Edusaintek: Jurnal Pendidikan, Sains dan Teknologi
Volume 11 Issue 2 2024 Pages 431 - 444
p-ISSN: 1858-005X e-ISSN: 2655-3392 DOI: https://doi.org/10.47668/edusaintek.v11i2.1026
website: https://journalstkippgrisitubondo.ac.id/index.php/EDUSAINTEK

PENGEMBANGAN *DASHBOARD MONITORING* CELAH KEAMANAN PADA *CLUSTER* KUBERNETES DENGAN METODE *CONTAINER IMAGES SCANNING*

Muhamad Riko Bediatra[1*], Sri Kusumadewi[2]
[1,2] Universitas Islam Indonesia, Indonesia
[*]e-mail: 19523201@students.uii.ac.id

Abstract: One form of technological development in the information technology industry is Application of Microservices and Containerization. Containerization has become an efficient way to increasing productivity in the process of developing microservicess technology. But along with the increasing uses of container applications, it is also accompanied by an increase of potential security holes such as improper configurations, vulnerabilities in software, malware and others contained in the containers. Therefore, we need a deterrent to reducing the security holes in container images. One form of preventing the security holes is to apply container images scanning. Trivy Operators is one of the container images scanning application that can be run on a cluster. Not only that, but it is also provides security vulnerability reports in the form of metrics. With some help from applications called Prometheus and Grafana, the results of security vulnerability reports can be processed into a dashboard form to make monitoring easier. The final result of this project is a monitoring dashboard using security data from container images scanning.

Keyword: Microservicess, Kubernetes, Container Images Scanning, Trivy Operator, Dashboard

Abstrak: Salah satu bentuk perkembangan teknologi dalam industri teknologi informasi adalah penerapan *micorservice* dan *containerization*. *Containerization* telah menjadi cara yang efisien untuk meningkatkan produktivitas dalam proses pengembangan teknologi informasi berbasis *microservicess.* Namun seiring dengan peningkatan penggunaan *container application,* diiringi juga dengan peningkatan potensi celah keamanan seperti konfigurasi yang kurang tepat, kerentanan pada perangkat lunak, *malware* dan lain lainnya yang terdapat pada *container.* Oleh karena itu diperlukannya sebuah pencegahan untuk mengurangi celah keamanan pada *container images.* Salah satu bentuk pencegahan tersebut adalah dengan menerapkan *container images scanning.* Trivy Operator merupakan salah satu aplikasi *container images scanning* yang dapat berjalan pada suatu *cluster.* Tidak hanya itu Trivy Operator menyediakan laporan celah keamanan dalam bentuk *metrics*. Dengan bantuan aplikasi *Prometheus* dan *Grafana*, laporan celah keamanan yang dihasilkan dapat diolah menjadi sebuah *dashboard* agar mempermudah dalam melakukan kegiatan *monitoring.* Hasil akhir pada proyek ini adalah sebuah *dashboard monitoring* menggunakan data celah keamanan yang berasal dari *container images scanning*.

Kata kunci: *Microservicess*, Kubernetes, *Container Images Scanning*, *Trivy* Operator, *Dashboard*

PENDAHULUAN

Ilmu pengetahuan yang terus berkembang dari masa ke masa mempengaruhi berbagai aspek yang ada di dunia. Salah satunya perkembangan teknologi digital dan

virtual yang terus berkembang sejalan dengan berkembangnya ilmu pengetahuan (Iksyam & Paputungan, 2024). Di era saat ini teknologi sudah menjadi kebutuhan bagi setiap individu mulai dari berkomunikasi, mencari informasi, membantu proses bisnis atau hanya sebagai sarana hiburan. Dikarenakan banyaknya manfaat yang didapatkan dari teknologi, hampir seluruh aspek kehidupan mulai mengadopsi penggunaan teknologi. Banyak perusahaan mulai berlomba-lomba untuk mengembangkan sebuah aplikasi yang dapat membantu kegiatan manusia di segala aspek. Menurut penelitian yang dilakukan oleh Danuri M., kegiatan perusahaan yang tidak berpindah ke era digital ini akan mengalami masalah baik dari sisi efisiensi maupun sisi efektivitas kerja perusahaan (Danuri, 2019).

Salah satu bentuk perkembangan teknologi dalam industri teknologi informasi adalah penerapan *microservicess*. Sistem informasi *enterprise* yang pada umumnya dibangun dengan pendekatan monolitik (aplikasi terbungkus dalam satu *package* besar, di mana perubahan pada salah satu bagian kode program akan besar pengaruhnya terhadap kode program yang lainnya) digeser menjadi pendekatan terdistribusi atau *microservicess* (Suryotrisongko, 2017). *Microservicess* sendiri adalah kumpulan proses independen dan kecil yang berkomunikasi antara satu dengan yang lainnya untuk membentuk aplikasi kompleks yang agnostik terhadap bahasa API apa pun (Putra, 2019). Apabila disederhanakan, *microservices* membagi sebuah aplikasi menjadi beberapa layanan (*service)* yang lebih kecil dan dapat saling terhubung antar satu sama lain menggunakan API. Servis-servis ini terdiri dari block-block kecil yang terpisah dan fokus terhadap tugasnya masing-masing. Hal inilah yang menjadi salah satu perbedaan antara aplikasi *microservices* dan aplikasi yang bersifat monolitik.

