Implementasi *Abstraction Layer* untuk Propagasi Daftar Blokir Dinamis pada Infrastruktur *Firewall Multi-Vendor*

Muhammad Abian Abdi Pratama

Program Studi Teknik Multimedia dan Jaringan, Jurusan Teknik Informatika dan Komputer
Politeknik Negeri Jakarta, Depok, Jawa Barat
muhammad.abian.abdi.pratama.tik22@mhsw.pnj.ac.id

Abstract*—This study addresses manual management constraints and heterogeneity issues in multi-vendor firewall infrastructures (pfSense and OPNsense) that cause propagation delays in security rules. The proposed solution is the design of an Abstraction Layer based on Factory and Adapter design patterns as a vendor-agnostic communication bridge. The system implements a parallel execution mechanism using a thread pool to minimize the propagation latency of dynamic blocklists, as well as securing access credentials using AES-128-CBC encryption. Empirical testing was conducted by comparing parallel and sequential execution methods under various IP payload variations and bandwidth constraints. The results show that the parallel method successfully reduces propagation latency significantly with high stability. Under limited bandwidth conditions (50 KBps) with a payload of 40 IPs, the parallel method maintained a 100% success rate, while the sequential method dropped to 90% due to the accumulation of queueing delay that triggered timeouts. This architecture is proven effective in increasing the responsiveness and reliability of threat mitigation in distributed network environments.*

Keywords: *Abstraction Layer, Factory Pattern, Parallel Execution, pfSense, OPNsense, Propagation Latency*

Abstrak*—Penelitian ini mengatasi kendala manajemen manual dan masalah heterogenitas pada infrastruktur firewall multi-vendor (pfSense dan OPNsense) yang memicu keterlambatan propagasi aturan keamanan. Solusi yang diusulkan adalah perancangan Abstraction Layer berbasis pola desain Factory dan Adapter sebagai jembatan komunikasi vendor-agnostic. Sistem ini mengimplementasikan mekanisme eksekusi paralel menggunakan thread pool untuk meminimalkan latensi distribusi daftar blokir dinamis, serta mengamankan kredensial akses menggunakan enkripsi AES-128-CBC. Pengujian dilakukan secara empiris dengan membandingkan metode eksekusi paralel dan sekuensial pada berbagai variasi beban payload IP dan batasan bandwidth. Hasil pengujian menunjukkan metode paralel berhasil mereduksi latensi propagasi secara signifikan dengan kestabilan yang tinggi. Pada kondisi bandwidth terbatas (50 KBps) dengan beban 40 IP, metode paralel mampu mempertahankan success rate 100%, sedangkan metode sekuensial menurun hingga 90% akibat akumulasi queueing delay yang memicu timeout. Arsitektur ini terbukti efektif meningkatkan responsivitas dan keandalan mitigasi ancaman pada lingkungan jaringan terdistribusi.*

Kata kunci: *Eksekusi Paralel, Firewall Multi-Vendor, Lapisan Abstraksi, Latensi Propagasi, OPNsense, pfSense*

I. PENDAHULUAN

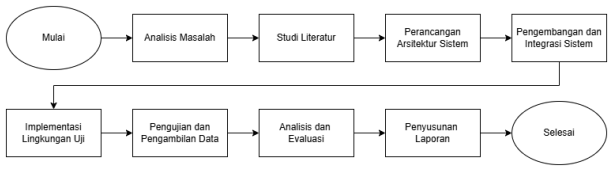
Keamanan infrastruktur jaringan komputer modern menghadapi ancaman siber yang terus berkembang, baik dalam volume maupun tingkat kecanggihannya. Berdasarkan Laporan BSSN 2024, serangan siber terdistribusi menuntut tim *Security Operations Center* (SOC) untuk dapat menerapkan kebijakan pemblokiran akses secara dinamis dan instan. Namun, operasional mitigasi ancaman pada organisasi berskala besar sering kali terbentur oleh kompleksitas heterogenitas perangkat keamanan jaringan. Pemanfaatan *firewall multi-vendor* umum dijumpai untuk mencegah vendor *lock-in* dan mengoptimalkan anggaran pengadaan perangkat keras [1].

