IMPLEMENTASI SISTEM KEAMANAN DAN HIGH AVAILABILITY PADA CLOUD SERVER MENGGUNAKAN AMAZON WEB SERVICES (AWS)

Andre Febrian Kasmar[1)*], Wahyuna[2)], Fanni Sukma[3)] , dan Sitti Amalia[4)]

[1,2,3]Jurusan Teknologi Informasi, [4]Teknik Elektro
[1,2,3]Politeknik Negeri Padang, [4]Institut Teknologi Padang
Email : andrefebrian@pnp.ac.id [1)], yuna7113@gmail.com [2)], fannisukma@pnp.ac.id [3)], sittiamalia@itp.ac.id [4)]*

Abstract

Service availability and security are critical aspects in web server management, particularly for educational institutions utilizing cloud-based technologies. This study aims to implement a high availability and security system on the SMKN 5 Padang web server using Amazon Web Services (AWS). The methods employed include load balancing using Elastic Load Balancing to evenly distribute workloads, Auto Scaling to dynamically adjust the number of instances based on traffic load, and Wazuh as a security monitoring system to detect and respond to cyber threats such as Denial of Service (DoS) attacks. System performance was evaluated through stress testing and security log analysis. The implementation results indicate that load balancing effectively optimizes network traffic distribution, Auto Scaling maintains service availability during traffic spikes, and Wazuh successfully detects and mitigates security threats in real time. The main contribution of this research is the enhancement of cloud-based web server architecture through a high availability and adaptive security approach, which can be adopted by other educational institutions. Future research is recommended to optimize the system using artificial intelligence to improve proactive cyber threat detection and response efficiency.

Keywords *- AWS, high availability, load balancing, auto scaling, wazuh, cyber security*

Intisari

Ketersediaan layanan dan keamanan merupakan aspek krusial dalam pengelolaan server web, terutama pada institusi pendidikan yang memanfaatkan teknologi berbasis cloud. Penelitian ini bertujuan untuk mengimplementasikan sistem high availability dan keamanan pada server web SMKN 5 Padang menggunakan Amazon Web Services (AWS). Metode yang digunakan meliputi penerapan load balancing (Elastic Load Balancing) dengan tujuan efektivitas distribusi beban menggunakan layanan untuk mendistribusikan beban server secara merata, Auto Scaling berguna menyesuaikan jumlah instance secara dinamis berdasarkan beban lalu lintas, serta Wazuh sebagai sistem pemantauan keamanan untuk mendeteksi dan merespons ancaman siber seperti serangan DoS (Denial of Service). Evaluasi kinerja sistem dilakukan melalui uji beban (stress test) pada web server dan analisis log keamanan. Hasil implementasi menunjukkan bahwa load balancing efektif dalam mengoptimalkan distribusi lalu lintas jaringan, Auto Scaling mampu menjaga ketersediaan layanan secara otomatis saat terjadi lonjakan akses, serta Wazuh berhasil mendeteksi dan memitigasi ancaman keamanan secara real-time. Kontribusi utama penelitian ini adalah penguatan arsitektur server web berbasis cloud dengan pendekatan high availability dan keamanan adaptif yang dapat diadopsi oleh institusi pendidikan lainnya.

Kata Kunci - *AWS, high availability, load balancing, auto scaling, wazuh, keamanan siber*

1. PENDAHULUAN

Cloud computing merupakan paradigma komputasi modern yang menyediakan kapabilitas sumber daya teknologi informasi sebagai layanan melalui internet, tanpa memerlukan pengguna untuk memahami ataupun mengelola infrastruktur yang mendasarinya [1][2][3][4]. Teknologi ini memungkinkan akses bersama terhadap sumber daya komputasi seperti perangkat keras, perangkat lunak, penyimpanan data, dan platform, dengan skema pembayaran berdasarkan pemakaian *(pay-as-you-go)* [5][6][7]. Layanan *cloud* seperti *Amazon Web Services (AWS*) menawarkan keunggulan dalam skalabilitas, fleksibilitas, dan efisiensi biaya, termasuk kemampuan penyimpanan data yang

DOI : https://doi.org/10.21063/jtif.2025.V13.1.40-47

dapat disesuaikan dengan kebutuhan pengguna [5][8][9].

Dalam konteks implementasi *cloud, web server* sering menjadi sasaran serangan siber, khususnya *Denial of Service (DoS)*, yang menyebabkan layanan tidak dapat diakses oleh pengguna sah akibat trafik palsu yang membanjiri jaringan [1][10][11]. Selain ancaman eksternal, tantangan internal seperti ketidakmampuan *server* dalam menangani lonjakan trafik juga menjadi isu krusial dalam pengelolaan layanan berbasis *web* [12][13]. Lonjakan permintaan dapat menyebabkan *overload*, peningkatan waktu respons, hingga *downtime* sistem. Upaya peningkatan kapasitas sumber daya secara manual, seperti penambahan *RAM* atau peningkatan prosesor, bersifat tidak efisien dan berisiko menimbulkan *packet loss* saat proses berlangsung [14]. Untuk mengatasi permasalahan tersebut, diperlukan strategi otomatis dan adaptif yang mampu menyesuaikan kapasitas sistem secara dinamis. Pendekatan seperti *load balancing* dan *Auto Scaling* pada layanan cloud menjadi solusi yang efektif untuk mendistribusikan trafik secara merata, meningkatkan ketersediaan layanan, serta meminimalisasi potensi gangguan akses akibat beban berlebih.