Dalam setiap usaha penerapan teknologi tentunya tidak terlepas kan dari kelebihan dan kelemahan pada saat teknologi itu digunakan. Beberapa kelebihan pada penerapan *microservices* dipaparkan oleh Newman pada buku yang berjudul Building Micorservices (Newman, 2021). Kelebihan tersebut diantaranya :

a. Dengan sistem yang dibangun dalam bentuk *services* dapat memungkinkan *developer* menentukan sendiri setiap teknologi yang akan digunakan pada setiap *services* nya. Hal ini yang membuat teknologi yang digunakan pada arsitektur *microservicess* lebih *heterogen* dibandingkan dengan menggunakan arsitektur monolitik.

b. Arsitektur berbasis *microservicess* menjadi lebih tangguh terhadap kerusakan yang terjadi. Tidak hanya itu, apabila terjadi kerusakan dapat mudah dideteksi dan hanya terlokalisasi pada *service* tertentu.
c. *Scalling* dapat dengan lebih mudah dilakukan, karena hanya melakukan *deploy* pada aplikasi tertentu. Sehingga ketika *server* melebihi kapasitas, hanya beberapa *service* yang dilakukan *deploy* ulang, tidak keseluruhan aplikasi.

Sedangkan kekurangan dari penggunaan *microservicess* adalah diperlukannya usaha lebih dalam membuat koneksi antar layanan dan mengatasi kegagalan di sebagian layanan. Tidak hanya itu dalam penerapannya dibutuhkan penggunaan memori yang lebih tinggi sehingga menyebabkan biaya yang dikeluarkan menjadi lebih besar.

Containerization telah menjadi cara yang efisien untuk meningkatkan produktivitas dalam proses pengembangan teknologi informasi berbasis *microservicess* karena memiliki *envirotment* yang lebih ringan dan terisolasi dari *container* yang lainnya (Chen et al., 2022). Namun seiring dengan peningkatan penggunaan *container application,* diiringi juga dengan peningkatan potensi celah keamanan seperti konfigurasi yang kurang tepat, kerentanan pada perangkat lunak, *malware* dan lain lainnya yang terdapat pada *container.* Kerentanan cenderung dapat terjadi ketika paket yang memiliki versi yang lama atau sudah usang secara tidak sengaja dipasang atau ditambahkan pada suatu *container application* (Jaisinghani, 2022). Lebih dari 30% *images* resmi pada Dockerhub memiliki celah keamanan dengan tingkat *high* dan rata-rata hal ini dikarenakan *package* yang sudah tidak diperbaharui (Kaur et al., 2021).

Dalam beberapa tahun terakhir, serangan keamanan menggunakan *container images* sering terjadi (Chen et al., 2022). Cara penyerang melakukan serangan dengan menyusup ke dalam *repository* dan menyuntikan kode *malware* ke dalam *images*. Selanjutnya para pengembang yang mengandalkan atau menggunakan *images* tersebut tidak sadar dan menggunakan *images* tersebut untuk pengembangan yang berulang. Hal ini mengakibatnya semakin cepat suatu aplikasi terinfeksi oleh sebuah *malware*.

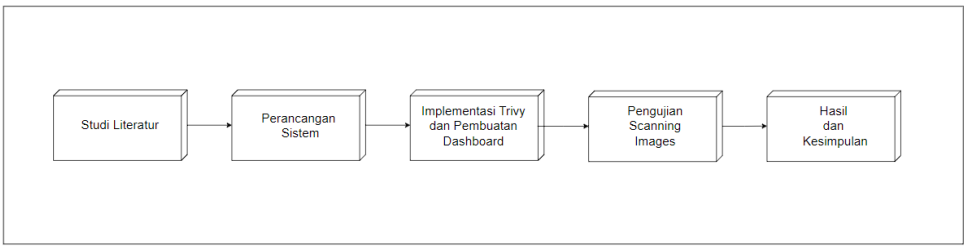
Oleh karena itu diperlukannya sebuah pencegahan untuk mengurangi celah keamanan pada *container images. Images scanning* menjadi sebuah pilihan dalam mendeteksi celah keamanan. Dengan melakukan *images scanning*, pengguna dapat mengetahui celah keamanan apa saja yang terdapat dalam *images* yang akan digunakan. Implementasi *images scanning* dapat dilakukan pada beberapa tahap pengembangan

aplikasi. *Scanning* dapat dilakukan pada saat melakukan *build images*, ketika berada di *images registry* dan pada *runtime cluster*.

Berdasarkan masalah di atas maka dibuatlah sebuah *dashboard* yang berisi laporan celah keamanan suatu *container image.* Dalam proses pembuatan *dashboard* digunakan aplikasi *Trivy* serta *Grafana*. *Trivy* dipilih karena dapat melakukan *scanning image* pada *runtime cluster. Trivy* juga merupakan *container scanner* paling cepat menghasilkan data Vulnerability dibandingkan dengan Anchore (Oktaviana et al., 2022). Tidak hanya itu, *Trivy* juga mampu mendukung prinsip *agile* pada SDLC yang bersifat *continuous* (Perkasa & Mailoa, 2023). *Trivy* dikembangkan oleh salah satu perusahaan yang bergerak pada bidang *cloud native securiy* yaitu Aqua Security. Perusahaan Aqua Security sendiri sudah banyak membantu perusahaan seperti Paypal, Juniper, Adobe, dan lainnya dalam meningkatkan keamanan pada aplikasi yang mereka gunakan. Tidak hanya itu bahkan *Trivy* digunakan oleh aplikasi Gitlab CI sebagai sarana untuk melakukan *container scanning.* Sedangkan alasan pemilihan aplikasi *Grafana* dikarenakan Grafana adalah sebuah *software opensource* yang membaca sebuah data *metrics* untuk dibuat menjadi sebuah grafik atau sebuah data tertulis. Grafana sering digunakan untuk melakukan analisis data dan monitoring. *Grafana* mendukung banyak *storage backends* yang berbeda untuk data *time series* (Source Data) (Mutiara et al., 2020a). *Grafana* menggunakan panel-panel yang dapat dikonfigurasi ataupun menggunakan *templating* dalam menampilkan visualisasi data.

