Jurnal Sistem Komputer dan Informatika (JSON)
Volume 4, Nomor 1, September 2022
e-ISSN 2685-998X
DOI 10.30865/json.v4i1.4789

Vulnerability Management Pada *Vulnerable* Docker Menggunakan Clair *Scanner* Dan Joomscan Berdasarkan Standar GSA CIO-IT *Security*-17-80

Ryan Supriadi Ramadhan, Adityas Widjajarto, Ahmad Almaarif *

Fakultas Rekayasa Industri, Sistem Informasi, Universitas Telkom, Bandung, Indonesia
Email: [1]ryansupriadi@student.telkomuniversity.ac.id, [2]adtwjrt@telkomuniversity.ac.id, [3,*] ahmadalmaarif@telkomuniversity.ac.id
Email Penulis Korespondensi: ahmadalmaarif@telkomuniversity.ac.id
Submitted: **03/09/2022**; Accepted: **26/09/2022**; Published: **30/09/2022**

Abstrak–Kerentanan pada Docker perlu dikelola mengingat bahwa kerentanan ini merupakan salah satu potensi terjadinya eksploitasi, ini bisa terjadi karena Docker merupakan suatu *container* yang berhubungan dengan keamanan aplikasi dan sistem. Penelitian ini menganalisis proses pengelolaan kerentanan pada Docker *Images* dan Aplikasi Docker *Images* dengan menggunakan standar GSA CIO-IT *Security*-17-80. Penelusuran kerentanan ini menggunakan dua tools *Scanning* yaitu Clair *Scanner* dan JoomScan. Kerentanan pada Docker *Images* dan Aplikasi Docker *Images* versi - 1, diatasi dengan membuat sistem baru yaitu versi – 2 yang meningkatkan versi *software* Docker *Images* dan Aplikasi Docker *Images*. Skenario pengujian dijalankan dengan melakukan *scanning* kerentanan pada dua versi sistem percobaan, berupa *vulnerability report*. Data tersebut dianalisis menggunakan Standar GSA CIO-IT *Security*-17-80 yang dibatasi pada tahapan *Scanning Capabilities, Vulnerability Scanning Process, Vulnerability Scan Reports, Remediation Verification,* dan *Re-Classification of Known Vulnerabilities.* Hasilnya berupa waktu *scanning* tercepat ada pada versi - 2, hasil perbandingan *vulnerabilities* yang didapatkan sebanyak 44,45% pada Docker *Images* dan 77,78% pada Joomla. Sehingga kontribusi yang dapat diberikan adalah memberikan gambaran penggunaan standar GSA CIO-IT *Security*-17-80 sebagai panduan pengelolaan keamanan suatu aset IT berdasarkan tahapan yang dilakukan. Kelanjutan penelitian dapat berupa penggunaan ke-6 tahapan GSA dengan dukungan data kerentanan yang memadai dari *software scanner* yang tepat.

Kata Kunci: *Vulnerability*; Docker *Images*; Aplikasi Docker *Images*; GSA CIO-IT *Security*-17-80; *Scanning;* Tahapan

Abstract–Vulnerabilities in Docker need to be managed considering that this vulnerability is one of the potentials for exploitation, this can happen because Docker is a container related to application and system security. This study analyzes the vulnerability management process in Docker Images and Docker Images Applications using the GSA CIO-IT Security-17-80 standard. This vulnerability search uses two scanning tools, namely Clair Scanner and JoomScan. Vulnerabilities in Docker Images and Docker Images application version - 1, were overcome by creating a new system, namely version - 2 which upgrades the Docker Images software and Docker Images application. The test scenario is run by scanning for vulnerabilities in two versions of the trial system, in the form of a vulnerability report. The data was analyzed using the GSA CIO-IT Security Standard-17-80 which was limited to the stages of Scanning Capabilities, Vulnerability Scanning Process, Vulnerability Scan Reports, Remediation Verification, and Re-Classification of Known Vulnerabilities. The result is the fastest scanning time is in version - 2, the results of the comparison of vulnerabilities obtained are 44.45% on Docker Images and 77.78% on Joomla. So that the contribution that can be given is to provide an overview of the use of the GSA CIO-IT Security-17-80 standard as a guide for managing the security of an IT asset based on the stages carried out. Continuation of research can be in the form of using the 6 stages of GSA with the support of adequate vulnerability data from the right scanner software.