Heterogenitas ini memicu masalah manajemen manual di mana analis keamanan harus masuk ke masing-masing *console firewall* secara bergantian (*context-switching*) untuk mendistribusikan daftar blokir IP bermasalah. Hal ini mengakibatkan adanya akumulasi penundaan waktu distribusi aturan keamanan (*propagation latency*). Selisih waktu antara pendeteksian ancaman hingga penerapan blokir fisik pada firewall (*breakout time*) menjadi celah krusial bagi aktor ancaman untuk bergerak secara lateral di dalam jaringan [2]. Apabila distribusi daftar IP bermasalah dilakukan secara manual lintas platform pfSense dan OPNsense, efektivitas mitigasi ancaman akan menurun drastis.

Untuk menyatukan dan mengotomatisasi proses tersebut, penelitian ini merancang dan mengimplementasikan sebuah lapisan abstraksi (*Abstraction Layer*) terpadu. Berbasis pola desain *Factory* dan *Adapter*, sistem ini menstandarisasi komunikasi API ke berbagai vendor *firewall*. Lebih lanjut, sistem memanfaatkan mekanisme *Thread Pool Executor* untuk memicu propagasi daftar blokir secara paralel ke seluruh *node firewall* target. Arsitektur ini diharapkan dapat mereduksi latensi propagasi secara signifikan, menjaga *success rate* distribusi di bawah batasan *bandwidth* jaringan, serta mengamankan kredensial administrasi dengan enkripsi AES-128-CBC.

II. METODOLOGI PENELITIAN

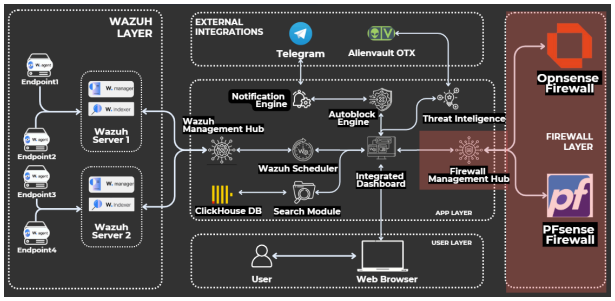
Metode penelitian ini didasarkan pada tahapan *System Engineering* dengan model *prototyping*. Langkah-langkah penelitian meliputi analisis kebutuhan perangkat keras dan lunak, desain arsitektur lapisan abstraksi, pembuatan modul program, implementasi lingkungan uji *multi-vendor*, pengambilan data empiris kualitatif-kuantitatif, serta evaluasi performa.



Gambar 1. Diagram Alur Tahapan Penelitian System Engineering

A. Desain Arsitektur Lapisan Abstraksi

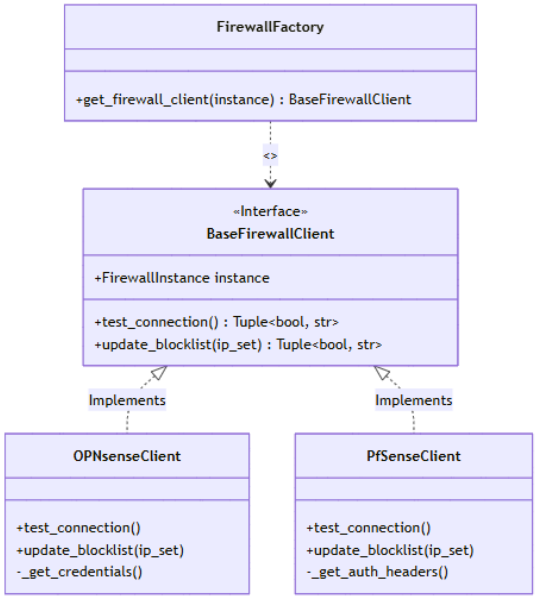
Lapisan abstraksi dirancang menggunakan pendekatan modular tiga lapis (*three-tier architecture*) yang memisahkan *User Layer, Application Layer* (berisi *logic* Hub, *database* PostgreSQL, caching Redis, dan *engine background scheduler*), serta *Firewall Layer* (terdiri atas *node* target pfSense dan OPNsense). Hub bertindak sebagai pengelola terpusat untuk mendeteksi ancaman dan meneruskannya ke lapisan abstraksi.



Gambar 2. Rancangan Arsitektur Hub Manajemen Firewall Multi-Vendor

B. Rancang Bangun Class Diagram Factory & Adapter

Standarisasi interaksi dengan API heterogen dicapai melalui implementasi abstract base class bernama `BaseFirewallClient`. Kelas abstrak ini mendefinisikan metode standard seperti `test_connection()` dan `update_blocklist()`. Kelas `FirewallFactory` bertugas membaca konfigurasi tipe vendor *instance* di basis data dan secara dinamis menginstansiasi adapter yang sesuai (yaitu *concrete class* `PfSenseClient` atau `OPNsenseClient`) melalui prinsip polimorfisme [3].