METODE

Dalam sebuah proyek diperlukan tahapan-tahapan yang berguna sebagai acuan dalam melakukan proyek tersebut. Berikut merupakan tahapan yang dilakukan. Metode yang digunakan pada proyek ini adalah *System Development Life Cycle*. Adapun tahapan SDLC secara umum adalah perencanaan, analisis, perancangan, implementasi dan pemeliharaan (Mutiara et al., 2020). Metode ini dapat diubah ataupun dibuat lebih spesifik tergantung dari kebutuhan proyek yang akan dilakukan. Pada penelitian yang dilakukan oleh Panca Rizki Perkasa dan Evangs Mailoa, tahapan SDLC yang dilakukan meliputi indetifikasi masalah, studi literatur, perancangan sistem, implementasi, pengujian, serta hasil dan kesimpulan (Perkasa & Mailoa, 2023). Adapun dalam proyek ini menggunakan tahapan SDLC diantaranya studi literatur, perancangan sistem, implementasi, pengujian, hasil dan kesimpulan.



Gambar 1. Metodologi Penelitian

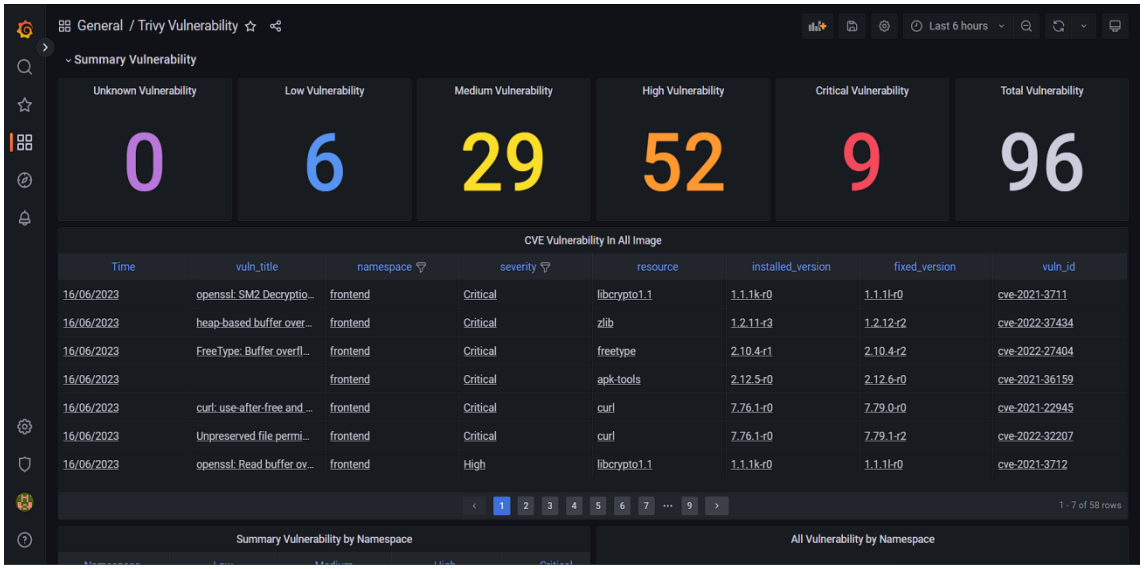
Tahapan awal yang dilakukan untuk mencapai tujuan adalah melakukan studi literatur. Pada tahapan ini akan dilakukan riset mengenai permasalahan yang telah diidentifikasi sebelumnya serta mencari solusi berdasarkan studi literatur yang didukung oleh penelitian-penelitian serupa. Setelah studi literatur dilakukan tahapan selanjutnya adalah melakukan perancangan sistem yang akan dibuat. Tahapan selanjutnya adalah melakukan implementasi *Trivy* pada *running cluster.* Apabila tahapan implementasi telah berhasil, langkah yang dilakukan selanjutnya adalah pembuatan *dashboard*. Hal ini bertujuan agar data hasil *scanning* yang dilakukan dapat dibaca dan dipahami pengguna dengan lebih mudah. Apabila semua telah dilaksanakan dilanjutkan pada tahap pengujian aplikasi *images scanning*. Pengujian ini dilakukan untuk mengetahui bagaimana aplikasi melakukan *scanning images* serta melihat apakah data hasil *scanning* berhasil diolah dan ditampilkan pada sebuah *dashboard.* Tahapan terakhir mengambil kesimpulan dari hasil yang didapatkan selama keseluruhan tahapan dilakukan.