2. METODOLOGI

Penelitian ini menggunakan pendekatan analisis sistem untuk menentukan kebutuhan dalam penerapan sistem keamanan dan ketersediaan tinggi (high availability) pada infrastruktur *Amazon Web Services (AWS*)[15]. Penerapan dilakukan dengan memanfaatkan teknologi cloud computing guna meningkatkan efisiensi *server*, di mana pengguna dapat menyewa kapasitas dan daya komputasi sesuai kebutuhan. Pendekatan ini menghilangkan kebutuhan akan infrastruktur fisik, sehingga lebih hemat biaya dan menawarkan tingkat keamanan yang lebih tinggi[6].

Arsitektur *high availability* dibangun dengan menggunakan layanan *Elastic Load Balancing (ELB)* untuk mendistribusikan beban trafik secara merata ke beberapa *instance EC2*, dengan tujuan mengoptimalkan pemanfaatan sumber daya, memaksimalkan *throughput*, meminimalkan waktu respons, dan menghindari *kelebihan beban pada satu server saja*[16][17][18]. Selain itu, *Auto Scaling* dikonfigurasi agar dapat menyesuaikan jumlah instance secara otomatis berdasarkan kebijakan lalu lintas yang telah ditentukan, sehingga sistem dapat beradaptasi secara dinamis terhadap fluktuasi permintaan pengguna.

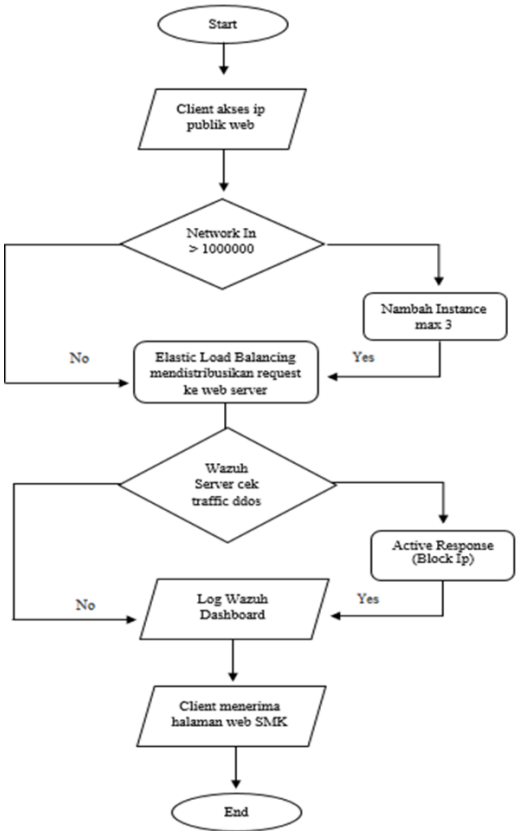
Untuk keperluan pengujian, situs web berbasis native di-deploy pada *instance EC2* dengan sistem operasi *Ubuntu 20.04 LTS[19]*. Layanan yang digunakan adalah *Amazon RDS* dengan *MySQL* untuk menyimpan data dinamis. Dari sisi keamanan, sistem dilengkapi dengan Wazuh, yang berperan sebagai *Host-Based Intrusion Detection System (HIDS)* serta alat monitoring log. Fitur *Active Response* pada Wazuh dikonfigurasi untuk memblokir alamat IP secara otomatis saat terdeteksi aktivitas serangan seperti *Denial of Service (DoS)*.

Pada saat pengguna mengakses halaman web, permintaan akan diterima oleh *ELB* dan secara otomatis dialihkan ke *instance availability dan health* pada *EC2.* Mekanisme ini mencegah terjadinya kelebihan beban pada satu server. Arsitektur ini juga menggunakan beberapa *availability zone* untuk meningkatkan toleransi terhadap kegagalan: ketika satu zona mengalami lonjakan trafik, trafik akan dialihkan ke zona lain yang lebih ringan bebannya. Jika salah satu zona mengalami gangguan, zona lainnya tetap dapat beroperasi untuk menjaga kontinuitas layanan.

Untuk mensimulasikan serangan siber, dilakukan uji coba serangan *DoS* menggunakan teknik *OVH Bypass* yang diarahkan ke web server yang telah diimplementasikan. *Wazuh Agent* yang terpasang pada *server web* akan memonitor pola permintaan yang mencurigakan, kemudian berkoordinasi dengan *Wazuh server* untuk mengidentifikasi dan memblokir IP penyerang secara otomatis menggunakan fitur *active response*. Selama proses ini, *ELB* terus memantau status kesehatan instance EC2. Jika ditemukan *instance* yang gagal atau tidak responsif, ELB akan mengalihkan trafik ke *instance* sehat lainnya. Di sisi lain, *Auto Scaling* akan menambahkan *instance EC2* baru ketika terjadi lonjakan trafik, dan akan mengurangi *instance* saat trafik menurun. Target group digunakan untuk mengatur routing permintaan ke satu atau lebih target berdasarkan aturan yang ditetapkan.