Keywords: Vulnerability; Docker Images; Docker Images Application; GSA CIO-IT Security-17-80; Scanning; Stages

1. PENDAHULUAN

Di era industri 4.0 teknologi informasi sudah sangat maju dan berkembang. Hal ini membuat sebuah perusahaan harus mengikuti perkembangannya dengan cara memanfaatkan sebaik mungkin teknologi informasi yang ada untuk meningkatkan layanan bisnis serta meningkatkan performanya. Untuk memaksimalkan manfaat dari teknologi informasi bagi perusahaan dibutuhkan suatu teknologi seperti server dan komputer dengan spesifikasi yang tinggi, akan tetapi hal tersebut akan menjadi masalah bagi perusahaan yang masih berskala kecil karena memerlukan biaya yang tidak sedikit untuk membeli teknologi tersebut. Untuk mengatasi masalah tersebut dapat menggunakan teknologi virtualisasi. Teknologi virtualisasi adalah teknologi untuk membuat komputer fisik menjadi suatu perangkat virtual seperti sistem operasi, media penyimpanan data, dan perangkat keras pada sebuah sistem komputer yang sedang berjalan[1]. Salah satu program kerangka kerja berdasarkan virtualisasi adalah container.

Teknologi terbaru dari *container* yang dikembangkan oleh Solomon Hykes pada tahun 2009 adalah Docker. Docker adalah sebuah platform virtualisasi *container* yang bisa digunakan untuk membangun, menjalankan, dan menyatukan berbagai software atau aplikasi lain yang dibutuhkan agar menjadi sebuah wadah (*container*)[2]. Adanya *container* pada Docker memberikan banyak kemudahan, akan tetapi perlu diperhatikan mengenai keamanan, kerentanan dan risiko dari penggunaannya. Seiring berjalannya waktu, laju pertumbuhan pada teknologi Docker meningkat sangat pesat[3]. Peningkatan pesat ini terjadi antara tahun 2015 sampai 2017. Hal tersebut membuat risiko dari penggunaan Docker juga semakin meningkat dan membuatnya menjadi lebih rentan. Kerentanan itu bisa menjadi celah dan membuatnya menjadi mudah terkena serangan siber seperti *malware*,

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 4, Nomor 1, September 2022
e-ISSN 2685-998X
DOI 10.30865/json.v4i1.4789

ransomware, *hacking*, dan lainnya oleh pihak yang tidak bertanggung jawab demi kepentingan pribadi atau kelompok.

Untuk mendeteksi adanya celah atau kerentanan pada suatu aplikasi kita dapat melakukan *scanning* terhadap aplikasi tersebut. *Scanning* yang digunakan itu disebut *vulnerability scanner*. *Vulnerability scanner* adalah suatu program komputer yang didesain untuk mencari dan memetakan kelemahan sistem pada sebuah aplikasi, komputer atau jaringan[4]. *Vulnerability scanner* yang digunakan pada penelitian ini adalah suatu *open source vulnerability scanner*. *Scanning* ini dilakukan untuk memperoleh informasi dari kerentanan tersebut yang kemudian akan dilakukan proses *Vulnerability Management*. *Vulnerability Management* adalah suatu proses identifikasi, evaluasi, menangani, dan melaporkan kerentanan keamanan pada sistem dan perangkat lunak yang berjalan di atasnya. Dalam pelaksanaanya pemindaian ini akan dilakukan berdasarkan suatu *CyberSecurity Framework*[5]. *CyberSecurity Framework* yang akan digunakan dalam penelitian ini adalah GSA CIO-IT *Security*-17-80. Standar ini dipilih karena mempunyai tahapan yang cukup lengkap dalam melakukan pengujian *Vulnerability Management* pada penelitian ini. Tahapan tersebut adalah *Scanning Capabilities*, *Vulnerability Scanning Process*, *Vulnerability Scan Reports*, *Remediation Verification*, *Re-Classification of Known Vulnerabilities*[6]. Selain memiliki fokus penelitian pada kerangka kerja *Vulnerability Management*, penelitian ini juga menganalisis penilaian proses mitigasi dari aset IT yang memiliki perbedaan versi. Dengan memiliki data kenaikan versi suatu *software container*, maka proses mitigasi kerentanan dapat dianalisis.