HASIL DAN PEMBAHASAN

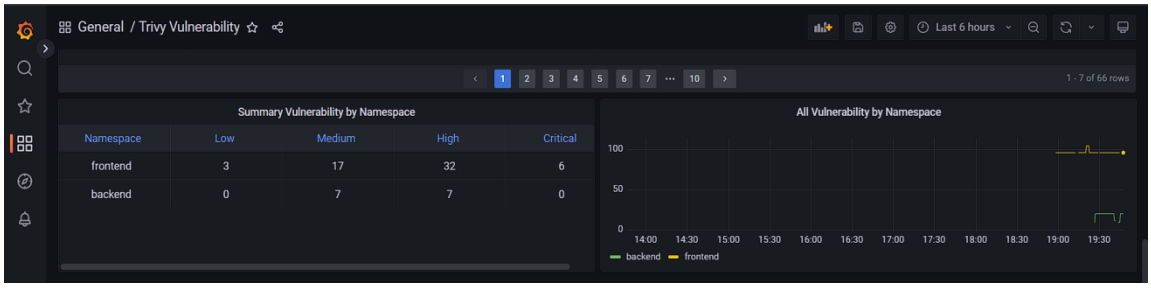
Hasil

1. Tampilan *Dashboard*

Tampilan *dashboard* telah terintegrasi dengan aplikasi *data source* dan secara garis besar tampilan terdiri dari dua bagian yaitu rangkuman celah keamanan dan celah keamanan berdasarkan *namespace*. Pada *dashboard* ini terdapat beberapa jenis panel seperti tabel, grafik garis, serta nilai total angka. Tujuannya agar dapat memudahkan dalam mendapatkan informasi secara detail mengenai celah keamanan pada *container images*.

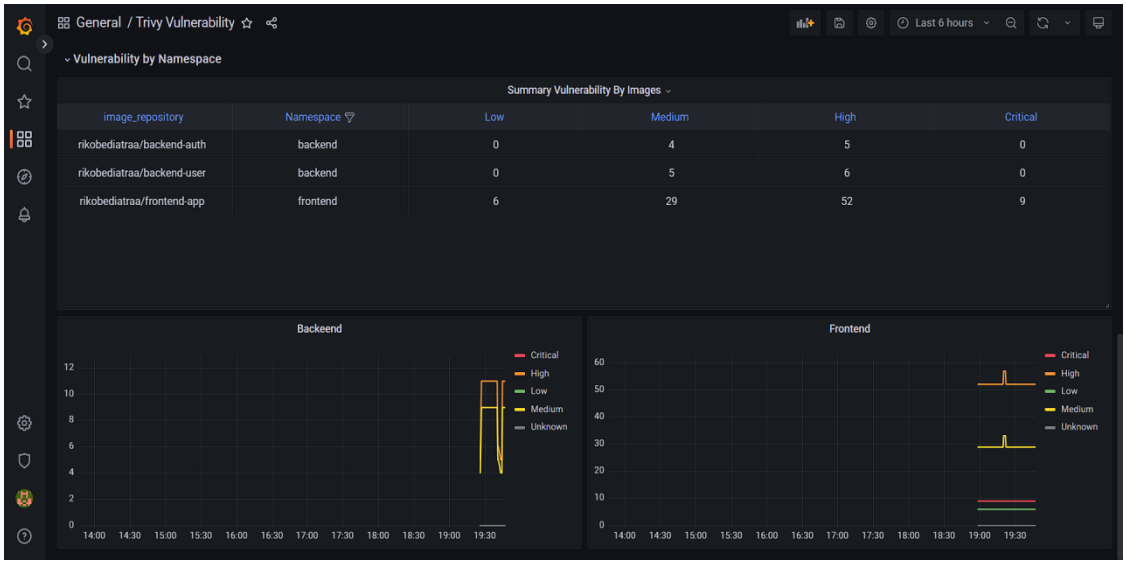


Gambar 2. Tampilan *dashboard* rangkuman celah keamanan



Gambar 3. Lanjutan tampilan rangkuman celah keamanan

Gambar 2 dan Gambar 3 merupakan tampilan rangkuman celah keamanan. Tampilan ini bertujuan untuk mengetahui jumlah keseluruhan celah keamanan berdasarkan tingkat suatu celah keamanan. Tampilan ini juga memberikan informasi mengenai detail celah keamanan, mulai dari nama celah keamanan, *namespace* tempat *container* berada, *resource* yang menjadi celah keamanan, versi yang di-*install* dan versi yang sudah dapat pembaharuan. Tidak hanya itu terdapat juga *vulnerability id* yang dapat terhubung dengan halaman *database vulnerability* yang dimiliki oleh Aqua Security. Selanjutnya terdapat panel yang memberikan informasi mengenai jumlah celah keamanan berdasarkan *namespace* yang ada. Terakhir terdapat sebuah grafik untuk melakukan *monitoring* total celah keamanan berdasarkan *namespace.* Fungsi panel ini adalah agar pengguna dapat mengetahui apabila suatu waktu terdapat penurunan ataupun lonjakan celah keamanan yang terjadi pada suatu *namespace*.