DOI : https://doi.org/10.21063/jtif.2025.V13.1.40-47

Kerangka metodologi ini menunjukkan pendekatan adaptif dan berlapis dalam manajemen server berbasis *cloud*, yang menggabungkan *high availability* dengan deteksi ancaman dan mitigasi secara *real-time*



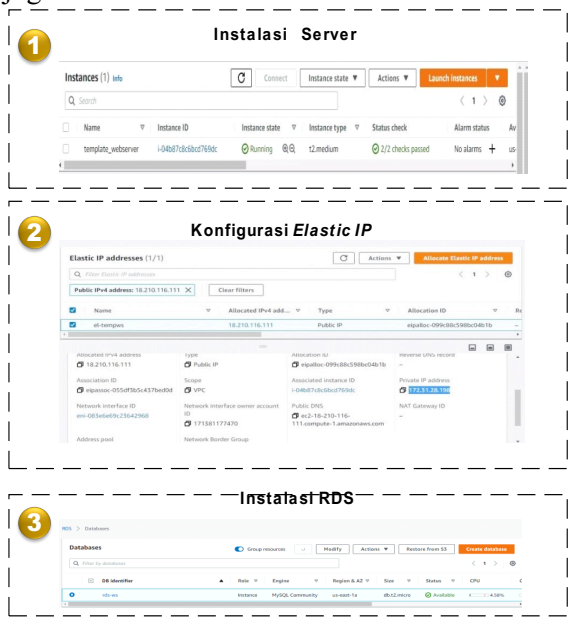
Gambar 1. *Flowchart* Sistem Keamanan dan *High Availiability*

Gambar 1 menggambarkan *flowchart* sistem keamanan dan *high availability web server* yang dibangun menggunakan load balancing dan *Auto Scaling* pada *platform AWS*. Proses dimulai saat klien mengakses alamat *IP web server*. Jika lalu lintas jaringan melebihi ambang batas 1.000.000 *byte* per detik, Auto Scaling akan secara otomatis menambahkan *instance* hingga maksimum tiga instance aktif. *Elastic Load Balancing (ELB)* kemudian akan mendistribusikan permintaan ke setiap instance secara merata. Namun, jika jumlah permintaan belum mencapai ambang batas tersebut, permintaan langsung diteruskan ke ELB untuk diarahkan ke satu instance minimum yang aktif. Selanjutnya, setiap *traffic* yang masuk akan diperiksa oleh sistem keamanan *Wazuh* untuk mendeteksi apakah permintaan tersebut merupakan serangan *DoS (Denial of Service)* atau bukan. Jika terdeteksi sebagai serangan DoS, *Wazuh* secara otomatis akan menjalankan fitur *active response* untuk memblokir alamat IP penyerang. Sebaliknya, jika traffic dianggap aman, *Wazuh* hanya akan mencatat aktivitas tersebut dalam *dashboard log* sebagai referensi pemantauan.

Akhir dari proses ini adalah klien menerima akses halaman *web server* secara normal, yang menunjukkan bahwa sistem berhasil menjaga ketersediaan dan keamanan layanan secara simultan.

3. HASIL DAN PEMBAHASAN

Pada bagian ini, dilakukan analisis terhadap implementasi sistem keamanan dan high availability pada *AWS* Analisis ini mencakup evaluasi performa server, efektivitas distribusi beban menggunakan *ELB*, dan adaptasi kapasitas layanan *web server* menggunakan *Auto Scaling* untuk memastikan ketersediaan layanan yang optimal. Selain itu, efektivitas sistem keamanan *Wazuh* dalam memonitor dan menangani serangan, khususnya serangan *DoS*, juga dianalisis.