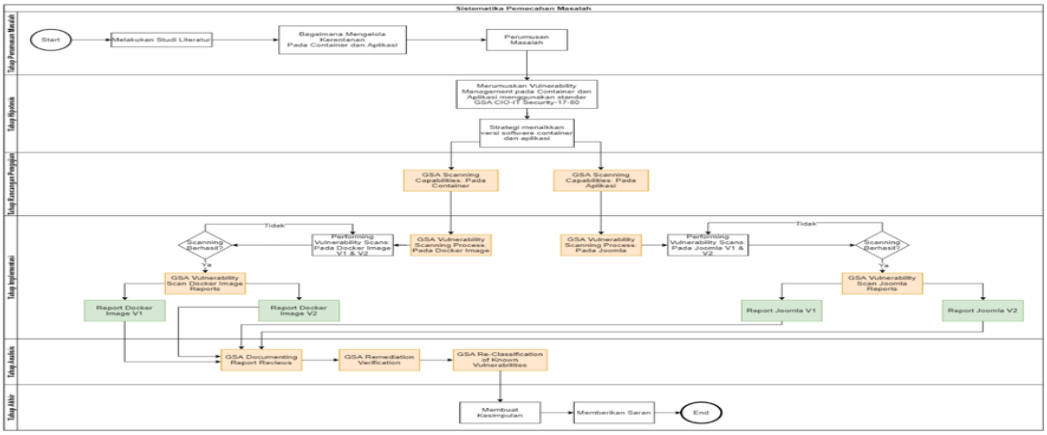
Penelitian ini didasari dari beberapa penelitian sebelumnya yaitu "Analisa Kerentanan Pada *Vulnerable* Docker Menggunakan *Scanner* Openvas Dan Docker *Scan* Dengan Acuan Standar NIST 800-115" oleh Astriani et al, pada tahun 2021. Penelitian ini membahas mengenai analisis resiko dari suatu kerentanan berdasarkan standar NIST 800-115 [7]. "Analisa Kerentanan Pada *Vulnerable* Docker Menggunakan Alienvault Dan Docker Bench *For Security* Dengan Acuan Standar *Framework* Cis Control (CSC)" oleh Hanifah et al, pada tahun 2021. Penelitian ini membahas mengenai analisis resiko dari suatu kerentanan berdasarkan standar CSC[8]. "*Approach to an Efficient Vulnerability Management Program*" oleh Nanda et al, pada tahun 2017. Penelitian ini membahas mengenai upaya pendekatan *Vulnerability Management* yang efisien & mampu menangani resiko *vulnerability* sehingga dihasilkan suatu parameter *Vulnerability Management* yang dapat digunakan[9]. " *Vulnerability Scanning*" oleh Subhangani et al, pada tahun 2022. Penelitian ini membahas mengenai *vulnerability scanning* pada *targeted website* menggunakan nessus dan nexpose[10]. "*GSA Should Establish Goals and Performance Measures to Manage the Smart Buildings Program*" oleh Armes et al, pada tahun 2018. Penelitian ini membahas mengenai penggunaan *framework* GSA untuk bisa mengelola secara lebih efisien konsumsi energi, tindakan operasi dan pemeliharaan pada sistem bangunan[11].

