Implementasi *Web Application Firewall* (WAF) pada Aplikasi Fishku Berbasis *Google Cloud Armor*

Nabila Apriliana Widiyono[1], Unan Yusmaniar Oktiawati[1,*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
nabilaapriliana@mail.ugm.ac.id
[*]Korespondensi: unan_yusmaniar@ugm.ac.id;

Abstract – The security of Website Applications has become a pressing issue in an increasingly complex digital era. This research aims to implement Google Cloud Armor security services on the Google Cloud Platform to protect the "Fishku" Website Application, specifically against three types of attacks: Local File Inclusion (LFI), Vulnerability Scanner, and Protocol Attack. Website Application security plays a crucial role, given the rising threats of cyberattacks that can jeopardize the integrity and confidentiality of user data. This study employs the "Fishku" Website Application as the test subject. Testing is conducted before and after the implementation of Google Cloud Armor, using a laptop as the tool and the Kali Linux operating system installed on VirtualBox to evaluate the effectiveness of the protection provided. Furthermore, this research involves configuring a Load Balancer and utilizing Alerts features to detect potential attacks. Comprehensive system performance monitoring is conducted through data metric analysis. The research results indicate that Google Cloud Armor successfully shields the "Fishku" Website Application from these attacks, highlighting its effectiveness in enhancing security and system performance. The implications of this research are significant in the context of developing secure Website Applications, with Google Cloud Armor as a viable solution. In conclusion, this study offers valuable insights into the necessity of safeguarding Website Applications and demonstrates how Google Cloud Armor can be a solution to combat cyber threats. The broader implications of these research findings can serve as a foundation for future developments in Website Application security.

Keywords – Fishku Application, Security System, Google Cloud Armor, Google Cloud Platform

Intisari – Keamanan Aplikasi *Website* menjadi isu mendesak dalam era digital yang semakin kompleks. Penelitian ini bertujuan untuk menerapkan layanan keamanan *Google Cloud Armor* pada *Google Cloud Platform* dalam melindungi Aplikasi *Website* Fishku, khususnya terhadap 3 (tiga) jenis serangan *Local File Inclusion (LFI)*, *Vulnerability Scanner*, dan *Protocol Attack*. Keamanan Aplikasi *Website* memiliki peran yang sangat penting mengingat meningkatnya serangan siber yang dapat mengancam integritas dan kerahasiaan data pengguna. Penelitian ini menggunakan Aplikasi *Website* "Fishku" sebagai subjek uji coba. Pengujian dilakukan sebelum dan setelah penerapan *Google Cloud Armor*, dengan menggunakan laptop sebagai alat dan sistem operasi *Kali Linux* yang telah dilakukan instalasi pada *VirtualBox* untuk menguji keberhasilan perlindungan yang diberikan. Selain itu, penelitian ini juga melibatkan konfigurasi *Load Balancer* dan pemanfaatan fitur *Alerts* untuk mendeteksi serangan potensial. Analisis data *metric* juga dilakukan untuk memantau kinerja sistem secara lebih komprehensif. Hasil penelitian menunjukkan bahwa *Google Cloud Armor* berhasil melindungi Aplikasi *Website* "Fishku" dari serangan tersebut dan mengungkapkan keberhasilan perlindungan dalam keamanan dan kinerja sistem. Dampak dari penelitian ini penting dalam konteks pengembangan Aplikasi *Website* yang aman, dengan *Google Cloud Armor* sebagai solusi yang layak dipertimbangkan. Dalam kesimpulannya, penelitian ini memberikan pandangan penting mengenai perlunya perlindungan pada Aplikasi *Website* dan memberikan pandangan bagaimana *Google Cloud Armor* dapat menjadi solusi dalam mengatasi ancaman siber. Implikasi lebih luas dari hasil penelitian ini dapat membentuk landasan untuk perkembangan keamanan Aplikasi *Website* di masa depan.

Kata kunci – Aplikasi Fishku, Sistem Keamanan, *Google Cloud Armor, Google Cloud Platform*

I. PENDAHULUAN

Di era disrupsi dan era revolusi industri 4.0 perkembangan teknologi aplikasi dalam dunia bisnis menuntut manusia selalu untuk berkembang dan berinovasi. Teknologi internet dan dunia digital mengalami disrupsi yang begitu cepat dan canggih yang semua itu membantu dalam kehidupan manusia [1]. Sistem yang dikembangkan para ilmuwan pasti mengalami keterbatasan atau kelemahan. Hal inilah yang menjadikan manusia untuk selalu berpikir dalam mengatasi masalah yang dihadapi. Setiap teknologi yang diciptakan pasti memiliki kelemahan. Kelemahan atau kekurangan dalam produk teknologi mendorong manusia untuk dapat memperbaiki dengan inovasi sehingga meminimalisir kelemahan tersebut. Kelemahan atau kekurangan dalam produk teknologi mendorong manusia untuk dapat memperbaiki dengan inovasi, sehingga meminimalisir kelemahan tersebut.

Keamanan atau *security* sebuah aplikasi harus diperhatikan oleh setiap pemilik, instansi atau perusahaan agar tehindar dari kejahatan. Masalah keamanan atau gangguan sudah terjadi dan bertebaran di internet seperti serangan *Malware*, eksploitasi, injeksi *database* dan sebagainya. Menurut [2] pada Tahun 2016, terdapat sekitar 90% kejahatan internet yang menyerang aplikasi *web* dengan serangan yang paling terkenal yaitu dengan cara menginjeksi *database* dengan capaian 47,06%. Kerusakan maupun kebocoran data ini akan mengancam setiap saat dengan meningkatnya kemahiran atau kepandaian sumber daya manusia. Dalam mengatasi masalah pengamanan *web* atau aplikasi dari serangan *hacker* dapat dilakukan dengan cara *self-test*. *Self-test* yaitu suatu pengujian yang dilakukan terhadap *web* server secara legal dengan aktivitas menyerupai *hacker*. [2] menyatakan *self test* dapat dilakukan dengan beberapa metode *penetrastion testing* seperti *Information System Security Assesment Framework (ISSAF)*, *Open Web Aplication Security Project (OWASP)* versi4 dan *Open Source Security Testing Methodologi Manual (OSSTMM)*. Dalam Penelitian ini akan dilakukan uji keamanan terhadap ancaman aplikasi *web* Fishku dengan metode *Web Application Firewall (WAF)* [3].

Dalam konteks inovasi di dunia aplikasi, aplikasi "Fishku" hadir sebagai sebuah *startup e-commerce* di bidang perikanan. Aplikasi ini bertujuan untuk mempermudah kegiatan jual beli ikan serta mendeteksi kesegaran ikan menggunakan teknologi *machine learning*. Fishku merupakan hasil kerja tim dari peneliti yang juga merupakan bagian dari tim pengembang aplikasi ini. Aplikasi Fishku diciptakan melalui Program Bangkit Academy 2022 (Google, GoTo, Traveloka) yang merupakan bagian dari Studi Independen Kampus Merdeka Kemendikbudristek. Fishku tersebut berhasil mendapatkan pendanaan dari Dikti dan Google serta penghargaan dalam *Top 15 Best Capstone Project-Bangkit Academy*.[4].

Latar belakang ini mendorong peneliti untuk mengembangkan sistem keamanan yang berjudul "Implementasi *Web Application Firewall (WAF)* pada aplikasi Fishku berbasis *Google Cloud Armor*". Dengan kata lain, tujuan utama untuk memastikan bahwa layanan keamanan yakni *Google Cloud Armor* yang digunakan mampu memberikan perlindungan yang lebih baik terhadap serangan dan potensi ancaman yang mungkin terjadi pada aplikasi *web* tersebut.