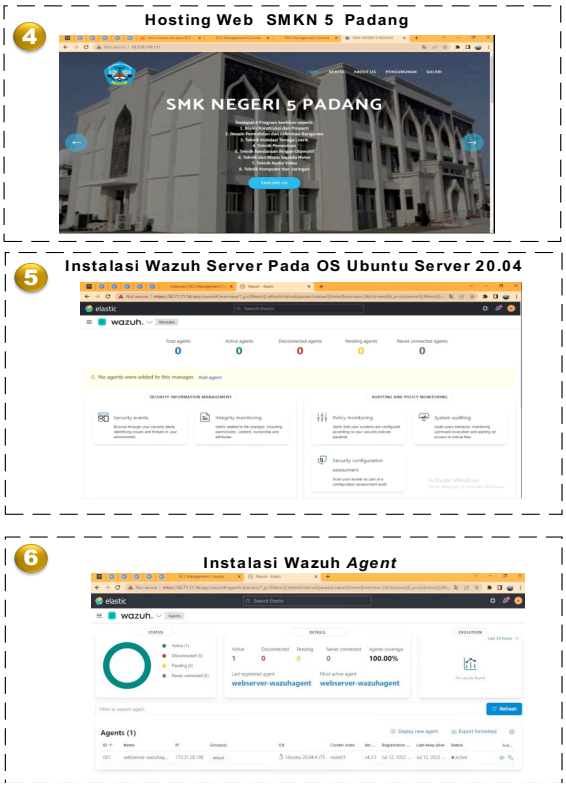


Gambar 2. tahapan utama dalam instalasi

Gambar 2 tersebut menunjukkan tiga tahapan utama dalam instalasi dan konfigurasi layanan AWS untuk membangun infrastruktur *server* yang andal. Pada langkah pertama, *server* diinstalasi menggunakan *Amazon EC2*. Sebuah instance *EC2* telah dibuat dan berjalan dengan status "*Running*", yang menunjukkan bahwa server dalam kondisi aktif dan siap dikonfigurasi lebih lanjut. Kemudian, pada langkah kedua, *Elastic IP* dialokasikan untuk instance EC2. *Elastic IP* ini memungkinkan

DOI : https://doi.org/10.21063/jtif.2025.V13.1.40-47

server memiliki alamat IP statis yang tetap, meskipun instance di-restart, sehingga mempermudah akses publik dan menjaga kestabilan layanan. Langkah ketiga adalah instalasi *Amazon RDS (Relational Database Service)* untuk pengelolaan database, layanan RDS digunakan jenis *MySQL*, yang memungkinkan penyimpanan dan pengelolaan data tanpa memerlukan instalasi perangkat keras tambahan. Ketiga langkah ini – instalasi server, konfigurasi Elastic IP, dan instalasi RDS – merupakan bagian dari penerapan cloud computing untuk membangun infrastruktur server yang efisien, *robust* dan dapat diakses secara global

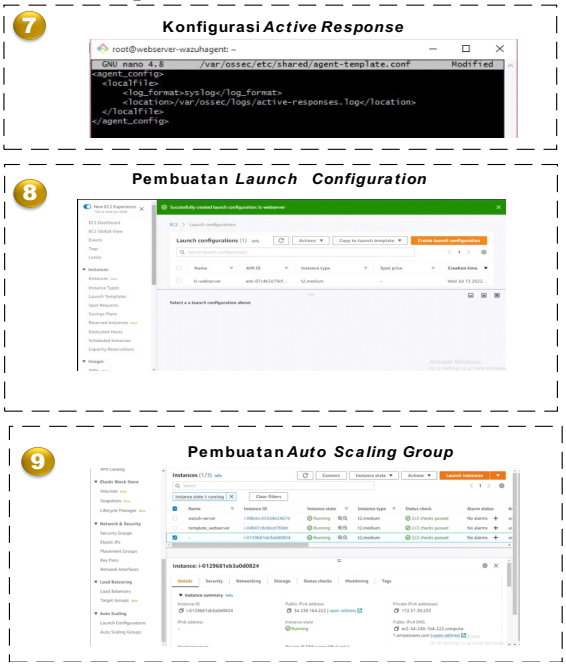


Gambar 3. Tampilan website SMKN

Langkah 3 menunjukkan tampilan website SMKN 5 Padang yang telah dihosting menggunakan platform *AWS*, instalasi *Wazuh server* dan *agent* pada *OS Ubuntu*. Website ini menampilkan informasi yang relevan bagi pengguna, seperti profil sekolah dan layanan yang tersedia. *Hosting* dilakukan untuk memastikan bahwa *website* dapat diakses dengan stabil oleh pengguna, dengan memanfaatkan infrastruktur *cloud* untuk meningkatkan ketersediaan dan keandalan layanan.

Langkah 5 adalah proses instalasi *Wazuh Server* pada *Operating System Ubuntu Server 20.04*. Wazuh Server berfungsi sebagai pusat manajemen keamanan, yang bertanggung jawab untuk memonitor aktivitas yang terjadi di seluruh sistem, menganalisis log dan memberikan perlindungan dengan *active response*. Tampilan pada gambar menunjukkan antarmuka *Wazuh Server*, yang memantau status agen, kejadian keamanan, dan aktivitas lainnya yang relevan untuk menjaga keamanan sistem

Langkah 6 adalah instalasi *Wazuh Agent*, yang berfungsi untuk memonitor setiap aktivitas yang terjadi di *server. Wazuh Agent* diinstal pada server web dan bekerja dengan *Wazuh Server* untuk mendeteksi potensi ancaman atau serangan, seperti serangan DoS. Pada gambar, terlihat bahwa agen *Wazuh* telah terhubung dan aktif, dengan persentase integrasi sebesar 100%, yang menunjukkan bahwa agen berfungsi secara optimal dalam memantau dan melaporkan aktivitas kepada *Wazuh Server*