Sedangkan berdasarkan permasalahan yang sudah disebutkan sebelumnya, penelitian ini akan memberikan sebuah hasil analisis *Vulnerability Management* pada Docker *Images* dan Aplikasi Joomla menggunakan *vulnerability Scanner*. *Vulnerability Scanner* yang akan digunakan pada penelitian ini yaitu Clair *Scanner* yang merupakan salah satu *open source Scanner* untuk menemukan *vulnerability* dalam docker *container*[12], dan JoomScan merupakan proyek *open source* yang dikembangkan dengan tujuan mengotomatiskan tugas deteksi *vulnerability* pada CMS Joomla[13], dengan mengacu pada standar GSA CIO-IT *Security*-17-80.

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Sistematika penyelesaian masalah ini merupakan sistematika penelitian untuk mendeskripsikan tahapan penelitian sehingga setiap tahapan yang dilakukan dapat terstruktur dan sistematis. Tahapan ini didasarkan pada metode yang digunakan yaitu metode GSA CIO-IT *Security*-17-80. Berikut sistematika penelitian yang dijelaskan dalam bentuk bagan pada Gambar.



Gambar 1. Sistematika Penyelesaian Masalah

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 4, Nomor 1, September 2022
e-ISSN 2685-998X
DOI 10.30865/json.v4i1.4789

2.1.1 Tahap Perumusan Masalah

Pada tahap awal penelitian ini dimulai dengan mengidentifikasi masalah berdasarkan latar belakang yaitu mengenai bagaimana mengelola kerentanan pada *container* dan aplikasi Docker *Images* menggunakan *Vulnerability Management*. Tahap ini juga menjelaskan metode mana yang akan digunakan untuk menganalisis *Vulnerability Management*. Kemudian dari latar belakang yang ada, dilakukan studi masalah dari berbagai sumber sesuai dengan batasan masalah untuk membantu mengetahui hal-hal yang harus diteliti lebih lanjut dalam penyelesaian masalah.

2.1.2 Tahap Hipotesis

Setelah tahap menganalisis permasalahan dan metode *Vulnerability Management*, pada tahap ini telah ditentukan strategi dan hipotesis untuk mengimplementasikan dan mengelola *Vulnerability Management* pada *container* dan aplikasi Docker *Images* menggunakan standar GSA CIO-IT *Security*-17-80. Selain itu juga dilakukan penaikan versi pada *container* dan aplikasi Docker *Images*.

2.1.3 Tahap Rancangan Pengujian

Pada tahap ini akan dilakukan perancangan sistem berdasarkan standar GSA CIO-IT *Security*-17-80 yaitu *Scanning Capabilities* dengan tujuan untuk mengidentifikasi semua hal yang berkaitan dengan pemindaian meliputi perangkat keras dan perangkat lunak yang akan digunakan. Kemudian membuat skenario pengujian untuk masing masing asset yang akan diuji dan merancang topologinya.

2.1.4 Tahap Implementasi

Pada tahap ini akan dilakukan *Vulnerability Scanning Process* berdasarkan standar GSA CIO-IT *Security*-17-80 untuk melakukan proses *Scanning* pada Docker *Images* dan Joomla. Setelah itu jika *scanning* berhasil maka akan menghasilkan *Vulnerability Scan Reports* yang berisi hasil laporan versi – 1 dan versi - 2 dari proses *scanning* sebelumnya. Hasil laporan tersebut juga akan dibagi menjadi dua bagian yaitu *General Report* dan *Executive Report*.

2.1.5 Tahap Analisis

Pada tahap ini akan dilakukan analisis data hasil pengujian. Analisis dilakukan berdasarkan standar GSA CIO-IT *Security*-17-80 yaitu pada tahapan *Documenting Report Reviews, Remediation Verification, dan Re-Classification of Known Vulnerabilities*.

2.1.6 Tahap Akhir

Tahap akhir adalah pembuatan laporan hasil akhir penelitian yang telah dilakukan. Pada tahap ini berisi kesimpulan dan saran dari setiap proses penelitian yang dilakukan. Selanjutnya akan dicantumkan dokumentasi dari tahap – tahap selama proses penelitian berlangsung.