II. DASAR TEORI

A. Keamanan Aplikasi *Web*

Keamanan dalam aplikasi *web* merupakan suatu isu yang memerlukan perhatian serius. Terdapat beberapa solusi yang dapat dilakukan untuk menerapkan layanan keamanan pada aplikasi *web*. Meskipun hal ini tidak sepenuhnya sempurna, tetapi tindakan preventif telah diambil untuk menghindari hal- hal yang tidak diinginkan. Salah satu solusi yang dapat diterapkan adalah dengan mengimplementasikan *Web Application Firewall* (WAF). WAF berfungsi untuk memeriksa lalu lintas data paket dari aplikasi *web* dan juga dapat berfungsi untuk memblokir beberapa serangan pada aplikasi *web* [5]. Berdasarkan daftar OWASP Top Ten 2021, tiga serangan teratas adalah *Broken Access Control*, *Cryptographic Failures*, dan *Injection* [6]. Ketiga serangan ini, seperti yang sudah umum diketahui, merupakan serangan terhadap target pada server *web*.

B. *Web Application Firewall (WAF)*

Web Application Firewall (WAF) adalah perangkat lunak yang bertugas memantau aliran lalu lintas antara aplikasi *web* dan internet. *Firewall* diletakkan di sisi server yang menganalisis setiap paket data yang masuk. Hanya paket- paket data yang dianggap aman yang diteruskan ke server, sedangkan semua paket data yang diidentifikasi sebagai berpotensi membahayakan akan ditolak oleh *firewall*. Meskipun diklaim sebagai metode perlindungan keamanan, WAF tidak selalu mampu memberikan perlindungan total terhadap segala bentuk serangan yang mungkin terjadi pada aplikasi [7].

C. *Cloud Armor*

Cloud Armor adalah layanan keamanan yang disediakan oleh *Google Cloud Platform* untuk melindungi aplikasi dan layanan *cloud* dari serangan DDoS dan serangan *web* lainnya. Layanan ini menggunakan infrastruktur global Google untuk memberikan pertahanan secara besar-besaran terhadap serangan DDoS. Cloud Armor terintegrasi dengan *Load Balancer* HTTP(S) global dan memblokir lalu lintas berdasarkan alamat IP atau rentangnya. Layanan ini dapat digunakan untuk mencegah pengguna atau lalu lintas yang berbahaya agar tidak mengakses sumber daya, atau yang lebih buruk lagi, mengambil alih kendali dari VPC berdasarkan aturan yang telah ditetapkan. Mode pratinjau memungkinkan pengguna menganalisis pola serangan tanpa mengganggu pengguna reguler [8].

D. *Local File Inclusion*

Local File Inclusion (LFI) adalah sebuah kerentanan yang memungkinkan seorang penyerang untuk menyisipkan atau memasukkan berkas menggunakan karakter khusus (seperti "../") ke dalam server melalui peramban *web*. Kerentanan ini terjadi ketika sebuah aplikasi *web* memasukkan berkas ke dalam proses tanpa melakukan proses pembersihan atau penyaringan input dengan benar. Akibatnya, hal ini memberikan peluang bagi penyerang untuk memanipulasi input yang dimasukkan dan menyisipkan muatan yang

memungkinkan akses ke direktori lain *(Path Ttraversal)* serta menyertakan berkas lain dari server *web* itu sendiri [9].

E. *Vulnerability Scanner*

Vulnerability scanner adalah alat atau perangkat lunak yang digunakan untuk menemukan dan menganalisis celah keamanan dalam sistem atau aplikasi. Tujuannya adalah untuk mengidentifikasi potensi kerentanan yang bisa dimanfaatkan oleh penyerang, sehingga langkah-langkah pencegahan dapat diambil sebelum kerusakan terjadi. *Scanner* ini biasanya digunakan untuk mencari kerentanan keamanan seperti *Cross-site scripting, SQL Injection, Command Injection, Path Traversal,* dan konfigurasi server yang tidak aman [10].

F. *Protocol Attack*

Protocol Attack adalah serangan yang memanfaatkan celah dalam protokol komunikasi. Salah satu jenis serangannya adalah HTTP *splitting*. Dalam serangan ini, penyerang memisahkan permintaan HTTP menjadi dua bagian dengan karakter CR *(carriage return)* dan LF *(line feed)* yang dimasukkan ke dalam *header* HTTP yang tidak divalidasi. Ini memungkinkan penyerang untuk mengendalikan *header* dan *body* dari *respons* HTTP yang diterima dari aplikasi *web*. Akibatnya, server atau aplikasi bisa memberikan *respons* yang tidak diharapkan, dan ini dapat memfasilitasi serangan seperti XSS atau pencurian datasensitif oleh penyerang [11].

G. *Log-Based Alerts*

Membuat kebijakan *alerting* berbasis metrik digunakan untuk melacak data metrik yang dikumpulkan *Cloud Monitoring* untuk memberikan notifikasi ketika ambang batas peristiwa dan metrik tercapai. Kebijakan ini dapat memiliki satu atau lebih kondisi untuk memicu *alerting* dan akan membuat insiden yang terlihat di konsol *Cloud Monitoring*. Ada beberapa tindakan yang dapat dipilih untuk menangani insiden terbuka, seperti mengakui insiden sebagai masalah yang diketahui, menyaring kondisi yang terkait, melihat kebijakan *alerting*, dan mengedit kebijakan *alerting*. Dalam hal notifikasi, konfigurasi dapat dilakukan untuk mengirimkan email, SMS, dan berbagai bentuk pemberitahuan lainnya. Untuk memberikan panduan mengenai langkah-langkah yang dapat diambil untuk menangani situasi, tautan menuju dokumentasi juga dapat disertakan dalam peringatan tersebut [8].

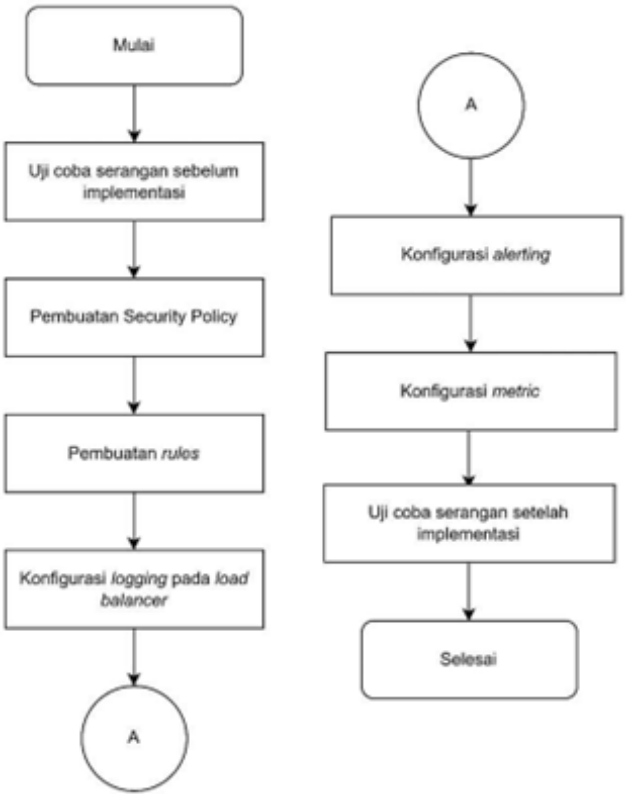
H. *Log-Based Metric*

Log memiliki potensi untuk digunakan sebagai dasar pembuatan metrik. Dalam *Cloud Logging*, *log* dapat dikumpulkan sesuai dengan filter yang ditetapkan setiap kali ada kecocokan pola tertentu. Informasi yang dihasilkan dari *log* ini kemudian akan diungkapkan dalam Monitoring dan bisa dimanfaatkan lebih lanjut untuk membuat tampilan informasi dan kebijakan peringatan. Sebagai ilustrasi, *log* yang berisi pesan kesalahan 404 tertentu dapat dihitung selama satu menit dan diperlihatkan sebagai ukuran atau metrik. Metrik berbasis *log* ini bisa termasuk metrik sistem, yang telah ditetapkan sebelumnya oleh *Cloud Logging*, atau metrik yang dibuat oleh pengguna berdasarkan proyek tertentu dengan mengatur kriteria penyaringan tertentu [8].

