sebesar 1059,8 Kbits/sec. Berdasarkan data pengujian *throughput* yang dihasilkan, algoritma *weighted least connection* mampu mengalokasikan jumlah permintaan atau jumlah koneksi secara baik dan mampu menghindari terjadinya *overload* permintaan atau *request* pada satu atau beberapa *webserver* yang ada.

2. Analisis Hasil Nilai Pengujian *Response Time*

Pengujian *Response Time* dilakukan dengan tujuan mengetahui besaran kecepatan respons *server* dalam menanggapi *request* klien. Pengujian dilakukan menggunakan JMeter dengan jumlah *request* sebesar 7500 dan 10000. Untuk nilai hasil pengujian *response time* pada penelitian ini dapat terlihat pada Tabel 4.

Tabel 4. *Response Time 3 Web Server*

Banyak *request*	*Throughput*					
	Weighted Round Robin			*Weighted Least Connection*		
	3:1:2	**2:5:3**	**2:1:3**	**3:1:2**	**2:5:3**	**2:1:3**
7.500	63	5	12	8	5	9
10.000	29	12	61	14	9	8

Berdasarkan data pada hasil pengujian *response time* yang tertera pada tabel menunjukkan bahwa *response time* yang didapatkan oleh *weighted least connection* memiliki kinerja yang lebih baik dan cepat dibandingkan dengan *Weighted Round Robin*. Dengan rata-rata yang dihasilkan ketika menggunakan *weighted least connection* sebesar 9ms, algoritma *Weighted Least Connection* memiliki *response time* lebih baik, dikarenakan *request* dapat diteruskan ke *server* dengan optimal sesuai bobot masing-masing *server* dan dialihkan menuju *server* yang memiliki koneksi sedikit sehingga beban kerja *server* menjadi lebih merata. *Response*

time yang baik juga didukung oleh jumlah *request error* yang terjadi sangat sedikit.

3. Analisis Hasil Nilai Pengujian *Error Request*

Pengujian nilai *Error Request* dilakukan dengan tujuan untuk melihat algoritma yang mana memiliki performa lebih baik dalam mendistribusikan *request* menuju *server-server* yang ada. Data pengujian diambil dengan beberapa pengujian yang menggunakan tools JMeter. Untuk nilai dari hasil pengujian *error request* dapat terlihat pada Tabel 5.

Tabel 5. *Error Request 3 Web Server*

Banyak *request*	*Error Request (%)*					
	Weighted Round Robin			*Weighted Least Connection*		
	3:1:2	2:5:3	2:1:3	3:1:2	2:5:3	2:1:3
7.500	0	0	0	0	0	0
10.000	27,23	5,55	28,19	0,05	12,47	0

Berdasarkan tabel hasil pengujian *error request* menunjukkan bahwa algoritma *weighted least connection* menghasilkan performa yang sangat baik dalam menimalisir *error* yang terjadi. Adanya *Error* pada penggunaan *weighted least connection* lebih sedikit dibanding *error* yang terjadi pada penggunaan *weighted round robin,* seperti yang tertera pada tabel hasil. Rata-rata yang dihasilkan oleh *weighted least connection* pada pengujian dengan konfigurasi *weight* pertama lebih sedikit dibanding *weighted round robin* dan selisih rata-rata antara keduanya sebesar 25,98%. Dengan hasil tersebut menujukkan bahwa *weighted least connection* lebih baik dalam hal mendistribusikan atau membagi *request* yang datang sehingga *error* yang terjadi dapat diminimalisir.

B. Analisis Hasil Pengujian *Weighted Round Robin* dengan Dua Buah *Webserver*

Berbeda dengan pengujian pertama, pengujian kedua dilakukan dengan mematikan salah satu *webserver* agar dapat melihat dan memahami lebih lagi terkait kinerja apabila hanya memiliki dua buah *webserver* atau pada suatu kondisi salah satu *webserver* tiba-tiba *down* dan hanya memiliki beberapa *webserver* yang tersedia. Konfigurasi yang diberikan pada dua buah *webserver* juga tidak jauh dari sebelumnya, hanya berbeda pada jumlah beban atau *weight* pada *webserver* yang digunakan. Konfigurasi *weight* pertama memiliki perbandingan 3 : 1, di mana *webserver* pertama memiliki bobot *weight* yang lebih besar agar dapat menerima lebih banyak beban, sedangkan pada *webserver* kedua memiliki bobot *weight* yang lebih kecil agar beban yang diterima tidak terlalu besar atau lebih kecil dibanding *webserver* pertama. Pada konfigurasi *weight* kedua memiliki perbandingan 1 : 2, dimana *webserver* kedua memiliki bobot yang lebih besar, dan *webserver* pertama memiliki bobot yang lebih kecil.

1. Analisis Hasil Nilai Pengujian *Throughput*

Data hasil pengujian nilai *throughput* yang diambil pada penelitian ini dilakukan dengan menjalankan beberapa penelitian yang terdiri dari penelitian menggunakan *weighted round robin* dan *weighted least connection*. Data yang didapatkan dari proses pengujian dengan tools JMeter. Terkait nilai dari hasil pengujian *throughput* pada penelitian ini dapat terlihat pada Tabel 6.

Tabel 6. *Throughput 2 Web Server*

Banyak *Request*	*Throughput*			
	Weighted Round Robin		*Weighted Least Connection*	
	3:1	1:2	3:1	1:2
7.500	987,7	999,6	999,9	999,7
10.000	870,1	997,4	1.000	998,3

Berdasarkan data dari hasil pengujian penggunaan *throughput* yang tertera pada tabel menunjukkan bahwa *throughput* yang dihasilkan dari penggunaan *weighted least connection* pada pengujian dengan dua buah *webserver* memiliki performa yang baik dibandingkan *throughput* yang dihasilkan oleh algoritma *Weighted Round Robin* meskipun dengan selisih diantara setiap pengujian sebesar 0,01%. Berdasarkan data pengujian *throughput* yang dihasilkan bahwa algoritma *weighted least connection* dan *Weighted Round Robin* mampu mengalokasikan jumlah permintaan atau jumlah koneksi secara baik dan mampu menghindari terjadinya *overload* permintaan atau *request* pada satu atau beberapa *webserver* yang ada. Meskipun *weighted least connection* memiliki keunggulan sebesar 0,01% pada setiap pengujian.

2. Analisis Hasil Nilai Pengujian *Response Time*

Pengujian *Response Time* dilakukan dengan tujuan mengetahui besaran kecepatan *response server* dalam menanggapi *request* klien. Pengujian dilakukan menggunakan JMeter dengan jumlah *request* sebesar 7.500 dan 10.000. Hasil dari pengujian nilai *response time* pada penelitian ini dapat terlihat pada Tabel 7.

Tabel 7. *Response Time 2 Web Server*

Banyak *Request*	*Response Time (ms)*			
	Weighted Round Robin		*Weighted Least Connection*	
	3:1	1:2	3:1	1:2
7.500	7	6	6	6
10.000	9	9	9	8

Berdasarkan data dari hasil pengujian nilai *response time* yang tertera pada tabel menunjukkan bahwa nilai *response time* yang didapatkan oleh sistem *load balancing* dengan menerapkan *weighted least connection* dan *weighted round*

robin memiliki kinerja yang cukup baik pada pengujian terhadap dua buah *webserver* dengan hasil kedua nya hampir sama baik. Akan tetapi, secara keseluruhan *weighted least connection* memiliki hasil yang lebih unggul dibanding *weighted round robin*. *Weighted Least Connection* memiliki nilai *response time* lebih baik dan lebih cepat sebesar 1ms dibanding *weighted round robin* pada dua buah pengujian yang dilakukan.