3. HASIL DAN PEMBAHASAN

3.1 Vulnerability Evaluation Berdasarkan Perbandingan Waktu Versi – 1 Dan Versi – 2

Tahapan analisis ini bertujuan untuk melakukan perbandingan waktu yang dibutuhkan untuk melakukan *Scanning* oleh *Vulnerability Scanner* pada data versi – 1 dan versi – 2 Docker *Images* dan Joomla.

3.1.1 Docker *Images*

Setiap versi pada Docker *Images* yang ada dalam penelitian ini menghasilkan waktu hasil *scanning*. Waktu hasil *scanning* ini disebut *Time for Detection* (TD). *Time for Detection* yang dihasilkan memiliki perbedaan. Tahap analisis ini menggunakan analisis statistik deskriptif untuk mengidentifikasikan perbandingan *Time for Detection* terhadap Docker *Images* versi – 1 dan versi – 2[14]. Berikut rumus persentase yang digunakan.

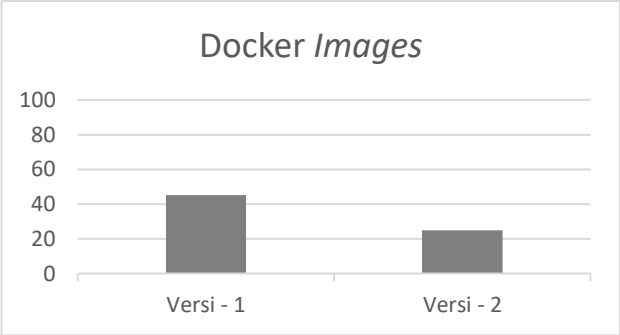
$$\frac{TD\ v2}{TD\ v1} \times 100 \quad (1)$$

Tabel 1. Perbandingan Time for Detection Docker Images

Host	*Time for Detection*	Keterangan
Docker *Images* Versi – 1	12 detik	Docker *Images* Versi – 2 41,6% lebih cepat selesai dalam menghasilkan data kerentanan dari Docker *Images* Versi – 1
Docker *Images* Versi – 2	5 detik	

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 4, Nomor 1, September 2022
e-ISSN 2685-998X
DOI 10.30865/json.v4i1.4789

Dari tabel diatas dapat disimpulkan bahwa hasil *Scan* dari Clair *Scanner* pada Docker *Images* versi – 2, 41,6% lebih cepat dalam melakukan *scanning* kerentanan dari pada Docker *Images* versi – 1. Hasil itu sebanding dengan data yang dihasilkan oleh Docker *Images* versi 1 yang lebih banyak dari Docker *Images* versi – 2.



Gambar 2. Diagram Perbandingan Total Vulnerability Docker Images Versi - 1 dan Versi - 2

3.1.2 Joomla

Setiap versi pada Joomla yang ada dalam penelitian ini menghasilkan waktu hasil *scanning*. Waktu hasil *scanning* ini disebut *Time for Detection* (TD). *Time for Detection* yang dihasilkan memiliki perbedaan. Tahap analisis ini menggunakan analisis statistik deskriptif untuk mengidentifikasikan perbandingan *Time for Detection* terhadap Joomla versi – 1 dan versi – 2[14]. Berikut rumus persentase yang digunakan.

$$\frac{TD\ v2}{TD\ v1} \times 100 \quad (2)$$

Tabel 2. Perbandingan Time for Detection Joomla

Host	*Time for Detection*	Keterangan
Joomla versi - 1	7 detik	Joomla versi – 2 57,14% lebih cepat selesai dalam menghasilkan data kerentanan dari Docker *Images* versi – 1
Joomla versi - 2	4 detik	

Dari tabel diatas dapat disimpulkan bahwa hasil *scan* dari JoomScan pada Joomla versi – 2, 57,14% lebih cepat dalam melakukan *scanning* kerentanan dari pada Joomla versi – 1. Hasil itu sebanding dengan data yang dihasilkan oleh Joomla versi 1 yang lebih banyak dari Joomla versi – 2.

3.2 Vulnerability Evaluation Berdasarkan Perbandingan Data Perubahan Total Vulnerabilities Versi – 1 Dan Versi – 2

Tahapan analisis ini bertujuan untuk melakukan perbandingan pada data perubahan *total vulnerabilities* versi – 1 dan versi – 2 Docker *Images* dan Joomla.