III. METODOLOGI

A. Tahapan Teknik

Tahapan teknik merupakan alur teknis dari penelitian ini. Tahapan teknik dalam penelitian ini disajikan pada Gambar 1.



Gambar 1.Tahapan teknik penelitian

Tahapan ini dimulai dengan melakukan uji coba serangan sebelum implementasi, yang melibatkan simulasi serangan untuk mengidentifikasi potensi kerentanan pada aplikasi *web* fishku. Selanjutnya, langkah pembuatan *security policy* dilakukan, di mana aturan- aturan keamanan yang mengatur lalu lintas aplikasi diformulasikan. Proses pembuatan *rules* dilanjutkan untuk mengidentifikasi pola serangan yang harus diantisipasi oleh WAF. Pada tahap konfigurasi *logging* pada *load balancer*, parameter pengumpulan data *log* diatur untuk mengawasi lalu lintas dan aktivitas aplikasi secara mendalam. Langkah selanjutnya adalah mengonfigurasi *alerting*, di mana mekanisme pemberitahuan otomatis diaktifkan saat terdeteksi aktivitas yang mencurigakan.

Pengaturan *metric* juga dilakukan untuk mengukur dan menganalisis performa serta keamanan aplikasi. Tahapan akhir melibatkan uji coba serangan setelah implementasi, di mana pengujian ulang dilakukan untuk menguji efektivitas *Web Application Firewall* dalam menanggapi serangan yang sebelumnya telah diidentifikasi. Selama seluruh tahapan ini, langkah-langkah teknis ini mengarah pada implementasi yang kuat dan lebih aman untuk aplikasi Fishku, dengan menggunakan *Web Application Firewall* berbasis *Google Cloud Armor*.

B. Aplikasi Fishku

Aplikasi Fishku merupakan *platform e-commerce* perikanan yang menonjolkan fitur pendeteksian kesegaran ikan melalui teknologi *machine learning*. Interaksi antarmuka aplikasi dirancang untuk mempermudah koneksi antara nelayan dan pembeli, menciptakan pengalaman yang lancar dan efisien. Aplikasi ini juga menghadirkan layanan deteksi kesegaran ikan sebagai solusi untuk memastikan kualitas produk yang dijual. Aplikasi Fishku beroperasi di lingkungan *Google Cloud Platform* (GCP) dengan pendekatan *serverless* yang mengoptimalkan efisiensi dan skalabilitas. Saat ini, Aplikasi Fishku terakreditasi status SLSA 3 yang menandakan *platform* sumber dan lingkungan memenuhi standar khusus untuk menjamin kemampuan perlindungan yang jauh lebih kuat terhadap gangguan dibandingkan level sebelumnya dengan mencegah kelompok ancaman tertentu, seperti kontaminasi *cross-build*. menggunakan layanan *Cloud Run* untuk menjalankan aplikasi sesuai permintaan tanpa mengurus infrastruktur fisik. Pengelolaan basis data dan informasi penting dikelola oleh *Cloud SQL* dan *Cloud Storage*, menawarkan keandalan dan skalabilitas dalam manajemen data. Keamanan aplikasi dijaga melalui pemantauan aktif dan konfigurasi tepat menggunakan *GCP Console*, memastikan integritas data. Kolaborasi teknologi dan inovasi mendasari lahirnya Aplikasi Fishku, sebuah solusi *e-commerce* aman dan handal dalam industri perikanan

C. Implementasi Pembuatan *Security Policy*

Dalam lingkungan *Google Cloud Platform* (GCP), terutama dengan menggunakan fitur *Cloud Armor*, langkah-langkah untuk membuat kebijakan keamanan adalah sebagaiberikut

1. Masuk ke panel *Cloud Armor* dan pilih opsi 'Buat Kebijakan Keamanan.'
2. Beri nama kebijakan yang akan dibuat dan tambahkan deskripsi kebijakan tersebut. Pilih jenis kebijakan sebagai 'Kebijakan keamanan *backend*' sesuai dengan kebutuhan. Selanjutnya, atur tindakan *default* ketika permintaan tidak sesuai dengan aturan yang telah ditetapkan. Atur tindakan *default* menjadi 'deny' dan kode *respons* menjadi '403 *Forbidden*.' Klik 'Lanjut' untuk melanjutkan konfigurasi kebijakan keamanan.
3. Buat aturan *default* pertama untuk mengizinkan akses dari semua alamat IP. Pilih 'Mode Dasar', masukkan karakter '*' pada kolom 'Cocok', dan pilih 'Izinkan'. Klik 'Selesai' untuk menyelesaikan konfigurasi aturan.
4. Klik 'Selesai' untuk meninjau ringkasan dan melanjutkan ke langkah selanjutnya.
5. Tambahkan target, klik 'SELESAI' dan 'BUAT KEBIJAKAN.'
6. Kebijakan Keamanan berhasil dibuat.

Untuk informasi lebih lanjut, klik nama kebijakan,. Tindakan ini akan membawa ke rincian lebih lanjut mengenaikebijakan yang telah dibuat. Selain itu, Kebijakan Keamanan ini akan menerapkan aturan-aturan keamanan pada *web load balancer*, sehingga melindungi situs *web* dari lalu lintas berbahaya.

D. Pembuatan *Rules Block Local File Inclusion*

Proses konfigurasi untuk mengatasi serangan LFI dapat dilakukan melalui serangkaian langkah sebagai berikut. Langkah pertama, menuliskan deskripsi jenis *rules* yang akan diterapkan di sini penulis menambahkan deskripsi "*Block Local File Inclusion*". Kemudian, menggunakan fitur *Advanced Mode* dalam konfigurasi. Pada bagian ini, menambahkan aturan dengan menggunakan *syntax* khusus yang ditunjukkan pada Gambar 2.

```
evaluatedPreconfiguredExpr('lfi-stable')
```

Gambar 2. *Syntax* pembuatan *rules block local file inclusion*

Setelah menambahkan *syntax*, langkah berikutnya adalah memilih tindakan yang akan diambil jika serangan terdeteksi. Dalam hal ini, memilih tindakan "Deny" untuk menolak permintaan yang sesuai dengan kondisi serangan LFI juga akan mengatur kode *respons* menjadi "403 *Forbidden*" untuk menunjukkan bahwa akses ditolak. Terakhir, perlu menentukan prioritas aturan. Prioritas ini menentukan urutan eksekusi aturan dalam daftar. Dalam contoh ini, aturan LFI akan diberi prioritas 9000 untuk memastikan bahwa ia diterapkan dengan benar.