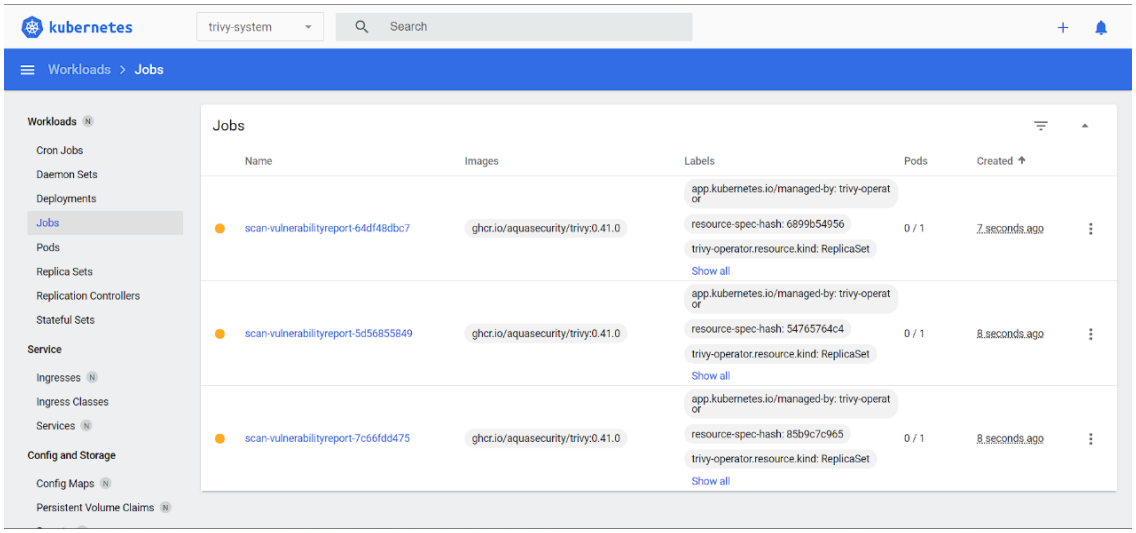


Gambar 4. Tampilan celah keamanan berdasarkan *namespace*

Selanjutnya terdapat bagian mengenai informasi celah keamanan berdasarkan *namespace* seperti pada Gambar 4. Pada bagian ini secara garis besar terdapat dua panel. Panel pertama berisi informasi mengenai jumlah celah keamanan berdasarkan tingkat celah keamanannya. Pada panel ini juga terdapat informasi mengenai *images repository* yang digunakan agar pengguna dapat mengetahui *images* mana yang memiliki tingkat celah keamanan paling tinggi. Selanjutnya terdapat panel berisi grafik yang menunjukan jumlah keamanan pada setiap *namespace.* Grafik ini berisi lima garis yang masing-masing melambangkan jumlah celah keamanan menurut tingkatannya. Fungsi dari panel ini adalah untuk memberikan informasi kepada pengguna apabila di dalam suatu *namespace* terdapat lonjakan ataupun penurunan celah keamanan. Hal yang membedakan dari grafik pada rangkuman celah keamanan adalah panel ini memberikan informasi detail setiap tingkat celah keamanan menurut tingkatannya.

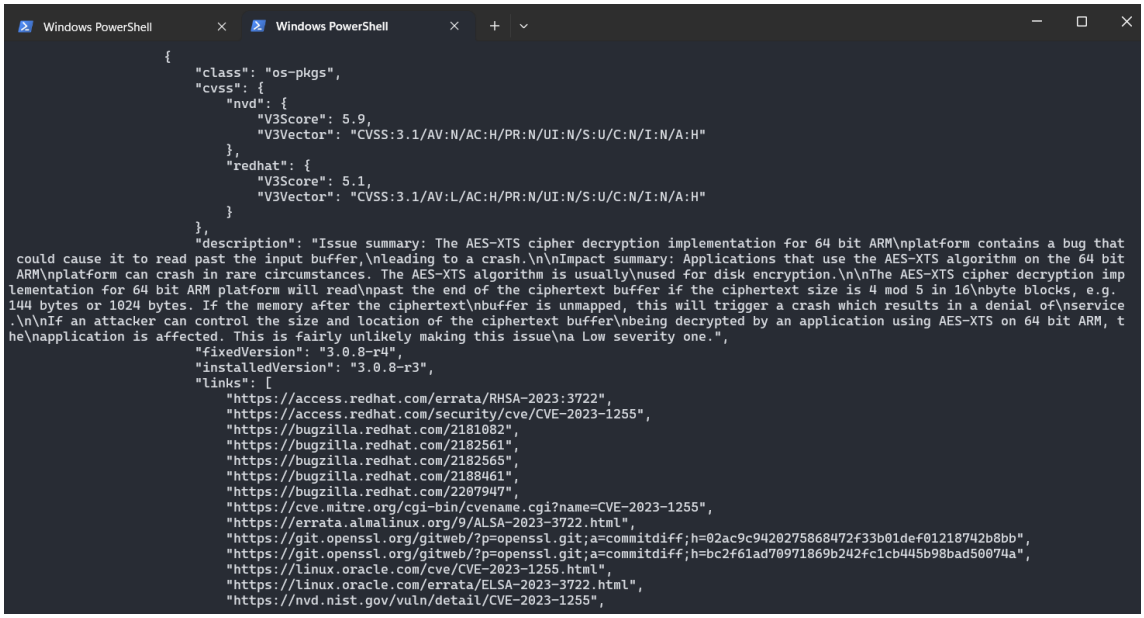
2. Hasil Ketika Pengujian *Scanning Images*

Pengujian *scanning* dilakukan dengan menggunakan *images* yang berasalah dari Dockerhub yang nantinya akan digunakan pada *container deployment.* Pada saat melakukan *deployment, container* akan diberikan *namespace* yaitu *backend* dan *frontend*. Tujuannya nanti adalah untuk menguji apakah *scanning* dapat dilakukan pada target tertentu. Selain itu pengujian juga meliputi uji rekomendasi yang diberikan ketika suatu celah keamanan ditemukan.