Gambar 4. proses konfigurasi Active Responsepada Wazuh Agent

Langkah 7 menunjukkan proses konfigurasi *Active Response* pada *Wazuh Agent*. Pada tahap ini, file konfigurasi *agent-template.conf* dimodifikasi untuk menambahkan pengaturan active response. Pengaturan ini berfungsi untuk secara otomatis merespons ancaman keamanan, seperti serangan *DoS*, dengan memblokir IP yang terdeteksi berbahaya. Konfigurasi ini memastikan bahwa Wazuh Agent dapat

DOI : https://doi.org/10.21063/jtif.2025.V13.1.40-47

memberikan tindakan cepat terhadap ancaman yang ditemukan pada *web server*

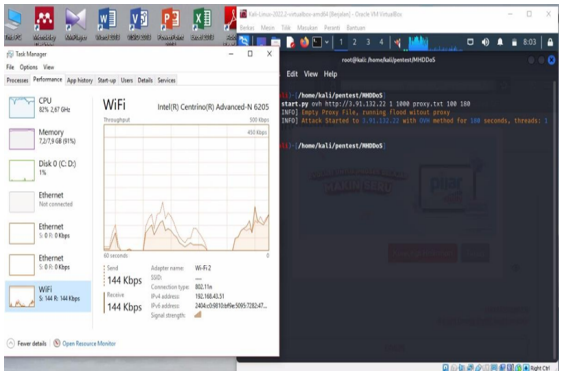
Langkah 8 adalah pembuatan *Launch Configuration* pada *AWS*. *Launch Configuration* adalah template yang digunakan oleh *Auto Scaling* untuk meluncurkan *instance EC2* baru. Pada gambar ini, tampilan menunjukkan bagaimana *Launch Configuration* dibuat dengan spesifikasi yang telah ditentukan, seperti jenis *instance* dan pengaturan jaringan. *Launch Configuration* memastikan bahwa instance yang baru diluncurkan memiliki konfigurasi yang sama dengan *instance* lainnya dalam grup

Langkah 9 menunjukkan pembuatan *Auto Scaling Group*. *Auto Scaling Group* digunakan untuk mengelola sejumlah instance EC2 secara otomatis sesuai dengan beban kerja yang dihadapi. Jika permintaan meningkat, Auto Scaling akan menambah *instance EC2* untuk memenuhi permintaan tersebut, dan sebaliknya, jika permintaan menurun, Auto Scaling akan mengurangi jumlah *instance* untuk mengoptimalkan penggunaan sumber daya. Gambar ini menunjukkan daftar *instance* yang termasuk dalam *Auto Scaling Group*, yang memungkinkan pengelolaan kapasitas secara dinamis untuk menjaga ketersediaan dan performa layanan.

3.1 Pengujian

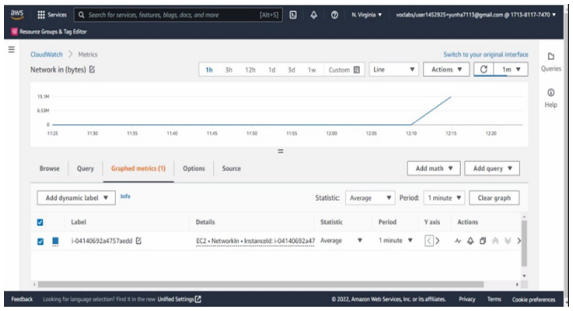
A. Pengujian *Attack, Auto Scaling,* dan *Load Balancing*

Melakukan pengujian dengan memberikan serangan DoS ke website SMKN 5 Padang menggunakan sistem operasi kali linux yang sudah diinstal di Virtuallbox. Pengujian serangan menggunakan *tools Massive HTTP Distributed Denial of Service* (MHDDoS) dengan satu metode yaitu OVH Bypass. Proses cepat turunnya *instance* bergantung pada lonjakan *traffic* dan penyerangan yang dilakukan oleh *Attacker*.



Gambar 5. *Virtualbox*

Hasil analisis selama beberapa metode penyerangan, ada beberapa faktor untuk menggunakan 180 detik pada layanan *auto scalling* yaitu ketika waktu penyerangan dilakukan selama 5 menit akan membuat traffic yang masuk banyak dan mempengaruhi proses cepat turunnya *instance*. Dan ketika waktu penyerangan 2 menit dengan jaringan yang digunakan rata-rata 100kbps, 200kbps, 300kbps dan lainnya membuat *instance* tidak bertambah. Speed internet saat pengiriman *attack web* yaitu rata-rata 100 kbps, data setiap wifi yang dikirim random tergantung speed internet yang dimiliki.