E. Pembuatan *Rules Block Vulnerability Scanner*

Proses implementasi membuat *rules* adalah menuliskan deskripsi yakni "*Block Vulnerabiliy Scanner*". Selanjutnya, pilih *mode advanced* dalam pengaturan, dan gunakan *syntax* spesifik yaitu ditunjukkan pada Gambar 3.

```
evaluatedPreconfiguredExpr('scannerdetectio
n-stable')
```

Gambar 3. *Syntax* pembuatan *rules block vulnerability scanner*

Syntax ini akan mengaktifkan kondisi yang telah dipersiapkan sebelumnya oleh *platform* keamanan untuk mendeteksi serangan dari *vulnerability scanner*. Setelah kondisi ditambahkan, langkah berikutnya adalah memilih tindakan yang akan diambil jika serangan terdeteksi. Pilih tindakan "*Deny*" untuk menolak permintaan yang sesuai dengan kondisi serangan dari *vulnerability scanner*. Tidak hanya itu, perlu dilakukan pengaturan kode *respons* menjadi "403 *Forbidden*" untuk menunjukkan bahwa akses ditolak. Terakhir, tentukan prioritas aturan. Prioritas ini menentukan urutan eksekusi aturan dalam daftar. Dalam contoh ini, aturan untuk menghadapi serangan dari *vulnerability scanner* akan diberi prioritas 9001 untuk memastikan bahwa aturan ini diterapkan dengan benar setelah aturan lainnya.

F. Pembuatan *Rule Block Protocol Attack*

Di dalam *Google Cloud Armor*, prosedur konfigurasi adalah sebagai berikut. Pertama, mengisikan deskripsi yaitu "Block *Protocol Attack*". Dalam *mode advanced*, gunakan *syntax* spesifik yang ditunjukkan pada Gambar 4.

```
evaluatedPreconfiguredExpr('protocolattack-
stable')
```

Gambar 4. *Syntax* pembuatan *rule block protocol attack*

Setelah kondisi ditambahkan, langkah selanjutnya adalah memilih tindakan saat serangan terdeteksi. Pilih opsi "Deny" untuk menolak permintaan sesuai kondisi serangan *protocol attack*. Tetapkan juga kode *respons* menjadi "403 *Forbidden*" agar terlihat bahwa akses ditolak. Lalu, atur prioritas urutan dalam daftar. Pada contoh ini, berikan prioritas 9003 pada aturan yang ditetapkan untuk menghadapi serangan *protocol attack*, agar aturan ini diterapkan dengan benar setelah aturan lainnya.

G. Konfigurasi *Load Balancer*

Mengaktifkan pada *load balancer* adalah langkah pertama untuk mengumpulkan *log*, dan kemudian modifikasi *log query* di *Google Cloud Armor* adalah langkah berikutnya untuk menganalisis dan memfilter *log* yang telah dikumpulkan. Kombinasi kedua langkah ini membantu mengamankan lingkungan sistem dengan mendapatkan. Pada bagian *log query* modifikasi *query* tersebut seperti pada Gambar 5.

```
ANDjsonPayload.enforcedSecurityPolicy.
configuredAction:(DENY)
```

Gambar 5. Konfigurasi *load balancer*

Dengan melakukan konfigurasi ini, *load balancer* akan mulai mencatat dan mencatat aktivitas serta trafik yang melintasi *backend*. Langkah ini memungkinkan untuk memantau dan menganalisis data *logging* guna mendapatkan wawasan yang lebih baik tentang penggunaan layanan, serta potensi ancaman atau masalah keamanan. Pada tahap selanjutnya, data *logging* yang telah terkumpul akan dapat digunakan untuk analisis lebih lanjut dan pengambilan tindakan yang sesuai

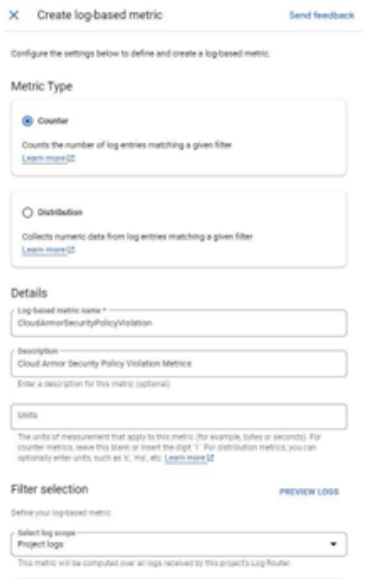
H. Pembuatan *Rules Log-Based Alerts*

Konfigurasi *alerting* pada *Google Cloud Armor* bertujuan untuk menerima notifikasi atau pemberitahuan ketika kebijakan keamanan yang telah dibuat dalam Cloud Armor mendeteksi serangan atau aktivitas yang mencurigakan. Pada langkah ini, perlu memastikan bahwa kueri *log* yang telah dimasukkan benar dan sesuai dengan kebutuhan. Dalam konteks pengaturan keamanan *Cloud Armor*, langkah pertama adalah mengonfigurasi notifikasi *log*. Dapat menentukan frekuensi notifikasi dan opsi penutupan otomatis untuk memudahkan manajemen notifikasi. Setelah konfigurasi selesai, simpan kebijakan notifikasi yang telah dibuat. Selanjutnya, konfigurasi notifikasi ke alamat email dengan mengedit saluran pemberitahuan. Masukkan alamat email dan nama yang sesuai. Terakhir, kembali ke konfigurasi kebijakan dan pilih alamat email yang telah ditambahkan pada saluran pemberitahuan. Dengan mengatur *alerting* ini, dapat memantau aktivitas pada layanan HTTP *Load Balancer*. Notifikasi akan dikirimkan melalui email saat ada aktivitas mencurigakan atau potensi serangan pada layanan aplikasi *web*. Ini memungkinkan *respons* yang cepat dan perlindungan sistem dari ancaman yang mungkin terjadi.

I. Konfigurasi *Load-Based Metric*

Pada konfigurasi *metric* dalam *Google Cloud Armor*, langkah-langkahnya melibatkan pengaturan metrik untuk memonitor aktivitas keamanan pada layanan *Load Balancer*.Dalam hal ini, dapat memanfaatkan *Log Explorer* dengan membuat kueri *log* spesifik yang menganalisis aktivitas terkait dengan kebijakan keamanan yang memiliki tindakan penolakan akses (*DENY*). Berikut adalah konfigurasi yang diperlukan disajikan pada Gambar 6. Bagian ini untuk verifikasi *kueri log*. Setelah semua selesai, maka Langkah terakhir adalah klik "*Create Metric*".

Dengan mengonfigurasi metrik ini, memiliki pemantauan aktif terhadap aktivitas yang berkaitan dengan kebijakan keamanan yang diterapkan pada *Load Balancer*. Ini membantu mengidentifikasi potensi ancaman atau insiden keamanan dengan lebih cepat, sehingga, dapat me*respons* dengan tindakan yang tepat untuk melindungi layanan aplikasi *web*.



Gambar 6. Konfigurasi *log-based metric*

J. Pengujian Penelitian

1. Uji Coba Serangan *Local File Inclusion*

Serangan *Local File Inclusion* (LFI) adalah teknik serangan di mana penyerang mencoba memanipulasi URL dengan mengirimkan parameter yang mengarahkan ke file sistem yang seharusnya tidak dapat diakses oleh pengguna. Langkah-langkah pengujian sebagai berikut :

a. Menggunakan perintah 'curl' di *Kali Linux* dengan perintah yang ditunjukkan pada Gambar 7.

```
$ curl -Ii (INPUT BASE
URL)/loadImage?filename=../../../
etc/passw d
```

Gambar 7. Perintah 'curl' di *Kali Linux* untuk uji coba serangan *local file inclusion*

b. Tujuan dari perintah ini adalah untuk menguji apakah sistem atau server memiliki kerentanan terhadap serangan yang melibatkan manipulasi direktori atau *path traversal*. *Path traversal* adalah serangan di mana penyerang mencoba mendapatkan akses ke direktori atau file di luar jalur normal yang diizinkan.

c. Aplikasi Fishku akan memproses permintaan dan memberikan *respons* sesuai dengan kebijakan keamanan yang diimplementasikan oleh *Google Cloud Armor*.