3. Analisis Hasil Nilai Pengujian *Error Request*

Pengujian nilai *Error Request* dilakukan untuk melihat algoritma yang mana memiliki performa lebih baik dalam mendistribusikan *request* menuju *server-server* yang ada. Data pengujian diambil dengan beberapa pengujian yang menggunakan tools JMeter. Untuk nilai dari hasil pengujian nilai *error request* pada penelitian berikut dapat dilihat pada Tabel 8.

Tabel 8. *Error Request 2 Web Server*

Banyak *Request*	***Error Request (%)***			
	Weighted Round Robin		***Weighted Least Connection***	
	3:1	**1:2**	**3:1**	**1:2**
7.500	0	0	0	0
10.000	0	0,05	0	0,01

Berdasarkan tabel hasil pengujian *error request* menunjukkan bahwa algoritma *weighted least connection* pada pengujian dengan dua *webserver* menghasilkan performa yang sangat baik dalam menimalisir *error* yang terjadi dengan rata-rata error sebesar 0,01%. *Error* yang terjadi pada penggunaan *weighted least connection* lebih sedikit dibanding *error* yang terjadi pada pengunaan *weighted round robin,* seperti yang tertera pada tabel hasil. Selisih dari rata-rata yang dihasilkan oleh kedua algoritma tidak terlalu jauh dengan algoritma *weighted least connection* lebih unggul. Dengan hasil tersebut menunjukkan bahwa *weighted least connection* lebih baik dalam hal mendistribusikan atau membagi *request* yang datang sehingga *error* yang terjadi dapat diminimalisir.

C. Analisis Hasil Pengujian dengan Skema Lonjakan beban dan Pola Acak

Pengujian berikut menggunakan skema lonjakan beban dan pola acak dilakukan untuk mengetahui besaran nilai *response time* dan *error request* pada system load balancing dengan menggunakan algoritma *weighted round robin* dan *weighted least connection* dalam menghadapi kondisi yang sering terjadi di dunia nyata. Hasil dari pengujian nilai *response time* dan *error request* pada penelitian ini dapat terlihat pada Tabel 9 dan Tabel 10.

Tabel 9. *Response Time* dan *Error Request* Lonjakan Beban

Skema Lonjakan Beban		
Algoritma	***Response Time***	***Error Request***
Weighted Round Robin	490 ms	5,2%
Weighted Least Connection	260 ms	1,1%

Tabel 10. *Response Time* dan *Error Request* Pola Acak

Skema Pola Acak		
Algoritma	***Response Time***	***Error Request***
Weighted Round Robin	310 ms	3,5%
Weighted Least Connection	190 ms	0,7%

Berdasarkan hasil yang tertera pada Tabel 9 dan Tabel 10 menunjukkan bahwa algoritma *weighted least connection* mampu menghasilkan performa lebih baik pada pengujian dengan skema lonjakan beban maupun pola acak. Dengan selisih terhadap algoritma *weighted round robin* sebesar 150 ms hingga 250 ms untuk nilai *response time*-nya, sedangkan untuk selisih pada parameter *error request* sebesar 3% hingga 4%. Algoritma *weighted least connection* mendapatkan hasil performa seperti pada tabel di atas, dikarenakan sistem pembagian bebannya mempertimbangkan jumlah koneksi aktif pada setiap *server*. *Server* yang memiliki jumlah koneksi aktif sedikit atau kecil akan dikirimkan *load* beban yang banyak, sedangkan *server* yang memiliki jumlah koneksi aktif besar akan dikirimkan *load* beban yang sedikit, dan hal tersebut membuat distribusi pembagian beban lebih seimbang.

D. Analisis Hasil Pengujian *CPU Usage* dan *Memory Usage*

Data hasil pengujian *Memory Usage* diambil dengan melakukan dua buah pengujian dengan mengaktifkan *proxy* dan menonaktifkan *proxy* terhadap *single server* dan *multi server.* Untuk hasil pengujian *Memory Usage* pada penelitian ini dapat dilihat pada Tabel 11.

Tabel 11. CPU *Usage* dan *Memory Result*

Status Proxy	***Memory (MB)***	
	Single Server	***Multi Server***
Proxy Aktif	160 MB	210 MB
Proxy Non-Aktif	100 MB	130 MB
Persentase	60%	61,50%

Berdasarkan hasil yang tertera pada Tabel 11, hasil dari pengujian *memory usage* pada sistem yang menonaktifkan *proxy* menghasilkan performa lebih baik dalam penggunaan *memory* dibandingkan sistem yang mengaktifkan *proxy*. Pada sistem yang mengaktifkan *proxy* terhadap *single server* menghasilkan penggunaan *memory* sebesar 160 MB lebih

besar 60% daripada sistem yang menonatifkan *proxy* dengan hasil yang didapatkan sebesar 100 MB. Pada pengujian kedua dengan *multi server*, sistem yang mengaktifkan *proxy* menggunakan *memory* sebsar 210 MB dengan presentasi 61,50% lebih besar dari pada penggunaan *memory* pada sistem yang tidak mengaktifkan *proxy*. Peningkatan penggunaan *memory* pada sistem yang mengaktifkan *proxy* disebabkan oleh *overhead* tambahan dari manajemen sesi, inspeksi *header*, dan fitur-fitur lainnya yang berjalan pada *layer* 7.

Meskipun hasil pengujian menunjukkan bahwa algoritma *weighted least connection* memiliki kinerja yang lebih unggul dalam berbagai skenario pengujian, tetapi penting untuk memahami bahwa pemilihan algoritma *load balancing* juga bergantung pada konteks operasional dan tujuan sistem. *Weighted Least Connection* dalam hal adaptivitas terhadap beban koneksi aktif. Algoritma ini memonitor jumlah koneksi secara *real-time*, *weighted least connection* dapat mendistribusikan beban secara lebih seimbang ke *server* yang memiliki kapasitas tersedia lebih besar. Namun, keunggulan ini disertai dengan kompleksitas tambahan dalam proses monitoring koneksi aktif, yang dapat meningkatkan beban kerja pada sistem *load balancer*, terutama pada skala infrastruktur yang besar.

Sementara itu, algoritma *Weighted Round Robin* menawarkan kemudahan dalam konfigurasi dan efisiensi sistem, karena pembagian beban dilakukan berdasarkan bobot statis yang telah ditentukan sebelumnya tanpa memperhitungkan kondisi koneksi aktif saat itu. Algoritma ini sangat cocok digunakan pada sistem dengan spesifikasi *server* yang homogen serta beban *traffic* yang relatif stabil.

Namun, algoritma ini tidak cocok digunakan jika beban *traffic*-nya dinamis, karena dapat menyebabkan ketidakseimbangan distribusi beban yang dapat berdampak pada meningkatnya waktu respons dan tingkat eror. Dengan demikian, *trade-off* utama antara kedua algoritma terletak pada aspek kesederhanaan versus adaptivitas. Pemilihan algoritma *load balancing* yang tepat sebaiknya disesuaikan dengan karakteristik sistem dan kebutuhan operasional.

Meskipun algoritma *Weighted Least Connection* menunjukkan kinerja unggul dalam lingkungan dinamis, beberapa studi seperti [17] menunjukkan bahwa algoritma *Weighted Round Robin* tetap relevan dalam lingkungan server homogen dengan beban stabil, karena proses distribusinya lebih ringan dan deterministik. Selain itu, studi terbaru [18] dan [19] menyajikan perkembangan algoritma dan aplikasi *load balancing* modern, termasuk evaluasi performa HAProxy dibanding platform lain seperti Nginx, Traefik, dan Envoy, serta penerapan algoritma adaptif berbasis optimasi dan *machine learning*.