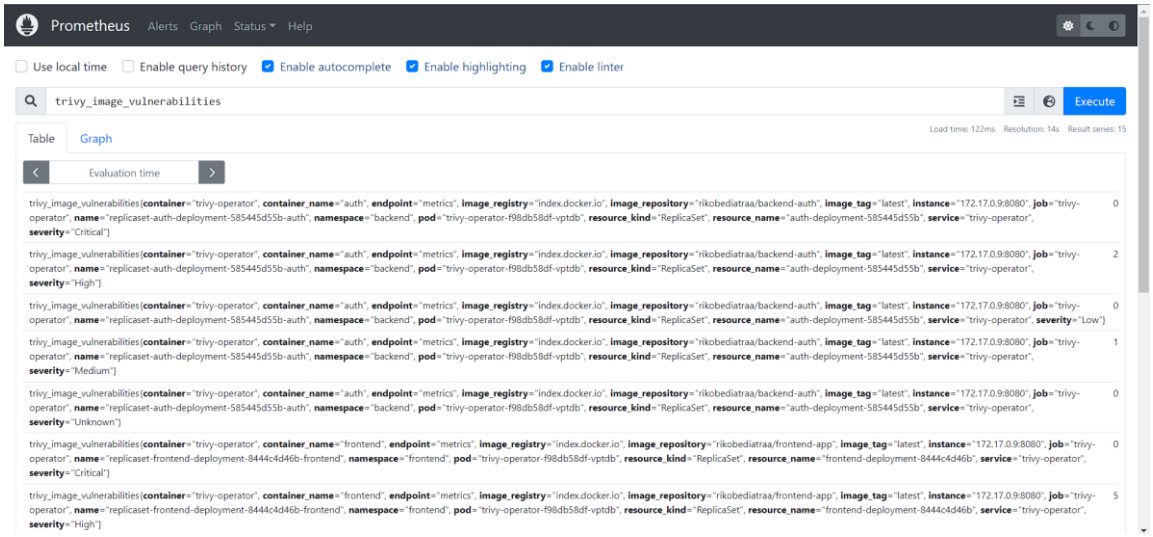


Gambar 5. *Jobs* yang menjalankan fungsi *scanning images*

Setelah aplikasi *Trivy* berhasil diimplementasikan, ketika melakukan *deployment* secara otomatis aplikasi tersebut akan membuat sebuah *pod* yang bertugas sebagai *job. Pod* merupakan representasi dari sekumpulan *container application* dan *volumes* yang berjalan pada satu *envirotment* (lingkungan) yang sama serta berbagi *resource* yang sama (Burns et al., 2022). Gambar 5 menunjukkan *job* yang terbuat saat sebuah *container images* di-*deploy.* Apabila *job* sudah terbuat maka bulatan di samping nama akan berubah menjadi hijau. Selanjutnya apabila telah melakukan tugasnya dalam melakukan *scanning*, secara otomatis *job* tersebut akan menghancurkan dirinya sendiri.

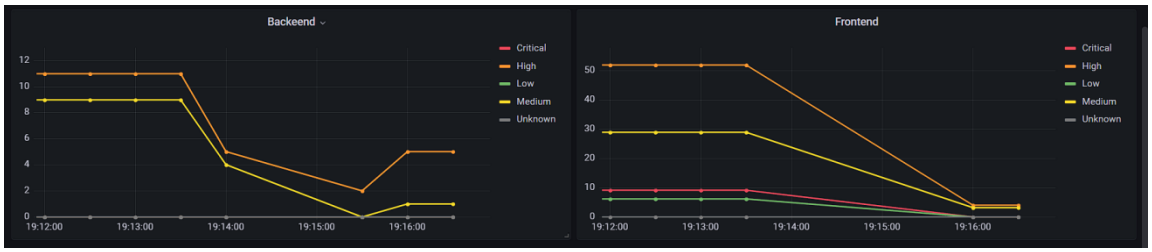


Gambar 6. Rekap dalam bentuk JSON *file*



Gambar 7. *Metrics images vulnerability*

Setelah berhasil melakukan *scanning* terhadap pada *container*, hasil celah keamanan dapat dilihat dalam bentuk format JSON *file* seperti pada Gambar 6. Pada bentuk ini terdapat informasi mengenai *container images* yang memiliki celah keamanan, nilai dari celah keamanan, versi yang telah di-*install*, saran versi yang sudah memperbaiki celah keamanan tersebut. Tidak hanya itu terdapat deskripsi mengenai celah keamanan yang ada. Deskripsi ini memberikan informasi mengapa celah keamanan dapat terjadi dan dampak apa yang akan terjadi apabila celah keamanan tetap dibiarkan. Selanjutnya terdapat laporan dalam bentuk *metrics* seperti pada Gambar 7. Isi laporan ini sama dengan yang berbentuk JSON *file*. Namun data *metrics* dapat diolah menggunakan Prometheus dan *Grafana* agar menjadi sebuah *dashboard*. Selanjutnya data akan diolah dan ditampilkan ke dalam *dashboard.*



Gambar 8. Grafik jumlah *vulnerability*

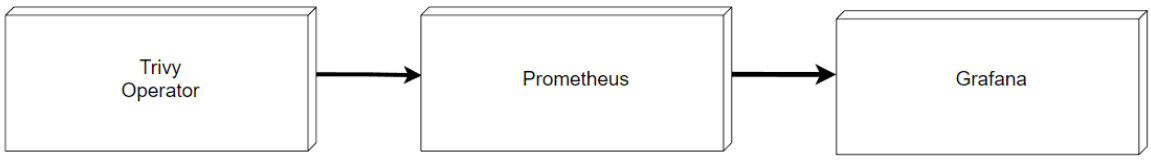
Terakhir adalah pengujian terhadap rekomendasi yang diberikan. Pada informasi mengenai celah keamanan terdapat juga informasi mengenai rekomendasi versi. Rekomendasi versi ini diberikan karena sudah terdapat penanganan mengenai celah keamanan yang ada. Rekomendasi versi ini tidak terbatas hanya pada *based OS*

yang digunakan, namun juga meliputi *package, OS* serta *dependency* yang digunakan. Apabila sudah memperbaharui *container images* dengan versi yang disarankan, maka akan ada penurunan pada jumlah celah keamanan seperti yang ditunjukkan pada Gambar 8.