Dengan melaksanakan uji serangan ini nantinya akan mengevaluasi *respons* dan perlindungan yang diberikan oleh *Google Cloud Armor* terhadap serangan *Local File Inclusion* (LFI). Apakah *Google Cloud Armor* dapat mencegah akses yang tidak sah ke direktori sistem yang sensitif melalui serangan LFI. Dari hasil uji coba ini, dapat menilai keberhasilan perlindungan yang diberikan oleh*Google Cloud Armor* terhadap serangan LFI.

2. Uji Coba Serangan *Vulnerability Scanner*

Serangan *Vulnerability Scanner* adalah upaya untuk mengidentifikasi kerentanan dalam sebuah aplikasi atau sistem. Sebelumnya aplikasi Fishku telah diimplementasikan perlindungan Google Cloud Armor. Langkah-Langkah Uji Coba :

a. Menjalankan perintah di *Kali Linux* menggunakan 'curl' seperti yang ditunjukkan pada Gambar 8.

```
$ curl -Ii (INPUT BASE
URL) -H "User- Agent:
```

Gambar 8. Perintah 'curl' di *Kali Linux* untuk uji coba serangan *vulnerability scanner*

b. Dalam perintah ini, menggunakan utilitas cURL untuk membuat permintaan HTTP ke URL. Opsi"-I" memerintahkan cURL hanya untuk mengambil *header respons* HTTP dan tidak mengunduh isi asli dari halaman. Opsi "-i" memungkinkan untuk melihat tampilan lengkap *header respons* dalam hasil, yang mencakup informasi tentang status *respons*, jenis konten, dan lainnya. Menggunakan opsi "-H" untuk mengatur *header* khusus dalam permintaan. *Header* yang diatur adalah "User-Agent: nikto", yang mengidentifikasi bahwa permintaan berasal dari alat pengujian keamanan Nikto. Tujuan dari uji coba ini adalah untuk melihat bagaimana sistem me*respons* serangan dari alat pengujian keamanan yang dikenal dan apakah ada kerentanan yang dapat dieksploitasi.

c. Permintaan ini akan diarahkan ke aplikasi Fishku yang telah diimplementasikan perlindungan oleh *Google Cloud Armor*.

Melalui pengujian penelitian ini, bertujuan untuk mengidentifikasi apakah *Google Cloud Armor* dapat mendeteksi dan menghalangi serangan dari alat *scanning* seperti Nikto. Dengan melakukan ini, dapat mengukur keberhasilan perlindungan *Google Cloud Armor* dalam menghadapi serangan yang mencoba untuk mengidentifikasi kerentanan pada aplikasi.

3. Uji Coba Serangan *Protocol Attack*

Pada bagian ini, menjelaskan uji coba yang akan dilakukan terhadap aplikasi Fishku dengan menggunakan teknik serangan *Protocol Attack* setelah menerapkan *Google Cloud Armor* sebagai lapisan keamanan tambahan. Serangan *protocol attack* melibatkan upaya untuk memanipulasi protokol komunikasi antara klien dan server. Dalam skenario ini, menyimulasikan serangan dengan menggunakan perintah *curl* yang dimodifikasi untuk mengganti karakteristik protokol HTTP. Langkah – Langkah uji coba serangan sebagai berikut:

a. Menjalankan perintah di *Kali Linux* menggunakan *curl* seperti yang ditunjukkan pada Gambar 9.

```
$ curl -Ii "(INPUT BASE URL)/index.html?foo=advanced%0d%0aContent-
Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-
Type:%20text/html%0d%0aContent-
Length:%2035%0d%0a%0d%0a<html>Sorry,%20System%20Down</html>"
```

Gambar 9. Perintah di *Kali Linux* untuk uji coba serangan *protocol attack*

b. Dalam perintah *curl* tersebut, karakter *newline* (%0d%0a) adalah representasi URL:-encoded dari karakter *newline* (CR-LF) yang disisipkan dalam URL. Ini bisa digunakan untuk mencoba memanipulasi bagaimana server-server yang berinteraksi dengan permintaan tersebut akan me*respons* dengan cara yang tidak diharapkan atau bahkan berpotensi terkena kerentanan keamanan. Potensi hasil dari manipulasi semacam ini adalah mungkin terjadi perbedaan dalam interpretasi protokol antara server *web* dan server *frontend* atau *proxy*, yang dapat dimanfaatkan oleh penyerang.

c. Permintaan ini akan diarahkan ke aplikasi Fishku yang telah diimplementasikan perlindungan oleh *Google Cloud Armor*.

Melalui pengujian ini bertujuan untuk menguji apakah *Google Cloud Armor* dapat mendeteksi dan menghalangi serangan yang berusaha memanipulasiprotokol komunikasi antara klien dan server. Serta mengetahui apakah perlindungan ini berhasil dalam mencegah serangan yang mencoba memanipulasi protokol untuk mendapatkan akses yang tidak sah ke server.

IV. HASIL DAN PEMBAHASAN

Hasil dalam bab ini, akan dibahas secara mendalam mengenai implementasi *Web Application Firewall* (WAF) pada aplikasi Fishku berbasis *Google Cloud Armor*. Penelitian ini mengarah pada tujuan utama yaitu melindungi aplikasi Fishku dari serangan-serangan berbahaya seperti *Local File Inclusion* (LFI), *Vulnerability Scanner*, dan *Protocol Attack*. Selain itu, telah dilakukan konfigurasi yang melibatkan penggunaan *load balancers*, penerapan *log-based alerts*, dan pengukuran metrik untuk mendukung upaya keamanan:

A. Hasil Uji Coba Serangan tanpa *Google Cloud Armor*

Dalam bab ini, akan dibahas secara mendalam mengenai implementasi *Web Application Firewall* (WAF) pada aplikasi Fishku berbasis *Google Cloud Armor*. Penelitian ini mengarah pada tujuan utama yaitu melindungi aplikasi Fishku dari serangan-serangan berbahaya seperti *Local File Inclusion* (LFI), *Vulnerability Scanner*, dan *Protocol Attack*. Berikut adalah Tabel 1 hasil uji serangan sebelum implementasi *security policy*.

Tabel 1. Hasil uji serangan sebelum implementasi *Google Cloud Armor*

Jenis Serangan	***Web Application Firewal***		**Keterangan**
	Pertama (tanpa *Google Cloud Armor*)	**Kedua (dengan *Google Cloud Armor*)**	
Local File Inclusion (LFI)	Berhasil	-	*Syntax* dikirimkan kepada server dan hasil respons HTTP/2 200 OK
Vulnerability Scanner	Berhasil	-	*Syntax* dikirimkan kepada server dan hasil respons HTTP/2 200 OK
Protocol Attack	Berhasil	-	*Syntax* dikirimkan kepada server dan hasil respons HTTP/2 200 OK

1. Hasil Serangan *Local File Inclusion* tanpa *Google Cloud Armor*

   Serangan LFI menjadi fokus penelitian untuk mengidentifikasi potensi kerentanan dalam aplikasi web Fishku. Berikut ini adalah hasil yang diperoleh dari pengujian serangan LFI sebelum adanya penerapan kebijakan keamanan.

   Pada pengujian serangan LFI, dilakukan pemanfaatan celah pada mekanisme keamanan aplikasi web Fishku guna mencoba mengakses file sensitif di dalam server. *Payload* digunakan untuk mencoba mengakses file *password* yang terletak di direktori /etc pada server Fishku.