3.2.1 Docker Images

Setiap versi pada Docker *Images* yang ada dalam penelitian ini menghasilkan suatu data *Vulnerability*. Data *Vulnerability* yang dihasilkan memiliki perbedaan total hasil *Vulnerability*, karena repositori versi yang berbeda dengan versi sebelumnya. Tahap analisis ini menggunakan analisis statistik deskriptif untuk mengidentifikasikan perubahan hasil *Vulnerability* yang dilakukan pada setiap versi yang dihasilkan dari data versi lama dan versi baru[14]. Berikut merupakan rumus persentase perubahan data berdasarkan total *Vulnerability* yang dihasilkan oleh masing-masing versi *Vulnerable* Docker *Images*.

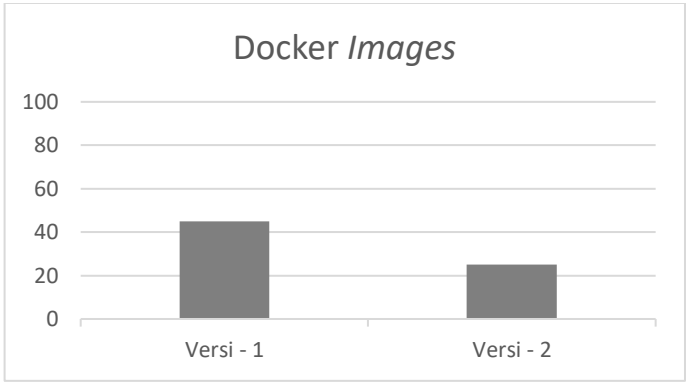
$$\frac{Total\ Vulnerability\ v2 - Total\ Vulnerability\ v1}{Total\ Vulnerability\ v1} \times 100 \quad (3)$$

Tabel 3. Perbandingan Data Perubahan Total Vulnerabilities Docker Images

Host	*Total Vulnerability*	Keterangan
Docker *Images* Versi – 1	45	Docker *Images* Versi – 1 menghasilkan 44,45% lebih banyak data kerentanan dari Docker *Images* Versi – 2
Docker *Images* Versi – 2	25	

Dari tabel diatas dapat disimpulkan bahwa Docker *Images* versi – 1 lebih banyak menghasilkan kerentanan dari pada Docker *Images* versi – 2. Kerentanan yang dihasilkan Docker *Images* versi – 1 44,45% lebih banyak dari kerentanan yang dihasilkan Docker *Images* versi – 2.

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 4, Nomor 1, September 2022
e-ISSN 2685-998X
DOI 10.30865/json.v4i1.4789



Gambar 3. Diagram Perbandingan Total Vulnerability Docker Images Versi - 1 dan Versi - 2

Dari diagram tersebut dapat diambil kesimpulan bahwa strategi menaikkan versi atau proses mitigasi *Vulnerability* Docker *Images* dapat dikatakan berhasil karena total *Vulnerability* versi – 2 mengalami jumlah penurunan dari total *Vulnerability* versi – 1.

3.2.2 Joomla

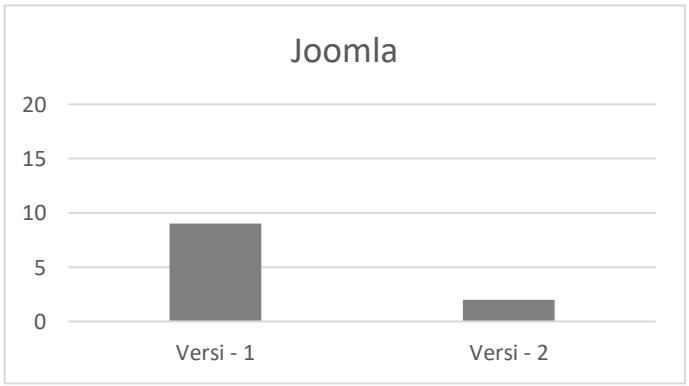
Setiap versi pada Joomla yang ada dalam penelitian ini menghasilkan suatu data *Vulnerability*. Data *Vulnerability* yang dihasilkan memiliki perbedaan total hasil *Vulnerability*, karena repositori versi yang berbeda dengan versi sebelumnya. Tahap analisis ini menggunakan analisis statistik deskriptif untuk mengidentifikasikan perubahan hasil *Vulnerability* yang dilakukan pada setiap versi yang dihasilkan dari data versi lama dan versi baru[14]. Berikut merupakan rumus persentase perubahan data berdasarkan total *Vulnerability* yang dihasilkan oleh masing-masing versi Joomla.