Pembahasan

Terdapat perbedaan antara *container images scanning* dengan deteksi celah keamanan yang lainnya. Perbedaan ini didapat dari cara suatu informasi celah keamanan didapatkan. Apabila menggunakan *Static Application Security Testing (*SAST) maka berfokus pada *source code* yang digunakan. Menurut penelitian yang dilakukan oleh Fauzan A. pada tahun 2021, SAST murni melakukan analisa pada sumber kode atau *source code* dari aplikasi yang diujikan (Awanda Alviansyah & Ramadhani, 2021). Berbeda dengan *Dynamic Application Security Testing* (DAST). DAST melakukan injeksi kepada aplikasi untuk mengidentifikasi apakah terdapat sebuah celah keamanan ataupun tidak. Sedangkan *container images scanning* berfokus untuk menemukan celah keamanan yang telah diketahui sebelumnya pada sistem aplikasi (Berkovich et al., 2020).

Jurnal ini membahas tentang implementasi *images scanning* pada *cluster* yang sedang berjalan. Setelah implementasi berhasil dilakukan, nantinya akan terdapat data mengenai celah keamanan yang dapat diolah menjadi sebuah *dashboard*. Untuk mewujudkan *dashboard* ini terdapat tiga aplikasi utama yang digunakan yaitu *Trivy* Operator sebagai *container images scanning*, *Prometheus* sebagai penyimpanan basis data dalam bentuk *time series* dan *Grafana* sebagai pembuat *dashboard.*



Gambar 9. Alur hasil *scanning images* hingga menjadi *dashboard*

Secara garis besar alur pembuatan *dashboard* seperti yang terdapat pada Gambar 9. Pertama *Trivy* akan menghasilkan hasil *report* dalam bentuk *metrics*. Selanjutnya *metrics* ini akan ditangkap oleh *Prometheus* dan disimpan dalam *time series database*. Lalu *Grafana* menggunakan dan mengelola *database* yang dimiliki oleh *Prometheus* untuk dijadikan sebuah dashboard.

Langkah pertama yang dilakukan dalam proses pembuatan *dashboard* adalah melakukan instalasi aplikasi *Trivy* Operator. Terdapat beberapa konfigurasi yang perlu dilakukan saat melakukan instalasi agar aplikasi bekerja seperti bagaimana mestinya. Hal pertama yang yang perlu disesuaikan adalah *target namespcae.* Tujuan konfigurasi ini adalah menentukan *namespace* mana yang akan dilakukan *scanning images*. Selanjutnya diperlukan konfigurasi untuk berapa lama sebuah *report* akan bertahan sebelum dihapus. Hal ini dapat diatur dengan merubah konfigurasi pada *report time to live* (reportTTL). Alasan konfigurasi ini agar hasil *report* hanya terdapat pada *storage dashboard* sehingga tidak membebani *storage* yang ada dengan melakukan dua kali penyimpanan. Selanjutnya agar hasil laporan dapat diolah menjadi sebuah *dashboard* diperlukan sebuah *service monitor. Service monitor* berperan sebagai penjebatan antara *report scanning* dan *dashboard* yang akan dibuat. Untuk melakukan hal ini diperlukan menambah *service monitor* dan *label. Label* diberikan nama sesuai dengan *label* yang digunakan oleh *Prometheus*. Apabila *label* pada aplikasi Trivy Operator dan Prometheus berbeda maka tidak akan terbaca dan data tidak dapat diolah.

Selanjutnya adalah proses pengolahan data hingga menjadi sebuah *dashboard. Prometheus* dan *Grafana* digunakan dalam melakukan visualisasi hasil *images scanning*. *Prometheus* sendiri adalah *toolkit* pemantauan dan peringatan sistem yang bersifat *opensource* dan dibuat oleh *SoundCloud*. Sebagai layanan *monitoring*, *Prometheus* memantau hal tertentu atau yang biasa disebut target. Target yang dimaksud dapat berbentuk *server* tunggal atau *endpoint* yang melalui HTTP, HTTPS, DNS, dan lainnya (Mutiara et al., 2020a). Hal yang harus dilakukan selanjutnya adalah memeriksa bahwa *metrics* yang dihasilkan telah dibaca oleh *Prometheus* sebagai sebuah target. Dapat diketahui bahwa *Trivy* menggunakan *port* 8080/metrics sebagai *endpoint* untuk mendapatkan hasil *scanning images.* Terdapat beberapa *metrics* yang dapat digunakan untuk mengelola data. Namun pada jurnal ini hanya menggunakan dua *metrics* yaitu *vulnerability id* dan *image vulnerability.* Perbedaan antara kedua *metrics* ini adalah *images vulnerability* menampilkan data celah keamanan berdasarkan *images* apa saja yang digunakan pada *namespace* sedangkan *vulnerability id* menampilkan data berdasarkan jenis celah keamanannya. Setelah semua disiapkan barulah *dashboard* dapat dibuat pada aplikasi *Grafana*. Dengan menggunakan *Grafana* data yang berbentuk *time*

*series database (*DTSB) dapat diubah menjadi grafik dengan visualisasi yang indah (Rahman et al., 2020)

Terakhir adalah pengujian menggunakan *images* yang berasal dari Dockerhub. Terdapat beberapa hal yang memicu aplikasi *Trivy* secara otomatis melakukan *scanning images.* Pertama adalah pada saat pertama kali aplikasi *Trivy* di-*deploy.* Selanjutnya pada saat sebuah *deployment* aplikasi berhasil dilakukan. Tidak hanya itu, Trivy juga akan melakukan *scanning* apabila *report time to live* sudah kadaluwarsa, aplikasi mengalami pembaharuan ataupun ketika aplikasi di *restart.* Pada saat melakukan rekomendasi perbaikan dengan menaikkan versi pada OS, *package* maupun *dependency* dapat menurunkan jumlah celah keamanan pada suatu *cluster.*