   Proses ini bertujuan untuk mengidentifikasi kemampuan aplikasi dalam mencegah akses tidak sah ke file sensitif. Jika respons HTTP yang diterima adalah 200 OK, maka ini mengindikasikan keberhasilan *payload* dalam melakukan *traversal* direktori dan mengakses file/*etc/password*. Temuan ini mengungkap potensi kelemahan dalam keamanan yang bisa dieksploitasi oleh penyerang untuk memperoleh informasi sensitif [9]. Hasil serangan LFI sebelum penerapan *Google Cloud Armor* memiliki dampak serius terhadap keamanan, karena penyerang bisa mendapatkan akses ke file-file sensitif seperti *password*. Hal ini membuka risiko potensial terhadap serangan lebih lanjut pada sistem. Temuan respons HTTP 200 OK juga menunjukkan bahwa aplikasi belum memiliki mekanisme keamanan yang memadai untuk melindungi dari serangan LFI. Dengan adopsi *Google Cloud Armor*, diharapkan bahwa kelemahan ini dapat diminimalkan, dengan sistem mendeteksi dan mencegah serangan LFI sebelum mencapai aplikasi.

2. Hasil Serangan *Vulnerability Scanner* tanpa *Google Cloud Armor*

   Hasil uji serangan menggunakan *Vulnerability Scanner* sebelum penerapan kebijakan keamanan menunjukkan adanya potensi celah dalam mekanisme keamanan aplikasi. Pada pengujian ini, pengiriman permintaan dengan *header "User-Agent:* nikto" dilakukan untuk mengevaluasi kerentanan aplikasi terhadap serangan tertentu. Respons HTTP 200 OK yang diterima menunjukkan bahwa kebijakan keamanan pada saat itu belum mampu mendeteksi dan mencegah akses dari alat *Vulnerability Scanner*. Temuan ini mengindikasikan celah dalam pertahanan sistem yang memungkinkan alat seperti Nikto mendapatkan respons yang diinginkan [12].

   Respons HTTP 200 OK dalam pengujian ini tidak sesuai dengan harapan, mengindikasikan bahwa mekanisme keamanan pada sistem belum cukup kuat untuk melawan serangan otomatis atau dari alat seperti Nikto. Hal ini menandakan bahwa kebijakan keamanan yang ada tidak dapat dalam mendeteksi dan mencegah serangan dari alat tersebut. Dalam pengujian ini menunjukkan urgensi untuk mengambil langkah-langkah perlindungan yang lebih kuat dan memadai, termasuk implementasi *Google Cloud Armor*, untuk melindungi aplikasi dari ancaman serangan yang memanfaatkan celah keamanan.

3. Hasil Serangan *Protocol Attack* tanpa *Google Cloud Armor*

   Hasil uji serangan *Protocol Attack* sebelum penerapan kebijakan keamanan menunjukkan bahwa protokol keamanan server tidak mampu mendeteksi dan mencegah upaya manipulasi protokol HTTP dengan karakteristik tertentu.

   Dalam pengujian ini, serangan dilakukan dengan mengirimkan permintaan manipulatif yang berisi karakteristik yang dapat memicu respons yang tidak semestinya. karakteristik manipulatif seperti %0d%0a digunakan untuk memanipulasi *header* permintaan HTTP [11]. Respons yang diharapkan seharusnya adalah kode status HTTP 200 OK, yang menunjukkan bahwa manipulasi karakteristik protokol berhasil.

   Namun, hasil uji serangan ini mengindikasikan bahwa protokol keamanan pada server tidak mampu mendeteksi dan mencegah tindakan manipulasi karakteristik protokol tersebut. Respons yang diterima adalah kode status HTTP 200 OK, yang seharusnya tidak seharusnya terjadi dalam kondisi normal. Temuan ini memberikan catatan penting tentang potensi celah dalam mekanisme keamanan server terhadap manipulasi protokol HTTP.

B. Hasil Uji Coba Serangan dengan *Google Cloud Armor*

Pada sub bab ini, akan diuraikan hasil dari uji coba serangan setelah dilakukan implementasi kebijakan keamanan menggunakan *Google Cloud Armor* pada aplikasi Fishku. Hasil yang diperoleh akan dianalisis untuk mengevaluasi perlindungan yang diberikan oleh *Google Cloud Armor* terhadap serangan-serangan yang telah direncanakan. Berikut ini Tabel 2 merupakan hasil uji serangan sesudah implementasi *security policy*.

Tabel 2. Hasil uji serangan sesudah implementasi *Google Cloud Armor*

Jenis Serangan	***Web Application Firewal***		**Keterangan**
	Pertama (tanpa *Google Cloud Armor*)	**Kedua (dengan *Google Cloud Armor*)**	
Local File Inclusion (LFI)	-	Berhasil	*Syntax* dikirimkan kepada server dan hasil respons HTTP/2 403
Vulnerability Scanner	-	Berhasil	*Syntax* dikirimkan kepada server dan hasil respons HTTP/2 403
Protocol Attack	-	Berhasil	*Syntax* dikirimkan kepada server dan hasil respons HTTP/2 403

1. Hasil Serangan *Local File Inclusion* dengan *Google Cloud Armor*

Pengujian dilakukan dengan menjalankan perintah pada lingkungan Kali Linux yang memanfaatkan celah LFI pada aplikasi Fishku. Dalam implementasi *Google Cloud Armor*, terdapat mekanisme yang memungkinkan penilaian terhadap ekspresi keamanan tertentu. Pada kasus ini, terdapat suatu aturan (*rules*) atau kondisi yang menggambarkan potensi serangan LFI dengan nama atau kode *lfi-stable* dalam konfigurasi *Google Cloud Armor* [13]. Evaluasi ekspresi tersebut kemungkinan memeriksa apakah ada karakteristik dari URL yang mencoba melakukan *traversal* direktori seperti ../../../, dan jika ada, sistem akan menolak akses dengan respons HTTP 403 *Forbidden*. Respons HTTP 403 *Forbidden* pada serangan LFI setelah implementasi *Google Cloud Armor* mengindikasikan bahwa aturan-aturan keamanan yang diatur dalam konfigurasi *Google Cloud Armor* telah berhasil mengidentifikasi dan mencegah upaya serangan LFI. Implikasinya, kebijakan keamanan yang diimplementasikan melalui *Google Cloud Armor* telah memberikan perlindungan yang kuat terhadap potensi celah LFI dan mampu mendeteksi serta mencegah serangan sebelum mencapai aplikasi *web* Fishku. Hal ini berdampak positif pada peningkatan tingkat keamanan keseluruhan dari aplikasi web Fishku [8].

2. Hasil Serangan *Vulnerability Scanner* dengan *Google Cloud Armor*

Pada sub bab ini, akan diuraikan hasil dan pembahasan dari uji coba serangan *Vulnerability Scanner* setelah implementasi kebijakan keamanan menggunakan *Google Cloud Armor*. Uji coba ini bertujuan untuk mengidentifikasi kemampuan kebijakan keamanan dalam menghadapi serangan yang dilakukan oleh alat otomatis seperti *Vulnerability Scanner,* yang dalam kasus ini akan diwakili oleh alat "Nikto".Web.

Pengujian dilakukan dengan menjalankan perintah pada lingkungan Kali Linux yang memanfaatkan alat "*curl*" untuk mengirimkan permintaan HTTP dengan *header "User-Agent*: Nikto". Dalam perintah ini, flag-1 digunakan untuk hanya mengambil *header* respons dari server, sedangkan flag-H digunakan untuk menambahkan *header* khusus ke dalam permintaan HTTP. *Header* yang ditambahkan dalam contoh ini adalah "*User-Agent*: Nikto", yang mengubah informasi pengenal alat yang melakukan permintaan menjadi "Nikto".