V. SIMPULAN

Melalui penelitian ini, ditarik kesimpulan yaitu *Load Balancing webserver* dengan HAProxy yang menggunakan dua buah algoritma berhasil dilakukan. Kemudian, pada pengujian *throughput* algoritma *weighted least connection* hampir secara keseluruhan memiliki hasil yang lebih bagus dengan hasil terbaik sebesar 1018,6 Kbits/s dan pada algoritma *Weighted Round Robin throughput* terbaik sebesar 1059,8. Pada parameter *response time*, algoritma *weighted least connection* memberikan hasil yang cukup maksimal. Walaupun pada pengujian yang menggunakan dua *web server*, kedua algoritma memberikan hasil yang cukup baik dan *weighted least connection* memiliki hasil 1ms lebih baik pada dua buah pengujian yang menggunakan dua *web server*. Dalam hal mengenai *error request*, kedua algoritma menujukkan hasil yang sangat baik. Akan tetapi, pada pengujian dengan tiga buah *web server* algoritma *weighted least connection* menunjukkan hasil yang lebih baik dibanding dengan algoritma *Weighted Round Robin* dengan selisih yang didapatkan sebesar 25,98%. Pada parameter pengujian penggunaan *memory* atau *memory usage*, sistem yang menonaktifkan *proxy* menghasilkan hasil yang lebih baik dibandingkan pada sistem yang mengaktifkan *proxy* dengan selisih persentase antara keduanya sebesar 60% untuk *single server* dan 61,50% untuk *multiserver*. Algoritma *weighted least connection* mempunyai nilai yang lebih unggul di segala aspek pengujian dibanding algoritma *Weighted Round Robin*.

Meskipun penelitian ini memberikan gambaran yang jelas mengenai perbandingan performa antara algoritma WRR dan WLC, terdapat beberapa keterbatasan yang perlu diperhatikan. Pengujian dilakukan dalam lingkungan virtual menggunakan VirtualBox, yang meskipun dapat merepresentasikan skenario nyata secara umum, tetap memiliki keterbatasan dalam mereplikasi performa dan kompleksitas sistem produksi skala besar. Selain itu, pengujian difokuskan pada trafik HTTP dengan parameter tertentu, tanpa melibatkan jenis trafik lain seperti HTTPS, WebSocket, atau API *burst* yang juga umum digunakan dalam sistem modern. Penelitian ini juga belum mengevaluasi dampak jangka panjang terhadap *resource* seperti konsumsi CPU dan I/O *disk* secara mendetail.

Untuk pengembangan penelitian selanjutnya, disarankan untuk menguji algoritma ini dalam lingkungan *cloud* berbasis kontainer seperti Docker dan Kubernetes, serta menambahkan parameter evaluasi seperti *latency* antar lokasi geografis (*geo-distributed systems*) dan efisiensi algoritma dalam mengelola sumber daya di bawah beban ekstrem. Selain itu, penggabungan algoritma WLC dengan pendekatan berbasis *machine learning* untuk prediksi beban masa depan dapat menjadi arah yang menarik guna menciptakan *sistem load balancing* yang adaptif dan cerdas.

REFERENSI

[1] R. Mustajab, “Pengguna E-Commerce RI Diproyeksi Capai 196,47 Juta pada 2023,” Data Indonesia: Data Indonesia for Better Decision. Valid, Accurate, Relevant, Sep. 04, 2023. Accessed: Dec. 21, 2023. [Online]. Available: https://dataindonesia.id/ekonomi-digital/detail/pengguna-ecommerce-ri-diproyeksi-capai-19647-juta-pada-2023.

[2] E. Raisah &. A. Nugroho, "*Ketahui Apa Itu Load Balancing, Cara Kerja, Kelebihannya, DCloud*. Accessed: Dec. 21, 2023. [Online] Available at: https://dcloud.co.id/blog/apa-itu-load-balancing.html

[3] M. Rosyida, "*Load Balancing : Pengatur Lalu Lintas Data Server,"* Dec. 6, 2023. [Online]. Available: https://www.domainesia.com/berita/load-balancing/

[4] Saputra, I. B., "Perancangan dan Implementasi Load Balancing Web Server menggunakan Haproxy,". *Skripsi Sarjana, Universitas Muhammadiyah Surakarta.* 2012.

[5] Setyawan, A. D., "*Mengenal Apa itu HAProxy serta Konsep Load Balancing, Rumahweb Journal – News, Article, and Tutorial of Web Dev.* Accessed: Dec. 22 2023. [Online]. Available at: https://www.rumahweb.com/journal/haproxy-adalah/

[6] Alsyaibani & Omar, M. A., "Performa Algoritma Load Balance pada Server Web Apache dan Nginx dengan Database Postgresgl," *Skripsi Sarjana, Universitas Negeri Yogyakarta,* 2013.

[7] Sumbayak, G. S. N. H. &. P. R., "Implementasi Algoritma Weighted Least Connection Berbasis Agen Pada POX Controller Untuk Load Balancing Web Server Pada Software Defined Network.". *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer.* 2019. Hal 7335-7344.

[8] Supramana, &. P. I. G. L. P, "Implementasi Load Balancing pada Web Server dengan Menggunakan Apache". *Jurnal Manajemen Informatika.* 2016.

[9] Anicas, M., "*Introduction to HAProxy and Load Balancing Concept,"* Accessed: Feb. 15, 2024. [Online]. https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts.

[10] Bustomi, Z. S. M. A. M. I. &. H. K. F. H., n.d, "Load Balancing Web Server Menggunakan Nginx pada Lingkungan Virtual". *Jurnal Informatika: Jurnal pengembangan IT,* 2019.

[11] Rachmawan, D. I. D. &. A. H., "Penerapan Teknik Load Balancing pada Web Server Lokal dengan Metode NTH Menggunakan Mikrotik," *Jurnal Penelitian Ilmu Komputer, System Embedded & Logic,* pp. 98-108, 2016.

[12] Rahmatulloh, A. &. M. F., "Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi," *Jurnal Nasional Teknologi dan Sistem Informasi.* 2017

[13] Apriliansyah, F. F. I. &. I. A., "Implementasi Load Balancing Pada Web Server Menggunakan Nginx," *Jurnal Teknologi dan Manajemen Informatika,* 2020.

[14] Riska, &. A. H., "Analisa Dan Perancangan Load Balancing Web Server menggunakan HAProxy. Techno.COM". 2021 hal. 552-565.

[15] Riskiono, S. D. &. P. D., "Analisis Perbandingan Server Load Balancing dengan Haproxy & Nginx dalam Mendukung Kinerja Server E- Learning,". *Jurnal Telekomunikasi dan Komputer. 2020.*

[16] Singh, G. &. K. K., "An Improved Weighted Least Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems,". *International Research Journal of Engineering and Technology.* 2018.

[17] Oliver, H., "*Load Balancing Matchup: Round Robin vs. Weighted Round Robin, Best DNS Servers - Uptime & Performance''* Accessed: Apr. 22, 2024. [Online] https://constellix.com/news/load-balancing-round-robin-vs-weighted-round-robin.

[18] Johansson, A, "HTTP Load Balancing Performance Evaluation of HAProxy, NGINX, Traefik and Envoy with the Round-Robin Algorithm," *Thesis of Undergraduate, University of Skovde).2022*

[19] Mbarek, F. &. M. V, "Load Balancing Based on Optimization Algorithms: An Overview," *Journal of Telecommunications and Information Technology.* 2019.

Analisis Efektivitas *Tools* SQLMap, Havij, dan Ghauri dalam Melakukan Serangan *SQL Injection* pada *Website*

Difa Maulana[1,*], Alif Subardono[1]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
difamaulana@mail.ugm.ac.id
[*]Korespondensi: alif@ugm.ac.id;

Abstract - A website is a collection of pages displaying information that is both static and dynamic. With the development of websites, they have evolved beyond being just informational platforms, also serving as e-commerce, media sharing, e-learning platforms, etc. One of the cyber threats that can jeopardize website security is SQL injection. SQL injection is a security attack technique on a website that manipulates user inputs using malicious SQL queries. SQL injection attacks can result in damage to the database and leakage of sensitive data. One way to anticipate SQL injection attacks is by conducting periodic vulnerability testing on the website. Vulnerability testing for SQL injection can be performed in various ways, one of which is testing using specialized tools for SQL injection vulnerabilities. SQLMap, Havij, and Ghauri are examples of specialized tools that can be used for vulnerability testing against SQL injection. Therefore, this research aims to identify the effective tool among SQLMap, Havij, and Ghauri in conducting SQL injection attacks on websites. The tool with the most complex attack complexity, detailed testing results, and the most comprehensive feature support will be considered the most effective. Based on the research findings, SQLMap is the most effective tool because it has the most complex attack complexity, provides detailed testing results, and has the most comprehensive feature support.