Gambar 3. Rancang Bangun Class Diagram Factory Pattern untuk Abstraksi API

C. Skenario Pengujian Kinerja

Performa sistem dievaluasi melalui serangkaian pengujian komparatif empiris antara metode eksekusi sekuensial (konvensional) dan paralel (*thread pool*) di bawah variasi ukuran *payload* (10, 20, 30, dan 40 IP) dengan masing-masing skenario diulang sebanyak 30 iterasi. Kestabilan distribusi divalidasi dengan membatasi kecepatan *bandwidth* jaringan (50 KBps, 100 KBps, 150 KBps, dan 200 KBps) menggunakan pembatas TC (*traffic control*).

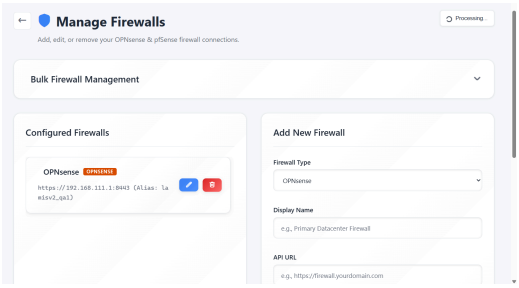
TABEL I. DAFTAR KEBUTUHAN SPESIFIKASI PERANGKAT KERAS

No	***Hardware***	**Spesifikasi**
1	Prosesor (CPU)	16 vCPU Intel® Xeon® E5-2630 v4 @ 2.20GHz (64-bit)
2	Memori (RAM)	16 GB DDR4 RAM
3	Penyimpanan	Solid State Drive (SSD) 500 GB
4	Jaringan	Virtualized Ethernet Interface 1 Gbps (Virtualized vif)

III. HASIL DAN PEMBAHASAN

A. Implementasi Web UI dan Manajemen Kredensial

Implementasi sistem menghasilkan antarmuka berbasis web Flask untuk memudahkan pendaftaran *instance firewall* baru. Untuk menjamin aspek kerahasiaan data (*data at rest*), kunci API dan kredensial sensitif dienkripsi menggunakan AES-128-CBC sebelum disimpan ke PostgreSQL. Modul `crypto.py` secara otomatis melakukan enkripsi dan dekripsi transparan saat kueri data dijalankan.



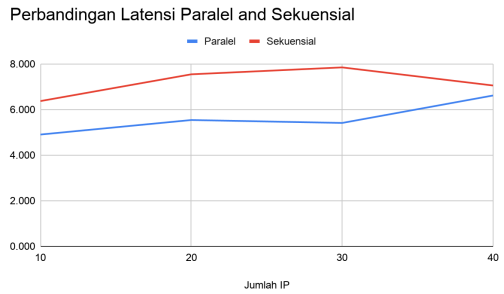
Gambar 4. Antarmuka Manajemen Instance Firewall pada Web Dashboard

B. Hasil Pengujian Fungsional (Black Box)

Pengujian fungsional sistem divalidasi terhadap skenario *Black Box*. Rekapitulasi hasil menunjukkan tingkat keberhasilan fungsional 100% pada penambahan *instance multi-vendor*, validasi token koneksi, penarikan status *instance*, hingga eksekusi pemblokiran IP dinamis. Enkripsi transparan di basis data dan perlindungan *session cookie* divalidasi berjalan lancar.

C. Hasil Pengujian Kinerja Latensi Propagasi

Performa sistem dievaluasi melalui serangkaian pengujian komparatif empiris antara metode eksekusi sekuensial (konvensional) dan paralel (*thread pool*) di bawah variasi ukuran *payload* (10, 20, 30, dan 40 IP) dengan masing-masing skenario diulang sebanyak 30 iterasi. Untuk menjamin validitas data dan menghindari bias akibat fluktuasi waktu sistem operasi (OS *jitter*), pengukuran latensi propagasi dilakukan secara absolut pada level kode menggunakan modul pencatat waktu presisi tinggi (time.monotonic()). Selanjutnya, kestabilan distribusi divalidasi dengan membatasi kecepatan *bandwidth* jaringan (50 KBps, 100 KBps, 150 KBps, dan 200 KBps) menggunakan pembatas TC (*traffic control*).