Alat "Nikto" adalah sebuah alat *open-source* yang digunakan untuk mengidentifikasi kerentanan potensial pada aplikasi web [12]. Hasil respons dari server menunjukkan status kode 403 (*Forbidden*), yang mengindikasikan bahwa sistem telah berhasil mencegah upaya akses oleh alat "Nikto". Penerapan mekanisme keamanan yang tepat dalam konfigurasi Gogle Cloud Armor mampu mendeteksi karakteristik dari alat "Nikto" dan mencegah upaya pemindaian atau penetrasi yang tidak sah [8].

3. Hasil *Protocol Attack* dengan *Google Cloud Armor*

Pada uji coba ini, menggunakan serangan *Protocol Attack* yang berfokus pada memanipulasi protokol komunikasi dengan memasukkan karakter khusus seperti %0d%0a yang merepresentasikan karakter *Carriage Return* (CR) dan *Line Feed* (LF). Serangan ini mencoba memanipulasi protokol untuk menciptakan respons palsu dari server. Uji coba dilakukan dengan menjalankan perintah pada lingkungan Kali Linux yang memanfaatkan alat "*curl*" untuk mengirimkan permintaan HTTP yang memasukkan karakter %0d%0a ke dalam parameter "*foo*". kode status HTTP 403 menunjukkan bahwa akses ditolak (*Forbidden*). Dalam konteks ini, serangan *Protocol Attack* yang menggunakan teknik HTTP *Splitting* mencoba memanipulasi karakteristik protokol HTTP dengan menyisipkan karakter %0d%0a untuk menciptakan respons palsu yang diinginkan [14]. Namun, mekanisme keamanan yang diimplementasikan dalam *Google Cloud Armor* berhasil mendeteksi anomali ini dan menolak akses dengan respons HTTP 403

Forbidden. Dengan demikian, hasil pengujian ini memberikan bukti bahwa implementasi kebijakan keamanan dalam *Google Cloud Armor* mampu melindungi aplikasi web Fishku dari serangan *Protocol Attack* yang berupaya memanipulasi protokol komunikasi [8]. Perlindungan ini memiliki dampak positif dalam meminimalkan risiko potensial dari serangan semacam itu, sehingga integritas dan keamanan aplikasi web tetap terjaga dengan baik.

C. Hasil Perbandingan Sebelum dan Sesudah Implementasi *Google Cloud Armor*

Pada pengujian ini telah melakukan serangkaian uji coba serangan *Local File Inclusion* (LFI), *Vulnerability Scanner*, dan *Protocol Attack* sebelum dan sesudah implementasi *Google Cloud Armor*. Hasil pengujian ini menunjukkan bahwa sebelum implementasi *Google Cloud Armor* pengujian serangan memberikan respons HTTP 200 OK yang menandakan bahwa ketiga jenis serangan tersebut diterima oleh server. Hal ini menunjukkan bahwa kemampuan keamanan Aplikasi Web Fishku yang masih rentan terhadap ancaman serangan. Sebaliknya, setelah implementasi *Google Cloud Armor* dan melakukan konfigurasi aturan *security policy*. Serangan LFI, *Vulnerability Scanner*, dan *Protocol Attack* telah berhasil diatasi. Respons HTTP 403 *Forbidden* menandakan bahwa kebijakan keamanan *Google Cloud Armor* berhasil mendeteksi dan mencegah akses dari serangan-serangan tersebut. Implementasi *Google Cloud Armor* memberikan perlindungan yang efektif terhadap potensi ancaman keamanan dan meningkatkan tingkat keamanan aplikasi web Fishku secara keseluruhan. Berikut adalah Tabel 3 hasil perbandingan dari pengujian.

Tabel 3. Hasil perbandingan sebelum dan sesudah implementasi *Google Cloud Armor*

Pengujian Serangan	**Tanpa *Google Cloud Armor***		**Dengan *Google Cloud Armor***	
	Diterima	**Ditolak**	**Diterima**	**Ditolak**
LFI (1)	200	0	0	403
Vulnera (1)	200	0	0	403
Protocol Attact (1)	200	0	0	403
LFI (2)	200	0	0	403
Vulnera (2)	200	0	0	403
Protocol Attact (2)	200	0	0	403

D. Hasil *Log-Base Alerts*

Pada sub-bab ini, akan dijelaskan hasil dari analisis implementasi *alerting* yang telah dilakukan setelah penerapan *security policy* pada *Google Cloud Armor*. Tujuan dari analisis ini adalah untuk mengidentifikasi bagaimana sistem *alerting* dapat mendeteksi dan memberikan pemberitahuan terkait aktivitas mencurigakan atau serangan yang terjadi pada aplikasi web Fishku. Sebelum membahas hasil analisis, berikut adalah konfigurasi *alerting* yang telah diterapkan:

1. *Log Query*: Digunakan untuk mencari pola aktivitas yang mencurigakan pada log.
2. *Alert Policy*: Kebijakan yang mendefinisikan kriteria yang harus terpenuhi agar sebuah pemberitahuan atau notifikasi dapat dikirimkan.

Alerting melalui email memiliki peran krusial dalam menjaga keamanan dan ketersediaan sistem. Notifikasi *real-time* memungkinkan deteksi dini terhadap masalah atau ancaman, memungkinkan tindakan respons segera sebelum situasi memburuk. Dalam aspek keamanan, notifikasi seketika melalui email membantu mengidentifikasi dan menangani serangan atau upaya penetrasi dengan cepat. Dengan demikian, *alerting* melalui email adalah alat penting dalam menjaga kontrol penuh terhadap lingkungan teknologi dan mengambil langkah proaktif untuk menjaga kinerja dan keamanan yang optimal.

F. Hasil *Log-Based Metric*

Melalui penggunaan *Metric Explorer*, grafik dan visualisasi dapat digunakan untuk memahami pola dan tren yang muncul saat serangan terjadi. Dalam skripsi ini, hasil dari pengujian serangan LFI, *Vulnerability Scanner*, dan *Protocol Attack* akan dianalisis melalui metrik yang telah dihasilkan pada Gambar 10 sebagai berikut.

Gambar 10. Hasil grafik *metric monitoring*

Dalam gambar grafik metrik yang telah disajikan. Sebagai tambahan, bahwa dalam *Metric Explorer* ini memiliki opsi untuk mengunduh file CSV yang memuat data yang diambil dalam grafik ini. Dengan memanfaatkan fitur ini, dapat mengakses data yang digunakan dalam penghitungan metrik tersebut. Berikutnya, akan menyajikan hasil dari data respons serangan yang tercatat dalam *log metric* yang dimuat pada Gambar 11 berikut ini.