Keywords – Website. SQL Injection, SQLMap, Havij, Ghauri

Intisari - *Website* merupakan kumpulan halaman yang menampilkan informasi yang bersifat statis maupun dinamis. Seiring dengan perkembangan *website*, *website* tidak sekedar menjadi media informasi saja melainkan juga sebagai *e-commerce*, *media sharing*, platform *e-learning*, dll. Salah satu serangan siber yang dapat mengancam keamanan *website* adalah *SQL injection*. *SQL injection* merupakan teknik serangan keamanan pada sebuah *website* dengan memanipulasi masukkan pengguna menggunakan queri SQL berbahaya. Serangan *SQL injection* dapat mengakibatkan kerusakan pada *database* dan kebocoran data sensitif. Salah satu cara untuk mengantisipasi serangan *SQL injection* adalah dengan melakukan pengujian kerentanan *SQL injection* pada *website* secara berkala. Pengujian kerentanan *SQL injection* dapat dilakukan dengan berbagai cara salah satunya adalah pengujian meggunakan *tools* khusus kerentanan *SQL injection*. SQLMap, Havij, dan Ghauri merupakan beberapa contoh *tools* khusus yang dapat digunakan untuk melakukan pengujian kerentanan *SQL injection*. Oleh karena itu, penelitian ini bertujuan untuk menemukan *tools* yang efektif diantara SQLMap, Havij, dan Ghauri dalam melakukan serangan *SQL injection* pada *website*. *Tools* yang memiliki kompleksitas serangan paling kompleks, dapat memberikan hasil pengujian secara detail dan memiliki dukungan fitur paling lengkap akan dianggap sebagai *tools* yang paling efektif. Berdasarkan hasil penelitian, SQLMap merupakan *tools* yang paling efektif karena memiliki kompleksitas serangan paling kompleks, dapat memberikan hasil pengujian secara detail dan memiliki dukungan fitur paling lengkap.

Kata Kunci: *Website*, *SQL Injection*, SQLMap, Havij, Ghauri

I. PENDAHULUAN

Pada era modernisasi saat ini, teknologi informasi telah berkembang sangat pesat dan selalu mengalami perubahan sehingga teknologi informasi memberikan dampak atau pengaruh yang sangat besar dalam kemajuan penyebaran informasi [1]. Informasi digunakan sebagai penentu dalam pengambilan suatu keputusan, pemecahan suatu masalah dan sebagai ilmu pengetahuan. Informasi bisa didapatkan dari berbagai media seperti media cetak, televisi dan salah satu media yang cukup popular untuk saat ini adalah *website*.

Website dapat diartikan sebagai kumpulan halaman yang menampilkan informasi data teks, data gambar diam atau gerak, data animasi, suara, video atau gabungan dari semuanya baik yang bersifat statis maupun dinamis yang membentuk rangkaian yang saling terkait dimana masing-masing dihubungkan dengan jaringan *interne* [2]. Saat ini, *website* tidak lagi hanya sekedar menjadi media informasi saja melainkan penggunaan *website* telah menyebar ke berbagai bidang seperti bisnis, pendidikan, kesehatan dan lain sebagainya. Beberapa contoh penggunaan *website* selain sebagai media informasi ialah *e-commerce*, *video sharing*, publikasi digital dan platform *e-learning*.

Seiring dengan perkembangan penggunaan *website*, teknologi yang digunakan dalam pembuatannya juga berkembang. Salah satu perkembangan dalam pembuatan *website* adalah penggunaan *database*. *Website* sebagai media informasi dapat dibuat hanya dengan menggunakan HTML, CSS dan Javascript serta sebagian *website* tersebut tidak memerlukan penggunaan *database*. Sedangkan *website* yang lebih kompleks seperti *website* dinamis, *e-commerce* dan platform *e-learning* sangat memerlukan *database*. *Database* digunakan untuk penyimpanan dan pengelolaan data konten dan informasi-informasi yang diperlukan *website*. Selain itu, *database* juga digunakan untuk menyimpan informasi dari *end user* seperti *username, password* dan *email* yang diperlukan untuk autentikasi dan akses lainnya pada *website.*

Website yang rentan akan mudah sekali mengalami kebocoran data. Pada 17 April 2020, salah satu *marketplace* milik Indonesia yaitu Tokopedia mengalami kebocoran data.

Data Tokopedia yang berhasil tercuri sangat besar dan krusial karena data yang tercuri merupakan data kredensial milik pengguna berupa *email*, *password* dan nama pengguna [3]. Menurut oketechno, pada tahun 2023 terjadi empat kasus kebocoran data diantaranya ialah kebocoran data BPJS Ketenagakerjaan. Total data yang bocor dari kasus ini berjumlah 19 juta data pengguna meliputi nama, *email,* NIK, nomor telepon, tanggal lahir dan jenis kelamin. Kebocoran data Bank Syariah Indonesia (BSI) juga mengalami kebocoran data. Total data yang bocor dari kasus ini berjumlah 1,5 TB data dengan 15 juta data diantaranya merupakan data pengguna dan *password* untuk akses internal [4].

Salah satu serangan siber yang dapat menyebabkan kebocoran data adalah *SQL Injection*. *SQL injection* adalah teknik serangan pada sebuah *website* yang memanipulasi masukkan pengguna ke dalam sebuah query SQL. Serangan *SQL injection* dapat mengakibatkan kerusakan pada *database* dan kebocoran data sensitif [5]. Berdasarkan OWASP TOP 10 tahun 2023, *SQL injection* berada pada posisi ketiga yang berarti *SQL injection* termasuk serangan yang cukup berbahaya dan mengancam bagi keamanan *website*.

Salah satu cara yang dapat dilakukan untuk mengantisipasi dari serangan *SQL injection* adalah dengan melakukan pengujian keamanan *website* secara berkala. Namun, tingginya kasus serangan yang diakibatkan oleh *SQL injection* membuat individu maupun berbagai organisasi dibidang keamanan siber berupaya untuk membuat *tools-tools* khusus yang dapat mendeteksi kerentanan *website* dan melakukan pengujian kerentanan *SQL injection*. Dengan banyaknya *tools* yang beredar membuat pengembang *web* atau praktisi keamanan perlu melakukan pengkajian terhadap *tools-tools* tersebut baik dari segi efisiensi waktu, efektivitas *tools* dan dukungan fitur yang dimiliki.

Oleh karena itu, penelitian ini bertujuan untuk dapat memberikan informasi-informasi baru mengenai keefektifan sebuah *tools* dari segi pengambilan data, kompleksitas serangan dan dukungan fitur serta dapat digunakan untuk melakukan pengujian kerentanan *website* dari serangan *SQL injection* beserta cara kerja *tools* tersebut dalam melakukan serangan.

II. DASAR TEORI

A. Keamanan Informasi

Keamanan informasi merupakan sebuah upaya dalam mengamankan aset informasi dari kemungkinan ancaman yang akan terjadi. Keamanan informasi berperan penting dalam mengurangi risiko yang terjadi dan mengoptimalkan pengembalian nilai investasi. Semakin banyak informasi yang disimpan, dikelola dan dibagikan maka semakin tinggi juga risiko kerusakan yang terjadi dan tereksposnya data ke pihak eksternal [6].