SIMPULAN

Simpulan yang didapat dari jurnal ini adalah bahwa *container images scanning* dapat menjadi salah satu cara untuk meningkatkan keamanan pada aplikasi berbasis *microservices*. Hal ini dikarenakan *container images scanning* dapat dilakukan pada *cluster* Kubernetes yang sedang berjalan. *Trivy* Operator merupakan salah satu aplikasi *container images scanning* yang dapat berjalan pada suatu *cluster.* Pada jurnal ini pun *Trivy* Operator berhasil diimplementasikan dan dapat melakukan *scanning images* sebagaimana mestinya. Dengan bantuan aplikasi *Prometheus* dan *Grafana*, laporan celah keamanan yang tadinya berbentuk JSON dan sedikit sulit untuk dibaca dapat diubah menjadi sebuah *dashboard.* Tujuan dari pembuatan *dashboard* adalah agar *monitoring* celah keamanan pada suatu *cluster* akan lebih mudah dilakukan.

DAFTAR RUJUKAN

Awanda Alviansyah, F., & Ramadhani, E. (2021). *Implementasi Dynamic Application Security Testing pada Aplikasi Berbasis Android*.

Berkovich, S., Kam, J., & Wurster, G. (2020, August). UBCIS: Ultimate Benchmark for Container Image Scanning. *13th USENIX Workshop on Cyber Security Experimentation and Test (CSET 20)*. https://www.usenix.org/conference/cset20/presentation/berkovich

Burns, B., Beda, J., Hightower, K., & Evenson, L. (2022). *Kubernetes: Up and Running*. O'Reilly Media. https://books.google.co.id/books?id=KeB-EAAAQBAJ

Chen, L., Xia, Y., Ma, Z., Zhao, R., Wang, Y., Liu, Y., Sun, W., & Xue, Z. (2022). SEAF: A Scalable, Efficient, and Application-independent Framework for container security detection. *Journal of Information Security and Applications*, *71*, 103351. https://doi.org/https://doi.org/10.1016/j.jisa.2022.103351

Danuri, M. (2019). *PERKEMBANGAN DAN TRANSFORMASI TEKNOLOGI DIGITAL*.

Iksyam, laode G. N., & Paputungan, I. (2024). *PENGEMBANGAN APLIKASI MECHA: LAYANAN PERBAIKANKENDARAAN BERBASIS SMARTPHONE MENGGUNAKAN MVP(MODEL VIEW PRESENTER) SEBAGAI DESIGN PATTERN*. https://doi.org/10.47668/edusaintek.v11i1.932

Jaisinghani, G. (2022). *Vulnerability Management in the Age of Containers*. https://doi.org/10.17605/OSF.IO/WJSV6

Kaur, B., Dugré, M., Hanna, A., & Glatard, T. (2021). An analysis of security vulnerabilities in container images for scientific data analysis. *GigaScience*, *10*(6), giab025. https://doi.org/10.1093/gigascience/giab025

Mutiara, R., Politeknik, F., & Jakarta, N. (2020a). *IMPLEMENTASI SISTEM MONITORING MENGGUNAKAN PROMETHEUS DAN GRAFANA*. https://www.researchgate.net/publication/342511231

Mutiara, R., Politeknik, F., & Jakarta, N. (2020b). *IMPLEMENTASI SISTEM MONITORING MENGGUNAKAN PROMETHEUS DAN GRAFANA*. https://www.researchgate.net/publication/342511231

Newman, S. (2021). *Building Microservicess* (2nd ed.). O'Reilly Media, Inc.

Oktaviana, M., Widjajarto, A., & Almaarif, A. (2022). Analisis Vulnerability Management Pada Container Docker Menggunakan Opensource Scanner Berdasarkan Standar Cyber Resilience Review (CRR). *Jurnal Sistem Komputer Dan Informatika (JSON)*, *4*(1), 77. https://doi.org/10.30865/json.v4i1.4787

Perkasa, P. R., & Mailoa, E. (2023). ADOPSI DEVSECOPS UNTUK MENDUKUNG METODE AGILE MENGGUNAKAN TRIVY SEBAGAI SECURITY SCANNER DOCKER IMAGE DAN DOCKERFILE. *Jurnal Indonesia : Manajemen*

Informatika Dan Komunikasi, *4*(3), 856–863. https://doi.org/10.35870/jimik.v4i3.291

Putra, R. A. (2019). *ANALISA IMPLEMENTASI ARSITEKTUR MICROSERVICESS BERBASIS KONTAINER PADA KOMUNITAS PENGEMBANG PERANGKAT LUNAK SUMBER TERBUKA (OPENDAYLIGHT DEVOPS COMMUNITY)*. Teknologi Informasi dan Komputer. https://jurnal.umj.ac.id

Rahman, D., Amnur, H., & Rahmayuni, I. (2020). *Monitoring Server dengan Prometheus dan Grafana serta Notifikasi Telegram* (Vol. 1, Issue 4). https://doi.org/https://doi.org/10.30630/jitsi.1.4.19

Suryotrisongko, H. (2017). Arsitektur Microservices untuk Resiliensi Sistem Informasi. *Sisfo*, *06*(02), 231–246. https://doi.org/10.24089/j.sisfo.2017.01.006