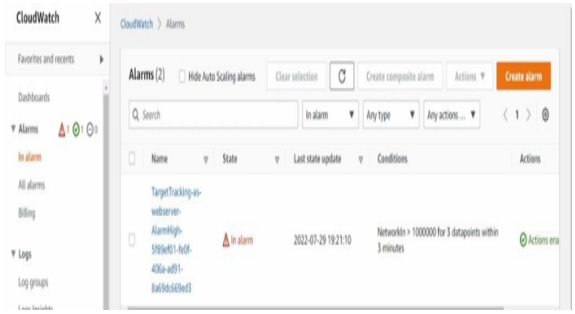
Gambar 6. Hasil Simulasi Pada Jaringan berbeda

Terlihat pada Gambar 6 di atas saat menggunakan jaringan yang berbeda, traffic yang semula berada di angka 0 dalam jangka waktu lima menit sudah naik mencapai 13Mb. Scale out akan terjadi jika Nerwork In keseluruhan telah mencapai lebih dari 1 Mbps. Dengan hasil ini *Auto Scaling* akan bekerja dan *instance* langsung bertambah karena sudah melewati batas metrik untuk setiap *instance* yang sudah ditentukan.

CloudWatch seperti pada gambar 7 di bawah ini merupakan salah satu tools yang disediakan AWS untuk memonitoring, pada Gambar 7 *CloudWatch* sedang memonitoring Network In,

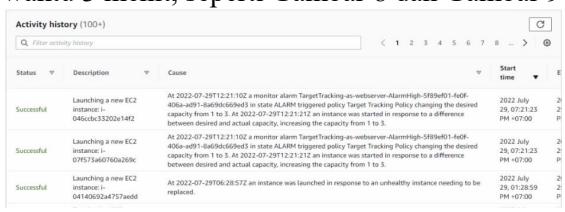
DOI : https://doi.org/10.21063/jtif.2025.V13.1.40-47

apapun traffic yang masuk ke *instance* akan di log, apakah itu traffic saat mengakses website, healthy check, maupun traffic DoS. Jika tidak ada traffic yang masuk maka cloudwatch tidak akan memberikan informasi apapun

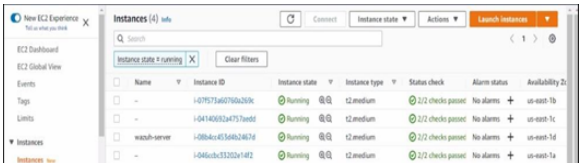


Gambar 7. *CloudWatch*

Cloudwatch alarm memberi informasi bahwa jaringan yang masuk sudah lebih dari 1000000 *byte* yang akan menambah *instance* dalam waktu 3 menit, seperti Gambar 8 dan Gambar 9



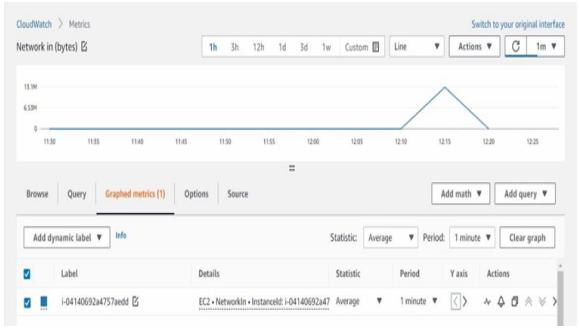
Gambar 8. *CloudWatch*



Gambar 9. Hasil *CloudWatch*

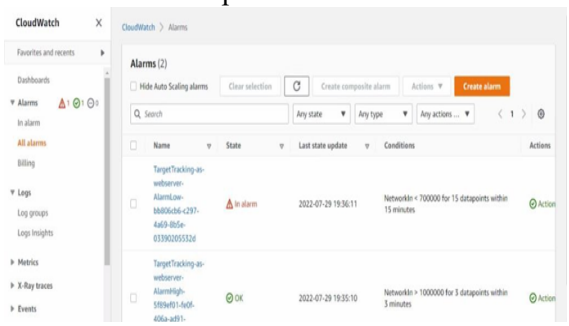
Pengujian *load balancer* dilakukan dengan cara mengakses domain *load balancer* dan menganalisa apakah scale out dan scale in bekerja pada jumlah permintaan yang masuk

Setelah tidak ada lagi proses serangan *DoS*, permintaan yang masuk ke *web server* akan berkurang dan akan membuat traffic menurun. Proses penurunan *instance* akan memakan waktu beberapa menit karena akan melalui proses *draining* terlebih dahulu sesuai dengan kondisi. Lama proses draining tergantung banyak traffic yang diterima semakin besar jaringan yang masuk semakin lama proses draining-nya



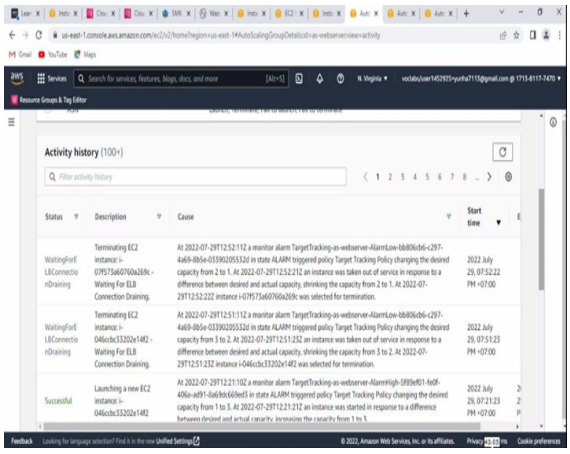
Gambar 10. *Traffic*

Pada Gambar 10 terlihat bahwa *traffic* yang masuk sudah mengalami penurunan otomatis *Auto Scaling* akan bekerja kembali dan melakukan proses *scale in*. akan memberikan alarm bahwa data kurang dari 700000 *byte* per second, waktu yang dibutuhkan untuk menurunkan *instance* adalah 15 menit dari waktu alarm ditampilkan.