Dalam perancangannya, keamanan informasi memiliki beberapa aspek-aspek antara lain *confidentiality, integrity* dan *availability* [7]. Aspek CIA Triad merupakan indikator keamanan informasi yang digunakan oleh para ahli sebagai alat ukur keamanan sebuah aplikasi web. *Confidentiality* merupakan suatu langkah untuk memastikan kerahasiaan suatu *website* agar orang yang tidak memiliki kewenangan tidak dapat mengaksesnya. *Integrity* merupakan keaslian data dimana data tidak boleh diubah oleh orang yang tidak memiliki kewenangan. *Availability* merupakan suatu langkah untuk memberikan jaminan autentikasi kepada pengguna bahwa data selalu tersedia ketika data diperlukan [8].

B. Serangan Siber

Serangan siber dapat didefinisikan sebagai serangan yang dilakukan oleh seseorang menggunakan sebuah komputer yang dapat mempengaruhi sistem serta mencuri data yang dirancang untuk menyebabkan kerusakan pada target dan memberikan keuntungan bagi penyerang [9]. Serangan siber dapat mencakup berbagai teknik seperti *malware, phising* dan penolakan layanan (DoS). Konsekuensi dari serangan ini sangat serius seperti kerugian finansial, pencurian identitas dan rusaknya reputasi bisnis dan individu. Faktor manusia menjadi kelemahan utama dalam keamanan siber. Banyak jenis kejahatan siber yang sering terjadi diantaranya rekayasa sosial, kejahatan terkait identitas, peretasan dan penolakan layanan dan informasi [10].

C. *SQL Injection*

SQL injection merupakan teknik serangan yang digunakan untuk mengetahui kerentanan pada *website* yang menyebabkan seorang penyerang dapat mengakses *database*. Serangan ini sangat berisiko karena dapat menyebabkan hilangnya data dan menyalahgunakan data oleh orang yang tidak bertanggungjawab [11]. Penyerang akan menyuntikkan perintah SQL berbahaya ke dalam *input* formulir untuk mendapatkan akses *database* atau memanipulasinya seperti memodifikasi data, mengambil data-data sensitif atau menghapus data [12]. kerentanan terhadap *SQL injection* dapat terjadi akibat kurangnya *filter* terhadap masukan dari pengguna dan tidak adanya verifikasi dari sisi *server* sehingga penyerang dapat dengan mudah mencuri data-data sensitif dan mendapatkan otoritas kontrol *server* [13].

D. Kriptografi

Kriptografi merupakan suatu cara untuk menjaga kerahasiaan suatu pesan baik berupa data maupun informasi yang mempunyai arti atau nilai dengan cara menyamarkan menjadi bentuk yang tidak dapat dimengerti dan hanya penerima yang dapat mengubah kode-kode tersebut menjadi pesan asli yang dapat dimengerti [14].

E. Kali Linux

Kali Linux merupakan platform pengujian penetrasi yang paling popular dan *powerful* di dunia. Kali linux biasa digunakan oleh para praktisi keamanan professional dalam berbagai spesialisasi seperti pengujian penetrasi, forensik, *reverse engineering* dan penilaian kerentanan [15].

F. SQLMap

SQLMap merupakan alat pengujian penetrasi bersifat *open-source* yang dapat melakukan deteksi dan eksploitasi kerentanan *SQL injection* secara otomasi dan mengambil alih *database server.* SQLMap memiliki banyak fitur yang digunakan oleh para penguji penetrasi diantaranya seperti *database fingerprinting*, pengambilan data pada *database* hingga mengakses *file system* [16]. SQLMap tidak memiliki antarmuka grafis melainkan menggunakan CLI sehingga semua perintah dapat dilakukan dengan mengetikkan perintah yang diinginkan pada *commands* konsol [17]. Dalam melakukan eksploitasi kerentanan *SQL injection*, SQLMap mendukung 6 teknik serangan *SQL injection* yaitu *boolean-based blind, time-based blind, error-based, union query-based, stacked queries* dan *inline queries.* Selain itu, SQLMap juga memiliki beberapa fitur yang mendukung eksploitasi target seperti dukungan pada beberapa sistem manajemen *database*, pengenalan otomatis format hash kata sandi dan dukungan untuk memecahkan format hash kata sandi menggunakan serangan *dictionary* [18].

G. Havij

Havij merupakan *tools* yang dijalankan untuk menemukan titik lemah dan mengeksploitasi halaman *web* dan berfokus pada pengaksesan *file* sistem [19]. Havij biasa digunakan untuk mencari kerentanan *SQL injection* pada sebuah halaman *web* atau melakukan pengujian penetrasi seperti melakukan *fingerprinting* pada *database*, mengambil hash dari *username* dan *password* pengguna, mengambil data yang ada di *database*, menjalankan perintah SQL dan bahkan mengakses *file* sistem [20]. Dalam melakukan serangan *SQL injection*, havij mendukung 4 teknik serangan *SQL injection* seperti *time-based blind, error-based, union-based* dan *boolean-based blind* [21].

H. Ghauri

Ghauri merupakan alat pengujian *SQL injection* yang bersifat *open-source* yang dirancang untuk identifikasi dan eksploitasi kerentanan *SQL injection* pada aplikasi *web* secara otomasi. Ghauri dilengkapi dengan beberapa fitur untuk melakukan pengujian kerentanan seperti pemindaian otomatis, identifikasi kerentanan, eksploitasi kerentanan dan mendukung *cross-platform*. Ghauri juga memiliki beberapa dukungan jenis manajemen *database* seperti MySQL, Microsoft SQL Server, Postgres, Oracle dan Microsoft Access. Dalam melakukan eksploitasi kerentanan *SQL injection*, Ghauri mendukung 4 teknik serangan *SQL injection* yaitu *boolean-based blind, error-based, time-based* dan *stacked queries* [22].

III. METODOLOGI

A. Alat Penelitian

Alat yang digunakan dalam penelitian berupa laptop dengan spesifikasi seperti pada Tabel 1.

Tabel 1. Spesifikasi Laptop

Spesifikasi	Detail
Seri	Asus
Operating System	Windows 11 Pro 64 bit
Processor	AMD Ryzen 7 3750H
RAM	16384 MB

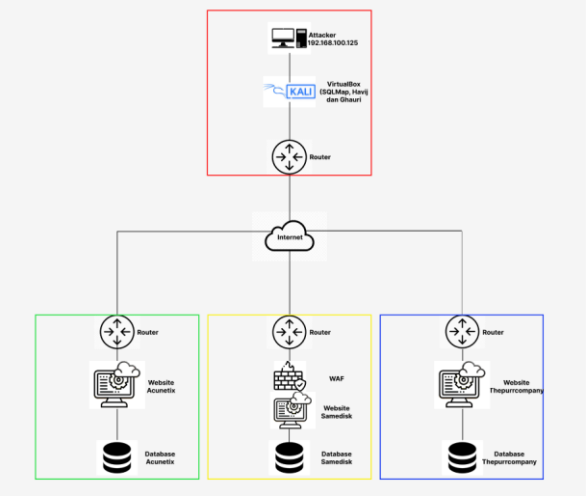
B. Bahan Penelitian

Bahan yang digunakan dalam penelitian ini adalah sebagai berikut.

- *Oracle VM VirtualBox*
  Spesifikasi *Oracle VM VirtualBox* menggunakan *Virtual Graphical User Interface* dengan versi 7.0.8.
- Kali Linux
  Kali Linux menggunakan *Operating System* Kali Linux 2023.2 dengan RAM 2048 MB, vCPU 2, dan *network* jenis *Bridge Adapter*.
- SQLMap
  SQLMap dijalankan menggunakan *Operating System* Kali Linux 2023.2 dengan RAM 2048 MB.
- Havij
  Havij dijalankan menggunakan *Operating System* Kali Linux 2023.2 dengan RAM 2048 MB
- Ghauri
  Ghauri dijalankan menggunakan *Operating System* Kali Linux 2023.2 dengan RAM 2048 MB

C. Topologi Sistem

Sistem yang dibangun memiliki topologi seperti pada Gambar 1.