$$\frac{Total\ Vulnerability\ v2 - Total\ Vulnerability\ v1}{Total\ Vulnerability\ v1} \times 100 \quad (4)$$

Tabel 4. Perbandingan Data Perubahan Total Vulnerabilities Joomla

Host	***Total Vulnerability***	**Keterangan**
Joomla Versi – 1	9	Joomla Versi – 1 menghasilkan 77,78% lebih banyak data kerentanan dari Joomla Versi – 2
Joomla Versi – 2	2	

Dari tabel diatas dapat disimpulkan bahwa Joomla versi – 1 lebih banyak menghasilkan kerentanan dari pada Joomla versi – 2. Kerentanan yang dihasilkan Joomla versi – 1 77,78% lebih banyak dari kerentanan yang dihasilkan Joomla versi – 2.



Gambar 4. Diagram Perbandingan Total Vulnerabilities Joomla Versi - 1 dan Versi – 2

Dari diagram tersebut dapat diambil kesimpulan bahwa strategi menaikkan versi atau proses mitigasi *Vulnerability* Joomla dapat dikatakan berhasil karena total *Vulnerability* versi – 2 mengalami jumlah penurunan dari total *Vulnerability* versi – 1.

3.3 Vulnerability Evaluation Berdasarkan Perbandingan Severity Level Vulnerablity Scanner dan Hasil Perhitungan CVSS

Tahapan analisis ini bertujuan untuk melakukan perbandingan dan memastikan data *Severity* level yang dihasilkan oleh *Vulnerability Scanner* Clair *Scanner* dan JoomScan sama dengan data hasil perhitungan CVSS[15].

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 4, Nomor 1, September 2022
e-ISSN 2685-998X
DOI 10.30865/json.v4i1.4789

3.3.1 Docker Images Versi - 1

Tabel 5. Perbandingan Severity Level Docker Images Versi - 1

Package Name	CVE	*Severity* Clair *Scanner*	*Severity Newest*
openssl	CVE-2016-6303	*High*	*Critical*
curl	CVE-2016-8619	*High*	*Critical*
curl	CVE-2016-7167	*High*	*Critical*
curl	CVE-2016-8622	*High*	*Critical*
openssl	CVE-2016-2182	*High*	*Critical*
curl	CVE-2016-9953	*High*	*Critical*
curl	CVE-2016-8618	*High*	*Critical*
musl	CVE-2016-8859	*High*	*Critical*
busybox	CVE-2016-6301	*High*	*High*
openssl	CVE-2016-6304	*High*	*High*

Dari tabel diatas dapat ditemukan bahwa terdapat beberapa perubahan dari *Severity level* pada data yang didapatkan oleh *Vulnerability Scanner* dan yang didapatkan dari hasil perhitungan CVSS. Beberapa data tersebut memiliki perubahan *Severity level* sebanyak satu tingkat yaitu dari *High* ke *Critical*. Sehingga terdapat 8 *Severity Critical* dan 2 *Severity High* pada Docker *Images* versi – 1.