Gambar 5. Grafik Komparasi Rata-rata Latensi Propagasi Metode Paralel vs Sekuensial

TABEL II. HASIL PENGUJIAN KINERJA METODE EKSEKUSI PARALEL (NORMAL)

Beban IP	Rata-rata (ms)	Min (ms)	Max (ms)	Success Rate
10 IP	4.902	3.171	9.735	100%
20 IP	5.539	3.062	15.438	100%
30 IP	5.411	3.328	9.406	100%
40 IP	6.615	3.125	12.547	100%

TABEL III. HASIL PENGUJIAN KINERJA METODE EKSEKUSI SEKUENSIAL (NORMAL)

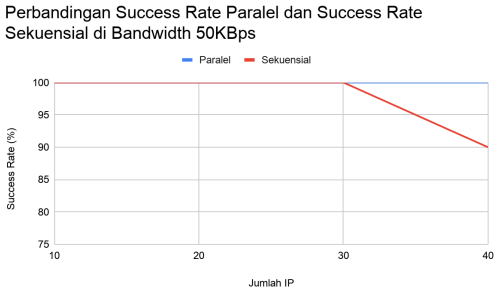
Beban IP	Rata-rata (ms)	Min (ms)	Max (ms)	Success Rate
10 IP	6.370	3.594	11.438	100%
20 IP	7.544	3.469	14.891	100%
30 IP	7.847	3.938	15.922	100%
40 IP	7.053	3.640	17.390	100%

D. Evaluasi Ketahanan pada Kondisi Bandwidth Terbatas

Pengujian pada *bandwidth* terbatas (50 KBps) memperlihatkan keunggulan keandalan metode paralel. Ketika *payload* mencapai 40 IP, metode paralel tetap mampu mempertahankan *success rate* 100% dengan latensi 9,718 ms. Sebaliknya, metode sekuensial mengalami kegagalan transmisi dengan *success rate* menurun hingga 90% (latensi rata-rata melonjak ke 19,937 ms) akibat akumulasi *queueing delay* yang memicu *timeout* koneksi [4].

E. Efisiensi Penggunaan Sumber Daya Komputasi

Selain keunggulan pada aspek kecepatan dan keandalan jaringan, penerapan mekanisme *multithreading* (*thread pool*) terbukti tidak menimbulkan beban yang berlebihan pada kinerja *server*. Data pemantauan sumber daya (*resource profiling*) menunjukkan bahwa selama proses distribusi beban maksimal (40 IP), rata-rata utilisasi prosesor (CPU) sistem hanya berkisar di angka 13-14%, dengan konsumsi memori (RAM) yang sangat stabil di kisaran 120 MB. Hal ini mengonfirmasi bahwa eksekusi paralel pada *Abstraction Layer* memiliki tingkat overhead komputasi yang sangat minim dan aman untuk diimplementasikan pada lingkungan produksi skala penuh tanpa mengganggu service lainnya.



Gambar 6. Grafik Komparasi *Success Rate* Paralel vs Sekuensial pada *Bandwidth* 50 KBps

TABEL IV. REKAPITULASI HASIL PENGUJIAN PADA KONDISI JARINGAN 50 KBps (LATENSI DALAM ms)

Beban IP	Mean P	SD P	SR P	Mean S	SD S	SR S
10 IP	10.093	2.639	100%	12.257	3.897	100%
20 IP	10.349	2.882	100%	12.388	4.367	100%
30 IP	10.583	2.293	100%	10.916	2.731	100%
40 IP	9.718	2.062	100%	19.937	17.072	90%

IV. KESIMPULAN DAN SARAN

Penelitian ini berhasil merancang dan mengimplementasikan *Abstraction Layer* terintegrasi untuk mengorkestrasi propagasi daftar blokir dinamis pada *firewall multi-vendor* pfSense dan OPNsense. Hasil pengujian performa mengonfirmasi keunggulan mutlak metode eksekusi paralel berbasis *thread pool*. Metode paralel mampu mempertahankan latensi rata-rata yang sangat rendah (4,9 ms hingga 6,6 ms) dengan success *rate* 100%. Pada skenario jaringan buruk dengan *bandwidth* 50 KBps dan beban *payload* tinggi 40 IP, metode paralel berhasil menjaga *success rate* 100%, sedangkan metode sekuensial mengalami kegagalan

koneksi hingga *success rate* 90% akibat penumpukan delay.