Gambar 1. Topologi Sistem

Berdasarkan Gambar 1, *attacker* yang akan melakukan penyerangan *SQL injection* menggunakan platform Kali Linux. Di dalam Kali Linux terdapat *tools* khusus untuk melakukan serangan *SQL injection* yaitu SQLMap, Havij dan Ghauri. *Attacker* melakukan penyerangan terhadap tiga target. Setiap target akan diserang menggunakan 3 *tools SQL injection* yaitu SQLMap, Havij dan Ghauri. Target pertama

yaitu *website* Acunetix, target kedua yaitu *website* samedisk, dan target ketiga yaitu *website* thepurrcompany.

D. Pengujian

Pengujian akan dilakukan dengan dua tahap yaitu pencarian *database* target dan pengambilan data *database*. Pengujian akan dilakukan pada setiap *tools* terhadap setiap target. Target yang akan digunakan adalah sebagai berikut.

- *Website* Acunetix http://testphp.vulnweb.com/listproducts.php?cat=1
- *Website* Samedisk https://samedisk.com/en/productinfo.php?id=1
- *Website* Thepurrcompany http://www.thepurrcompany.com/cat-articles/linking-info.php?id=7

Tahapan pencarian *database* target digunakan untuk menentukan kemampuan *tools* dalam menembus pertahanan target sedangkan tahapan pengambilan data *database* digunakan untuk menentukan kemampuan *tools* dalam melakukan eksploitasi.

a. Pencarian *Database* Target

Langkah untuk melakukan pencarian *database target* menggunakan perintah seperti pada Gambar 2.

```
└─$ time sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --leve
l 5 --risk 3 --dbs
```

Gambar 2. Perintah Serangan Pencarian *Database Website Acunetix* Menggunakan SQLMap

Gambar 2 merupakan salah satu contoh perintah serangan *SQL injection* yang digunakan untuk pencarian *database* target. Pada SQLMap dan Ghauri menggunakan *flag* --level dan --risk pada tingkatan tertinggi *tools* tersebut guna memaksimalkan kemampuan serangan.

b. Pengambilan Data *Database* Target

Langkah untuk melakukan pengambilan data *database target* menggunakan perintah seperti pada Gambar 3.

```
└─$ time sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --leve
l 5 --risk 3 -D acuart -T users -C name,email,pass --dump
```

Gambar 3. Perintah Serangan Pengambilan Data *Database Website Acunetix* Menggunakan SQLMap

Gambar 3 merupakan salah satu contoh pengambilan data *database* target menggunakan SQLMap. Data yang diambil pada tahapan pengambilan data *database* adalah data tabel, data kolom dan data sensitif. Data sensitif yang akan diambil adalah data akses pengguna seperti *username* dan *password* atau data diri pengguna seperti nama, nomor telepon dan *email*. Namun, untuk mendapatkan data tersebut perlu dilakukan beberapa tahapan.

Tahapan pertama yaitu mendapatkan data tabel yang ada pada *database* target. Mendapatkan data tabel pada *database* target dapat menggunakan perintah *SQL injection* seperti pada Gambar 4.

```
└─$ time sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --level 5 --risk 3 -D acuart
--tables
```

Gambar 4. Perintah Serangan Pengambilan Data Tabel pada *Database Website Acunetix* Menggunakan SQLMap

Tahapan kedua yaitu mendapatkan data kolom yang ada pada tabel. Mendapatkan data kolom pada tabel di *database* target dapat menggunakan perintah *SQL injection* seperti pada Gambar 5.

```
└─$ time sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --level 5 --risk 3 -D acuart
-T users --columns
```

Gambar 5. Perintah Serangan Pengambilan Data Kolom pada Tabel *Users* pada *Database Website Acunetix* Menggunakan SQLMap

Tahapan ketiga yaitu mendapatkan data yang ada pada kolom di tabel dari *database* target. Tahapan ketiga ini merupakan tahapan terakhir dimana pada tahapan ini kita dapat mengambil data yang ada pada *database* target seperti data *username* atau *password* maupun data diri pengguna seperti nama, nomor telepon dan *email.*

IV. HASIL DAN PEMBAHASAN

A. Pencarian *Database* Target

Serangan *SQL injection* yang dijalankan setiap *tools* untuk mendapatkan daftar *database* pada *website* target dilakukan sebanyak 15 kali. Dari hasil serangan tersebut, didapatkan waktu rata-rata setiap *tools* dalam melakukan serangan pada setiap target seperti pada Tabel 2.

Tabel 2. Waktu Rata-Rata Serangan

Tools	**Target**	**Waktu (s)**
SQLMap	Acunetix	5.436,73
	Samedisk	12.579,89
	Thepurrcompany	7.303,06
Havij	Acunetix	13,42
	Samedisk	-
	Thepurrcompany	10,21
Ghauri	Acunetix	60,39
	Samedisk	484,37
	Thepurrcompany	282,84

Berdasarkan Tabel 6, waktu rata-rata serangan setiap *tools* pada setiap target berbeda. Hal tersebut disebabkan karena adanya perbedaan dari kemampuan setiap *tools* dalam melakukan kerentanan. Setiap *tools* memiliki dukungan teknik serangan *SQL injection* yang berbeda. SQLMap memiliki dukungan terhadap 6 teknik serangan *SQL injection* yaitu *boolean-based blind, time-based blind, error-based, union query, stacked queries* dan *inline queries*. Havij memiliki dukungan terhadap empat teknik serangan *SQL injection* yaitu *time-based blind, error-based, union-based* dan *boolean-based blind*. Sedangkan Ghauri memiliki dukungan terhadap empat teknik serangan *SQL injection* yaitu *boolean-based blind, error-based, stacked queries* dan *time-based blind.*

Selain dukungan teknik serangan, ketiga *tools* juga memiliki kemampuan yang berbeda dalam melakukan pendeteksian kerentanan. Pada SQLMap, terdapat fitur --*level* dan --*risk*. Kedua fitur ini berguna untuk meningkatkan kemampuan serangan. --*level* digunakan untuk menentukan seberapa luas pengujian yang akan dilakukan. --*risk* digunakan untuk menentukan seberapa kompleks *payloads* yang akan digunakan. Pada Ghauri hanya terdapat fitur –*level* sedangkan Havij tidak memiliki keduanya karena Havij berbasis GUI.

Hasil dari pengujian pencarian *database* pada setiap *tools* dalam melakukan serangan *SQL injection* pada setiap target adalah SQLMap berhasil mendapatkan daftar *database* yang ada pada *website* acunetix dan thepurrcompany, tetapi gagal mendapatkan daftar *database* pada *website* samedisk. Hal tersebut terjadi karena fitur yang digunakan pada pengujian tidak dapat menembus pertahanan target yang menggunakan protokol HTTPS sehingga diperlukan penggunaan fitur yang lebih *advance* untuk menembus pertahanan tersebut. Havij berhasil mendapatkan daftar *database* yang ada pada *website* acunetix dan thepurrcompany, tetapi gagal mendapatkan daftar *database* pada *website* samedisk. Hal tersebut terjadi karena *website* samedisk menggunakan protokol HTTPS sedangkan Havij tidak mendukung target yang menggunakan protokol HTTPS sehingga Havij tidak dapat melakukan serangan terhadap *website* samedisk. Ghauri berhasil mendapatkan daftar *database* pada ketiga target.