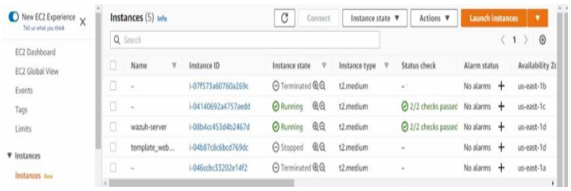
Gambar 11. *CloudWatch*

Aktivitas pengurangan *instance* dapat dilihat pada Gambar 11 pada status proses penurunan *instance* akan memalui proses tunggu untuk load balance draining setelah beberapa saat *instance* akan *terminate* seperti pada Gambar 12



Gambar12. *Load Balance*

DOI : https://doi.org/10.21063/jtif.2025.V13.1.40-47



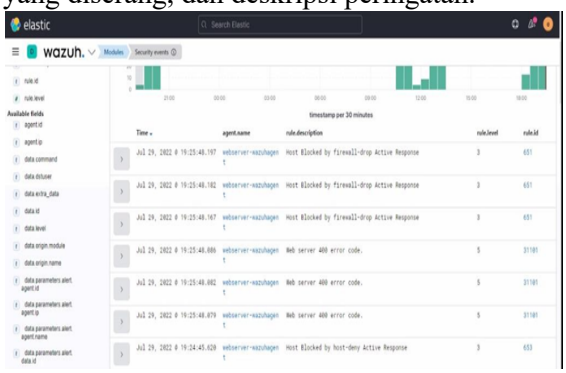
Gambar 13. *Server Instance*

Dalam pengujian ini *server instance* seperti pada gambar 13 di atas tidak akan mengalami *overload* karena beberapa faktor seperti spesifikasi *instance*, jaringan internet yang digunakan, seberapa banyak paket yang dikirimkan, dan spesifikasi laptop yang digunakan, untuk detail faktor yang mempengaruhi hal tersebut telah dijelaskan pada point diatas. Pada pengujian ini *instance* menggunakan spesifikasi. medium dengan kapasitas jaringan mencapai 1Gbps, agar *server* ini dapat *overload* harus menggunakan jaringan atau mengirimkan paket yang melebihi kapasitas dari spesifikasi *instance* tersebut.

Tidak ada angka yang spesifik pada pengujian ini, Walaupun menggunakan angka yang sama untuk proses penyerangan seperti mengirim paket 1000 *thread* 100 *timeout* 180 detik, naik turunnya *instance* dan berapa data yang diperoleh tergantung kepada speed internet yang digunakan. Semakin cepat *speed* internetnya, maka semakin tinggi *traffic* yang masuk ke server.

B. Pengujian *Active Response* di *Wazuh Dashboard*

Melihat *log* di ketika *web server* diserang DoS. Dari informasi yang tampil pada *Wazuh dashboard* terlihat waktu kejadian, nama agent yang diserang, dan deskripsi peringatan.



Gambar 14. *Wazuh dashboard*

Dari gambar 14 diatas dapat dilihat detail informasi dari *web server 400 error code*. Setelah *Wazuh* mengidentifikasi permintaan yang masuk untuk mengatasi supaya server tidak mengalami penyerangan berulang maka active response akan melakukan pemblokiran IP penyerang dalam waktu yang sudah di konfigurasi dari 60, 120, dan 180 menit. Dalam rentang waktu tersebut penyerang masih bisa melakukan attack Dos hanya saja data yang dikirim tidak akan masuk ke *web server*.

4. KESIMPULAN DAN SARAN

Penerapan sistem keamanan dan high availability pada *server website* menggunakan *AWS*, beberapa kesimpulan dapat diambil. Pertama, penggunaan metode *load balancing* merupakan solusi yang tepat untuk menangani beban kerja yang tinggi dan menghindari *overload* pada salah satu server. *Load balancing* mendistribusikan beban *traffic* secara seimbang ke dua atau lebih *server* yang tersedia, sehingga server dapat bekerja secara optimal dan mencegah adanya titik kegagalan tunggal. Selain itu, *Auto Scaling* membantu menjaga kinerja server tetap stabil dan andal dengan menyesuaikan kapasitas secara otomatis sesuai kebutuhan. *Auto Scaling* secara otomatis akan menambah atau mengurangi jumlah *instance* berdasarkan beban kerja, sehingga memastikan aplikasi selalu tersedia.