Untuk penelitian selanjutnya, direkomendasikan perluasan modul *adapter* agar mendukung *platform firewall* komersial terkemuka seperti Fortinet, Palo Alto, atau Cisco ASA. Selain itu, integrasi mekanisme *caching* berbasis Redis pada modul pencarian status koneksi *firewall* dapat dikaji lebih lanjut untuk meminimalkan beban *overhead* API *gateway* selama proses *monitoring live* status.

REFERENSI

[1] Imoukhuede, A.B., Mahmoud, A.H., Sheltami, T.R. dan Barnawi, A.Y. 2025. "Optimization of network device hardening in a multivendor environment". Scientific Reports, 15(1). doi:https://doi.org/10.1038/s41598-025-97894-4.

[2] Costa, M.A., Costa, Y.M., Almeida, Y.O., Cardoso, F.J. dan Gomes, R.L. 2024. "Connection Management Using Automated Firewall Based on Threat Intelligence". LANC ’24: Proceedings of the 2024 Latin America Networking Conference, pp.32–37.

[3] Patel, H. 2024. "A research paper on software design patterns". World Journal of Advanced Engineering Technology and Sciences, 13(1), pp.803–813.

[4] Ivanov, I.G., Hristov, G.V. dan Stoykova, V.D. 2021. "Algorithms for optimizing packet propagation latency in software-defined networks". IOP Conference Series Materials Science and Engineering, 1031(1), pp.012072–012072.

[5] Ahmed, D. dan Amin, R.H. 2022. "Multi-layered firewall to mitigate the impact of Distributed Denial of Service on a network". Passer journal of basic and applied sciences, 4(Special issue), pp.51–62.

[6] Atri, P. 2024. "Enhancing Big Data Security through Comprehensive Data Protection Measures: A Focus on Securing Data at Rest and In-Transit". International journal of computing and engineering, 5(4), pp.44–55.

[7] Chao, C.-S. 2023. "Developing a Feasible Firewall System with Parallel Rule Allocation Optimization for High Service Availability under Large-Scale Network Attacks". IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), pp.56–60.

[8] Damanik, H.A. dan Anggraeni, M. 2025. "Sistem Deteksi dan Pencegahan System Attack pada Infrastruktur Jaringan Menggunakan Realtime Honeypot dan Automatic Iptables". Jurnal Teknologi Informasi dan Ilmu Komputer, 12(6).

[9] Hugo, V. dan Proust, M. 2022. "Integrating Firewalls with SIEM and SOAR Platforms for Automated Threat Response". International Journal of Trend in Scientific Research and Development, 6(3).

[10] Jorepalli, S.K.R. dan Bairy, V. 2024. "Leveraging Network Automation with Python, Terraform, and Ansible to Enhance Security and Operational Efficiency in Large-Scale Networks". ResearchGate.

[11] Kim, T.N., Tri, T.N., Nguyen, L.T. dan Truong, D.T. 2022. "A Combination of the Intrusion Detection System and the Open-Source Firewall Using Python Language". International journal of Computer Networks & Communications, 14(1), pp.59–69.

[12] Kreculj, D., Dihovični, Đ., Ratković Kovačević, N., Gaborov, M. dan Zajeganović, M. 2023. "pfSense Router and Firewall Software". Proceedings of the International Scientific Conference - Sinteza 2023.

[13] Lin, H. 2024. "Journal of Telecommunication System and Management Commentary Challenges and Solutions in Multi-site Networking". doi:https://doi.org/10.37421/2167-0919.2024.13.417.

[14] Sari, R., Cakraningrat, M.S. dan Zain, A.R. 2023. "Implementasi Enkripsi Dekripsi Paket Data pada Rancang Bangun Smart Home Menggunakan Protokol MQTT". MULTINETICS: Jurnal Multimedia Networking Informatics, 8(2), pp.168–176.

[15] Sholihan, A.W.N., Mukti, A.R., Suryayusra, S. dan Dasmen, R.N. 2023. "Implementation of Network Security and Anticipating Attackers Using pfSense Firewall". CESS (Journal of Computer Engineering, System and Science), 8(1), pp.175–183.

[16] Swanzy, P.N., Abukari, A.M. dan Ansong, E.D. 2024. "Data Security Framework for Protecting Data in Transit and Data at Rest in the Cloud". Current journal of applied science and technology, 43(6), pp.61–77.