Waktu	Hasil	Keterangan
Wed Aug 23 2023 09:58:00	0	Tidak ada aktivitas yang terdeteksi
Wed Aug 23 2023 09:59:00	0.05	**Terdeteksi aktivitas dengan nilai respons 0.05 (5% dari skala respons)**
Wed Aug 23 2023 10:00:00	0.05	**Terdeteksi aktivitas dengan nilai respons 0.05 (5% dari skala respons)**
Wed Aug 23 2023 10:01:00	0	Tidak ada aktivitas yang terdeteksi
Wed Aug 23 2023 10:02:00	0	Tidak ada aktivitas yang terdeteksi
Wed Aug 23 2023 10:14:00	0	Tidak ada aktivitas yang terdeteksi
Wed Aug 23 2023 10:15:00	0.016666666666666666	**Terdeteksi aktivitas dengan nilai respons 0.0167 (1.67% dari skala respons)**
Wed Aug 23 2023 10:16:00	0.016666666666666666	**Terdeteksi aktivitas dengan nilai respons 0.0167 (1.67% dari skala respons)**
Wed Aug 23 2023 10:17:00	0	Tidak ada aktivitas yang terdeteksi pada interval ini.
Wed Aug 23 2023 10:18:00	0.05	**Terdeteksi aktivitas dengan nilai respons 0.05 (5% dari skala respons)**
Wed Aug 23 2023 10:19:00	0	Tidak ada aktivitas yang terdeteksi pada interval ini.
Wed Aug 23 2023 10:20:00	0	Tidak ada aktivitas yang terdeteksi pada interval ini.
Wed Aug 23 2023 10:29:00	0	Tidak ada aktivitas yang terdeteksi pada interval ini.
Wed Aug 23 2023 10:30:00	0.016666666666666666	**Terdeteksi aktivitas dengan nilai respons 0.0167 (1.67% dari skala respons)**
Wed Aug 23 2023 10:31:00	0	Tidak ada aktivitas yang terdeteksi pada interval ini.
Wed Aug 23 2023 10:41:00	0	Tidak ada aktivitas yang terdeteksi pada interval ini.
Wed Aug 23 2023 10:42:00	0.03333333333333333	**Terdeteksi aktivitas dengan nilai respons 0.0333 (3.33% dari skala respons)**
Wed Aug 23 2023 10:43:00	0	Tidak ada aktivitas yang terdeteksi pada interval ini.
Wed Aug 23 2023 10:44:00	0	Tidak ada aktivitas yang terdeteksi pada interval ini.
Wed Aug 23 2023 11:00:00	0	Tidak ada aktivitas yang terdeteksi pada interval ini.
Wed Aug 23 2023 11:01:00	0.016666666666666666	**Terdeteksi aktivitas dengan nilai respons 0.0167 (1.67% dari skala respons)**
Wed Aug 23 2023 11:02:00	0	Tidak ada aktivitas yang terdeteksi pada akhir periode.

Gambar 11. Detail aktivitas *log metric*

Analisis keseluruhan dari data menunjukkan pola aktivitas yang bervariasi selama rentang waktu yang diamati. Terlihat bahwa pada beberapa interval waktu, hasil respons mencapai nilai tertentu yang mengindikasikan adanya aktivitas pelanggaran dalam sistem. Secara keseluruhan, analisis data memberikan wawasan tentang pola, respons data, waktu aktivitas, dan intensitas aktivitas pelanggaran dalam sistem.

V. SIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan dengan judul "Implementasi WAF pada Aplikasi Fishku Berbasis *Google Cloud Armor*", beberapa kesimpulan dapat diambil, bahwa implementasi *Web Application Firewall* (WAF) menggunakan layanan *Google Cloud Armor* berhasil meningkatkan tingkat keamanan aplikasi web Fishku terhadap jenis serangan, termasuk *Local File Inclusion* (LFI), *Vulnerability Scanner*, dan *Protocol Attack*. Penerapan WAF ini menghasilkan perubahan signifikan pada respons server dari sebelumnya yang cenderung positif (respons kode 200) menjadi lebih aman dengan penolakan akses (respons kode 403), menunjukkan efektivitas perlindungan yang lebih baik. Pengujian juga melibatkan konfigurasi *Load Balancer*, yang memungkinkan distribusi lalu lintas aplikasi secara efisien di antara beberapa sumber daya server. Hal ini dapat meningkatkan ketersediaan dan kinerja aplikasi, serta mengoptimalkan tata letak data untuk mengurangi risiko *overloading* pada satu server. Penggunaan fitur *Alerting* juga telah memberikan dampak positif dalam menginformasikan secara cepat jika terjadi serangan atau aktivitas mencurigakan. Notifikasi melalui email memungkinkan respons cepat dari tim keamanan untuk mengidentifikasi dan merespons ancaman segera setelah terdeteksi. Analisis data metrik dan visualisasi grafik menjadi alat yang penting dalam pemantauan kinerja aplikasi dan deteksi dini ancaman. Metrik performa yang diambil dari *Google Cloud Platform* memberikan wawasan yang lebih dalam tentang bagaimana aplikasi beroperasi.

REFERENSI

[1] A. Aprilia and S. Subiyantoro, "Peluang dan Tantangan : Bisnis di era disrupsi industri," J. Eduscience, vol. 9, no. 2, pp. 377–387, 2022, doi: 10.36987/jes.v9i2.2820.

[2] M. Yunus, "Analisis Kerentanan Aplikasi Berbasis Web Menggunakan Kombinasi Security Tools Project Berdasarkan Framework Owasp Versi 4," J. Ilm. Inform. Komput., vol. 24, no. 1, pp. 37–48, 2019, doi: 10.35760/ik.2019.v24i1.1988.

[3] I. Barokah and A. Asriyanik, "Analisis Perbandingan Serverless Computing Pada Google Cloud Platform," J. Teknol. Inform. dan Komput., vol. 7, no. 2, pp. 169– 187, 2021, doi: 10.37012/jtik.v7i2.662.

[4] Ika, "Student-Made E-Commerce Platform Fi$hku Enables Easy Trading of Fishery Products," Universitas Gadjah Mada, 2022. https://www.ugm.ac.id/en/news/22980-student- made-e-commerce-platform-fi-hku-enables-easy- trading-of-fishery-products (accessed Apr. 13, 2023).

[5] R. A. Muzaki, O. C. Briliyant, M. A. Hasditama, and H. Ritchi, "Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall," 2020 Int. Work. Big Data Inf. Secur. IWBIS 2020, pp. 85–90, 2020, doi: 10.1109/IWBIS50925.2020.9255601.

[6] OWASP, "Top 10 Web Application Security Risks," OWASP, 2021. https://owasp.org/www-project-top- ten/ (accessed Aug. 29, 2023).

[7] B. Reddy Bhimireddy, A. Nimmagadda, H. Kurapati, L. Reddy Gogula, R. Rani Chintala, and V. Chandra Jadala, "Web Security and Web Application Security: Attacks and Prevention," 2023 9th Int. Conf. Adv. Comput. Commun. Syst. ICACCS 2023, vol. 1, pp. 2095–2099, 2023, doi: 10.1109/ICACCS57279.2023.10112741.

[8] B. Gerrad and K. Clapa, Professional Cloud Architect- Google Cloud Certification Guide. Packt Publishing Ltd., 2019.

[9] M. Chawda, D. P. Sharma, and M. J. Patel, "Deep Dive into Directory Traversal and File Inclusion Attacks leads to Privilege Escalation," Int. J. Sci. Res. Sci. Eng. Technol., vol. 4099, pp. 115–120, 2021, doi: 10.32628/ijsrset218384.

[10] B. Zukran and M. M. Siraj, "Performance Comparison on SQL Injection and XSS Detection using Open Source Vulnerability Scanners," 2021 Int. Conf. Data Sci. Its Appl. ICoDSA 2021, pp. 61–65, 2021, doi: 10.1109/ICoDSA53588.2021.9617484.

[11] K. Bremberg, "HTTP response splitting exploitations and mitigations," Detectify, 2019, [Online]. Available: https://blog.detectify.com/2019/06/14/http-response- splitting-exploitations-and-mitigations/

[12] M. I. Irawan, U. Y. K. Hediyanto, and , "Implementasi Keamanan Jaringan Pada Cloudfri Dengan Metode Hardening," eProceedings …, vol. 9, no. 2, pp. 644–649, 2022, [Online]. Available: https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/17632

[13] D. P. Iskandar, "Implementasi Web Application Firewall (WAF) Mod Security dan Pengujian Menggunakan SQL Injection l," no. December, 2020.

[14] S. Chavan, P. Jadhav, R. Mahajan, and K. Thopate, "Web Application Security Threats : SQL Injection Attack," vol. 12, no. 7, pp. 401–416.