Berdasarkan hasil waktu rata-rata serangan yang didapatkan dari ketiga *tools* yang dapat dilihat pada Tabel 6, didapati bahwa SQLMap memiliki rata-rata waktu serangan yang lebih lama dibandingkan dengan Havij dan Ghauri. Hal tersebut disebabkan karena adanya perbedaan dukungan teknik serangan yang dimiliki oleh masing-masing *tools*. SQLMap memiliki dukungan teknik serangan yang lebih banyak dibandingkan dengan Havij dan Ghauri yaitu sebanyak enam teknik serangan *SQL injection* sedangkan Havij dan Ghauri hanya memiliki empat teknik serangan *SQL injection.* Hal ini sangat berpengaruh terhadap waktu serangan yang dilakukan oleh setiap *tools*. Fitur pendeteksian kerentanan yang dimiliki masing-masing *tools* juga mempengaruhi waktu serangan dimana pada SQLMap dan Ghauri memiliki fitur pendeteksian kerentanan dan kompleksitas *payload* yang dapat memperluas pengujian serangan sedangkan Havij tidak memiliki fitur tersebut sehingga didapati hasil bahwa SQLMap membutuhkan waktu yang lebih lama untuk mendapatkan data *database* target diikuti Ghauri kemudian Havij.

B. Pengambilan Data *Database*

Pengambilan data pada *database* dilakukan sebanyak 15 kali untuk setiap *tools* pada setiap target. Jenis data yang diambil adalah daftar data tabel, daftar data kolom dan data sensitif. Data sensitif yang akan diambil adalah data akses pengguna seperti *username* dan *password* atau data diri pengguna seperti nama, nomor telepon dan *email*.

Tabel 3. Waktu Rata-Rata Pengambilan Data *Database*

Tools	**Target**	**Jenis Data**	**Waktu (s)**
SQLMap	Acunetix	Tabel	1,06
		Kolom	1,08
		Data Sensitif	1,04
	Samedisk	Tabel	-
		Kolom	-
		Data Sensitif	-
	Thepurrcompany	Tabel	11,29
		Kolom	3,61
		Data Sensitif	731,55
Havij	Acunetix	Tabel	1,35
		Kolom	1,23
		Data Sensitif	3,28
	Samedisk	Tabel	-
		Kolom	-
		Data Sensitif	-
	Thepurrcompany	Tabel	2,74
		Kolom	2,26
		Data Sensitif	3,93
Ghauri	Acunetix	Tabel	12,98
		Kolom	1,90
		Data Sensitif	1,83
	Samedisk	Tabel	5.757,63
		Kolom	3.757,16
		Data Sensitif	8.033,85
	Thepurrcompany	Tabel	16,42
		Kolom	3,44
		Data Sensitif	34,37

Berdasarkan Tabel 7, waktu rata-rata pengambilan data setiap *tools* pada setiap jenis data berbeda-beda terutama dalam pengambilan data sensitif. Hal tersebut disebabkan karena adanya perbedaan jumlah data yang disimpan pada *database*. Semakin banyak jumlah data yang disimpan maka waktu yang dibutuhkan untuk mengambil data tersebut semakin lama.

Perbedaan yang terjadi diantara ketiga *tools* juga terdapat pada kemampuan dalam pengambilan data sensitif terutama pada data *password*. Pada *website* acunetix, data *password* disimpan pada *database* dengan cara dienkripsi sehingga setiap *tools* harus melakukan dekripsi untuk mendapatkan data yang sebenarnya. Pada *website* samedisk, data *password* disimpan pada *database* dengan cara merubah data *password* ke nilai hash sehingga diperlukan kemampuan dalam melakukan *dictionary attack* untuk mendapatkan data *password*.

Hasil dari pengujian pengambilan data *database* pada setiap *tools* di setiap target adalah SQLMap berhasil mendapatkan data sensitif yang ada pada *website* acunetix dan thepurrcompany, tetapi gagal mendapatkan data sensitif pada *website* samedisk. Hal tersebut terjadi karena SQLMap gagal mendapatkan daftar *database* yang ada pada *website* samedisk sehingga SQLMap tidak dapat melakukan

eksploitasi lebih lanjut. Havij berhasil mendapatkan data sensitif yang ada pada *website* acunetix dan thepurrcompany, tetapi gagal mendapatkan data sensitif pada *website* samedisk. Hal tersebut terjadi karena Havij tidak memiliki dukungan serangan terhadap target yang menggunakan protokol HTTPS. Sedangkan, Ghauri berhasil mendapatkan data sensitif yang ada pada ketiga target, tetapi tidak berhasil mendapatkan data *password* pada ketiga target. Ghauri berhasil mendapatkan data *password* yang ada pada *website* samedisk, tetapi hanya mendapatkan nilai hashnya saja. Hal tersebut terjadi karena Ghauri tidak memiliki kemampuan dalam melakukan dekripsi *password* dan tidak memiliki dukungan terhadap *dictionary attack*.

C. Perbandingan Kompleksitas Serangan

Berdasarkan kompleksitas serangan, setiap *tools* memiliki kompleksitas serangan yang berbeda-beda. Hal tersebut terjadi karena setiap *tools* memiliki dukungan teknik serangan *SQL injection* yang berbeda-beda. SQLMap mendukung enam teknik serangan *SQL injection* yaitu *boolean-based blind, time-based blind, error-based, union query-based, stacked queries* dan *inline queries*. Havij mendukung empat teknik serangan *SQL injection* yaitu *time-based blind, error-based, union-based* dan *boolean-based blind* sedangkan Ghauri mendukung empat teknik serangan *SQL injection* yaitu *boolean-based blind, error-based, stacked queries* dan *time-based*.

Berdasarkan pengujian yang telah dilakukan, ketiga *tools* tidak menspesifikkan teknik serangan *SQL injection* yang digunakan. Maka secara *default*, ketiga *tools* melakukan serangan *SQL injection* dengan menggunakan seluruh teknik serangan yang didukung. Kompleksitas serangan akan berpengaruh terhadap waktu serangan, jumlah kerentanan yang ditemukan dan keberhasilan serangan. Semakin banyak teknik serangan yang digunakan maka semakin kompleks serangan yang akan dilakukan. Dengan semakin kompleks serangan yang dilakukan maka waktu serangan yang dibutuhkan akan semakin lama. Selain itu, semakin kompleks serangan yang dilakukan, jumlah kerentanan yang ditemukan akan semakin banyak. Kompleksitas serangan juga berpengaruh terhadap persentase keberhasilan serangan. Semakin kompleks serangan yang dilakukan maka persentase keberhasil serangan akan semakin tinggi.

D. Perbandingan Dukungan Fitur

setiap *tools* memiliki dukungan fitur yang dapat meningkatkan persentase keberhasilan pengujian serangan *SQL injection* dan melakukan serangan *SQL injection* sesuai dengan yang diinginkan. Untuk mengaktifkan fitur yang dimiliki, setiap *tools* memiliki caranya tersendiri. Pada SQLMap dan Ghauri menggunakan fitur yang tersedia dapat menggunakan *flag.* Hal tersebut dapat terjadi karena kedua *tools* tersebut berbasis CLI sedangkan pada Havij menggunakan fitur yang tersedia cukup memilih opsi-opsi yang ada karena Havij merupakan *tools* berbasis GUI.

SQLMap memiliki dukungan fitur yang cukup banyak dengan total fitur sebanyak 201 fitur. Dukungan fitur yang dimiliki SQLMap memiliki beragam fungsi seperti fitur untuk menentukan target, menspesifikkan cara untuk terhubung ke target, optimasi, injeksi, deteksi, menspesifikkan teknik serangan, *fingerprint*, dan sebagainya. SQLMap mendukung berbagai jenis *database* seperti MySQL, Oracle, PostgreSQL, dan lain-lain. SQLMap juga memiliki dukungan terhadap 6 teknik serangan *SQL injection* yaitu *boolean-based blind, time-based blind, error-based, union query-based, stacked queries* dan *inline queries*.