Selanjutnya, *monitoring server* menggunakan Wazuh memungkinkan log aktivitas server agen terkirim secara otomatis ke *Wazuh Manager*. Hal ini memudahkan administrator dalam memantau aktivitas server tanpa harus melakukan pengecekan langsung ke server agen. Selain itu, fitur *active response* di Wazuh mampu memblokir IP penyerang *DoS* yang membanjiri traffic *web server*. Tindakan ini menghasilkan log aktivitas error dan *firewall-drop* di *Wazuh Manager*, yang memberikan informasi rinci kepada administrator untuk menganalisis dan mengambil tindakan pencegahan terhadap serangan selanjutnya

DAFTAR PUSTAKA

[1] T. Singh, “The Effect of Amazon Web Services (Aws) on,” Vol. 10, No. 11, Pp. 480–482, 2021.

[2] P. Borra, “Comparison and Analysis of Leading Cloud Service Providers (Aws , Azure And Gcp),” Vol. 15, No. 3, Pp. 266–278, 2024.

DOI : https://doi.org/10.21063/jtif.2025.V13.1.40-47

[3] W. Setiawan, N. Fajriyah, T. Duha, U. M. Tangerang, and U. Nias, "Analisa Layanan Cloud Computing Di Era Digital," Vol. 1, No. 1, 2022.
[4] B. G. Tabish Mufti, Pooja Mittal, "A Review on Amazon Web Service (AWS), Microsoft Azure & Google Cloud A Review on Amazon Web Service (AWS), Microsoft Azure Google Cloud Platform (Gcp) Services," No. March, 2021, Doi: 10.4108/Eai.27-2-2020.2303255.
[5] P. K. Illa, V. Naresh, and S. Kodukula, "Web Hosting in The Cloud : Key Considerations For The Deployment of Web Applications To Attain High-Availability , Scalability And Reliability In The Cloud," Vol. 1, Pp. 33–42, 2012.
[6] S. W. 3 M Lintang Ashshofa Walmarwah , Akhyar Lubis "Implementasi Cloud Computing Menggunakan Platform Aws Pada Website Rumah Kue Havivya Medan," Vol. 19, No. 1, Pp. 134–147, 2024.
[7] E. Kurniawan, "Penerapan Teknologi Cloud Computing di Universitas Studi Kasus : Fakultas Teknologi Informasi Ukdw," 2011.
[8] M. S. Rumetna, "Pemanfaatan Cloud Computing Pada Dunia Bisnis : Studi Literatur The Utilization of Cloud Computing in The World of Business :," Vol. 5, No. 3, Pp. 305–314, 2018, Doi: 10.25126/Jtiik.201853595.
[9] L. Sugiyanta, W. Adinda, and N. Rakhmah, "Analisis Perbandingan Antara Colocation Server Dengan Amazon Web Services (Cloud) Untuk Usabilitas Portal S Wa . Co . Id Di Pt . Swa Media Bisnis) Avalaiable At : Avalaiable At :," Vol. 1, No. 1, Pp. 58–63, 2017.
[10] H. S. Ahmad Ashari, "Cloud Computing : Solusi Ict ?," No. 0711, Pp. 336–345, 2011.
[11] S. R. P. 5 M Mangayarkarasi , S Tamil Selvan , R Kuppuchamy , S Shanthi 4 "Highly Scalable and Load Balanced Web Server on Aws Cloud," 2021, Doi: 10.1088/1757-899x/1055/1/012113.
[12] M. S. Mubarok and M. I. Herdiansyah, "Implementasi Cloud Computing Amazon Web Services (AWS) Pada Web Reservasi Kamar Hotel," Vol. 4, No. 2, Pp. 698–708, 2023, Doi: 10.30865/Klik.V4i2.1212.
[13] A. F. Kasmar, "Analysis Delay on Load Balancing NDN , Case Study : Palapa Ring," 2017.
[14] H. Fryonanda *Et Al.*, "Analisis Kinerja Linux Pada Mini Pc: Kinerja Prosesor, Memori Dan Disk Drive," No. September, Pp. 1–13, 2023.
[15] R. P. Mukti Qamal1, Defry Hamdhana2, "Website Media Pembelajaran online Amazon Web Services," Vol. 11, No. 2, Pp. 319–327, 2019.
[16] R. Pratama, A. Lubis, S. Wahyuni, U. Pembangunan, And P. Budi, "Rancang Bangun Sistem Load Balancer Dengan Layanan Cloud Amazone Web Services Design Of Application Load Balancer With Cloud Computing Services - Amazon Web Services," Vol. 5, 2022.
[17] M. Shafira, H. Amnur, And R. Afyenni, "Load Balancing Menggunakan Algortima Round Robin Dengan Stickiness Pada Aws," Vol. 2, No. 4, 2021.
[18] A. Wibowo, A. Virgono, R. Latuconstina, F. T. Elektro, And U. Telkom, "Load Balancing Pada Cloud Computing Menggunakan Metode Least Connection," Vol. 5, No. 3, Pp. 6210–6217, 2018.
[19] Md. Abdul Kadar, "Website Hosting Using Amazon Web Services For Internet Of Things Applications," No. Figure 1, Pp. 1–5, 2020.

DOI : https://doi.org/10.21063/jtif.2025.V13.1.40-47