Havij merupakan *tools* yang berbasis GUI. Havij memiliki total fitur sebanyak 30 fitur. Dukungan fitur yang dimiliki Havij memiliki beragam fungsi seperti fitur untuk menspesifikkan *jenis database*, menentukan *method*, menentukan tipe *payload*, dan sebagainya. Havij mendukung berbagai jenis *database* seperti MySQL, MsAcess, Oracle, dan lain-lain. Havij juga memiliki dukungan terhadap empat teknik serangan *SQL injection* yaitu *time-based blind, error-based, union-based* dan *boolean-based blind*.

Ghauri memiliki total fitur sebanyak 50 fitur. Dukungan fitur yang dimiliki Ghauri beragam fungsi seperti fitur untuk menentukan target, menspesifikkan cara untuk terhubung ke target, optimasi, injeksi, deteksi, menspesifikkan teknik serangan dan enumerasi. Ghauri mendukung berbagai jenis *database* seperti MySQL, Microsoft SQL Server, Postgres, Oracle dan Microsoft Access. Ghauri juga memiliki dukungan terhadap empat teknik serangan *SQL injection* yaitu *boolean-based blind, error-based, stacked queries* dan *time-based*.

V. SIMPULAN

Melalui penelitian ini, dapat ditarik beberapa kesimpulan. Pertama, SQLMap berhasil melakukan serangan *SQL injection* dan mengeksploitasi dua target yaitu *website* acunetix dan *website* thepurrcompany. Kedua, Havij berhasil melakukan serangan *SQL injection* dan mengeksploitasi dua target yaitu *website* acunetix dan *website* thepurrcompany. Ketiga, Ghauri berhasil melakukan serangan *SQL injection* dan mengeksploitasi seluruh target. Kemudian, berdasarkan kompleksitas serangan, SQLMap memiliki serangan yang paling kompleks karena menggunakan enam teknik serangan *SQL injection* yaitu *boolean-based blind, error-based, stacked queries, time-based blind, union queries* dan *inline queries*. Selajutnya, SQLMap tidak berhasil melakukan serangan pada target yang menggunakan protokol HTTPS dan dibutuhkan fitur yang dikhususkan untuk menyerang target yang menggunakan protokol HTTPS. Terakhir, Havij tidak memiliki dukungan serangan pada target yang menggunakan protokol HTTPS.

REFERENSI

[1] W. D. Kurniawan, A. P. Budijono and Yunus, "Pengembangan Web Sebagai Media Informasi dan Promosi Program Studi S1 Pendidikan Teknik Mesin Jurusan Teknik Mesin UNESA," *Journal of Vocational and Technical Education,* vol. 02, pp. 41-49, 2020.

[2] W. Agustin, U. Rio, R. Muzawi, T. Nasution and D. Haryono, "Penguatan Pengelolaan Website Desa untuk Meningkatkan Layanan Administrasi Kependudukan di Desa Pasir Baru Rokan Hulu," *Jurnal Pengabdian Masyarakat Informatika,* vol. 1, pp. 8-17, 2021.

[3] R. Muhammad, "Perlindungan Data Diri Konsumen dan Tanggung Jawab Marketplace Terhadap Data Diri Konsumen (Studi Kasus: Kebocoran Data 91 Juta Akun Tokopedia)," *Jurnal Inovasi Penelitian,* vol. 3, 2023.

[4] Y. Tangguh, "4 Kasus Kebocoran Data Paling Heboh di Tahun 2023," Oketechno, 18 Desember 2023. [Online]. Available: https://techno.okezone.com/read/2023/12/18/54/2940876/4-kasus-kebocoran-data-paling-heboh-di-tahun-2023?page=2.

[5] M. Aprianto, "Desain dan Implementasi intrusion Detection System Menggunakan Debian 7 dan SNORT," *Teknologipintar.org,* vol. 3, 2023.

[6] A. Komelia and D. Irawan, "Analisis Keamanan Informasi Menggunakan Tools Indeks Kami ISO 4.1," *Jurnal Pengembangan Sistem Informasi dan Informatika,* vol. 2, 2023.

[7] B. Triandi, "Keamanan Informasi secara Aksiologi dalam Menghadapi Era Revolusi Industri 4.0," *Jurnal Riset Komputer,* Vols. 477-483, 2019.

[8] A. Hermawan, T. Hartati and Y. A. Wijaya, "Analisa Keamanan Data melalui Website Zahra Software Menggunakan Metode Keamanan Informasi CIA Triad," *Jurnal Pengembangan IT,* vol. 7, 2022.

[9] I. Dermawan, A. Baidawi, I. and S. M. Dewi, "Serangan Cyber dan Kesiapan Keamanan Cyber Terhadap Bank Indonesia," *Jurnal Informasi dan Teknologi,* vol. 5, 2023.

[10] D. A. Herera and M. H. Sebyar, "Perlindungan Hukum Terhadap Serangan Siber Tinjauan Atas Kebijakan dan Regulasi Terbaru," *Jurnal Hukum dan Kewarganegaraan,* vol. 1, 2023.

[11] M. A. Z. Risky and Yuhandri, "Optimalisasi dalam Penetrasi Testing Keamanan Website Menggunakan Teknik SQL Injection dan XSS," *Jurnal Sistim Informasi dan Teknologi,* vol. 3, pp. 215-220, 2021.

[12] A. Alanda, D. Satria, M. I. Ardhana, A. A. Dahlan and H. A. Mooduto, "Web Application Penetration Testing Using SQL Injection Attack," *International Journal on Informatics Visualization,* vol. 5, pp. 320-326, 2021.

[13] D. Chen, Q. Yan, C. Wu and J. Zhao, "SQL Injection Attack Detection and Prevention Techniques Using Deep Learning," *Journal of Physics: Conference Series,* 2020.

[14] Y. T. Andita, H. Aspriyono and E. P. Rohmawan, "Application of the Vigenere Cipher Algorithm in Database Security of Employee Performance Values at the South Bengkulu Regency Education and Culture Office," *Jurnal Komputer Indonesia,* pp. 89-92, 2022.

[15] R. R. Asaad, "Penetration Testing Wireless Network Attacks Methods on Kali Linux OS," *Academic Journal of Nawroz University,* vol. 10, 2021.

[16] S. D. U. K. Singh, A. Raghuvanshi and V. Rathore, "Towards Identification of Vulnerabilities and Their Exploits Using Penetration Testing Tools," *International Research Journal of Modernization in Engineering Technologi and Science,* vol. 05, 2023.

[17] N. Bedekovic, L. Havas, T. Horvat and D. Crcic, "The Importance of Developing Preventive Techniques for SQL Injection Attacks," *Tehnicki Glasnik,* vol. 16, pp. 523-529, 2022.

[18] A. M. Sari, T. Santhi, D. K. A. M. Putra and M. B. Haekal, "Pengukuran Efektivitas SQL Injection pada Website dengan Menggunakan Tools JSQL, Havij dan The Mole," *Jurnal Informatika dan Teknologi Komputer,* vol. 04, pp. 65-71, 2023.

[19] L. Febrianto, "Security Analysis of Authentication and Database on the Dock Management System Website against SQL Injection Attacks Using Penetration Testing Methods," *International Research Journal of Advanced Engineering and Science,* vol. 6, no. 1, pp. 199-204, 2021.

[20] N. Khan, "Ghauri," Github, 04 Oktober 2022. [Online]. Available: https://github.com/r0oth3x49/ghauri.

[21] W. Irawan, "SQLMap: Alat SQL Injection Otomatis dan Database Takeover," Waldika Irawan, 03 April 2021. [Online]. Available: https://www.waldikairawan.com/2021/04/sqlmap-tools-sql-injection.html.

[22] H. Adha, "Software yang dipakai untuk Hacking," Masherul.com Media Informasi, 30 Juli 2022. [Online]. Available: https://www.masherul.com/2022/07/software-yang-dipakai-untuk-hacking.html.