3.3.2 Docker Images Versi – 2

Tabel 6. Perbandingan Severity Level Docker Images Versi - 2

Package Name	CVE	*Severity* Clair *Scanner*	*Severity Newest*
busybox	CVE-2022-28391	*High*	*Critical*
openssl	CVE-2021-3711	*High*	*Critical*
apk-*tools*	CVE-2021-36159	*Medium*	*Critical*
openssl	CVE-2021-3450	*Medium*	*High*
busybox	CVE-2021-42378	*Medium*	*High*
busybox	CVE-2021-42381	*Medium*	*High*
busybox	CVE-2021-42380	*Medium*	*High*
busybox	CVE-2021-42382	*Medium*	*High*
zlib	CVE-2018-25032	*Medium*	*High*
busybox	CVE-2021-28831	*Medium*	*High*

Dari tabel diatas dapat ditemukan bahwa terdapat perubahan dari *Severity level* pada semua data yang didapatkan oleh *Vulnerability Scanner* dan yang didapatkan dari hasil perhitungan CVSS. Data tersebut memiliki perubahan *Severity level* sebanyak satu dan dua tingkat yaitu dari *High* ke *Critical*, *medium* ke *Critical*, dan *medium* ke *High*. Sehingga terdapat 3 *Severity Critical* dan 7 *Severity High* pada Docker *Images* versi – 2.\

3.3.3 Joomla Versi – 1

Tabel 7. Perbandingan Severity Level Docker Joomla Versi - 1

Package Name	CVE	*Severity* JoomScan	*Severity Newest* CVSS V3	*Severity Newest* CVSS V2
PHPMailer	CVE-2016-10033	*Critical*	*Critical*	
PHPMailer	CVE-2016-10045	*Critical*	*Critical*	
Joomla!	CVE-2016-9838	*High*	*High*	
Joomla!	CVE-2015-8565	N/A		*High*
Joomla!	CVE-2015-8564	N/A		*High*
Joomla!	CVE-2016-9837	*High*	*High*	
Joomla!	CVE-2015-6939	N/A		*Medium*
Joomla!	CVE-2015-7859	N/A		*Medium*
Joomla!	CVE-2015-8563	N/A		*Medium*

Dari tabel diatas dapat ditemukan bahwa terdapat beberapa perubahan dari *Severity level* pada data yang didapatkan oleh *Vulnerability Scanner* dan yang didapatkan dari hasil perhitungan CVSS. Beberapa data tersebut memiliki perubahan *Severity level* pada CVSS versinya. *Vulnerability Scanner* JoomScan tidak dapat mengidentifikasi beberapa data *Severity level* dari Joomla versi - 1 dikarenakan setelah dilakukan perhitungan CVSS data tersebut tidak terdapat pada CVSS versi 3 melainkan terdapat pada CVSS versi 2. Sehingga terdapat 2 *Severity Critical*, 4 *Severity High*, dan 3 *Severity Medium* pada Joomla versi – 1.

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 4, Nomor 1, September 2022
e-ISSN 2685-998X
DOI 10.30865/json.v4i1.4789

3.3.4 Joomla Versi – 2

Tabel 8. Perbandingan Severity Level Docker Joomla Versi - 2

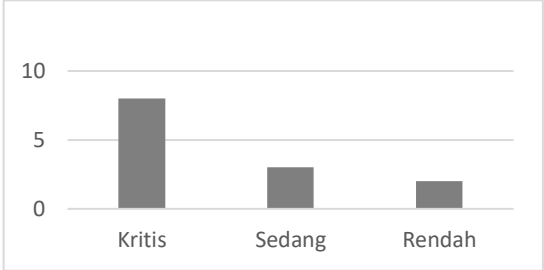
Package Name	CVE	*Severity* JoomScan	*Severity Newest*
PHPMailer	CVE-2016-10033	*Critical*	*Critical*
PHPMailer	CVE-2016-10045	*Critical*	*Critical*

Dari tabel diatas dapat ditemukan bahwa tidak ada perubahan dari *Severity level* pada semua data yang didapatkan oleh *Vulnerability Scanner* dan yang didapatkan dari hasil perhitungan CVSS. Data tersebut sudah selaras, sehingga terdapat 2 *Severity Critical* pada Joomla versi – 2.

3.4 Vulnerability Evaluation Berdasarkan Remediation Verification

Tahapan analisis ini bertujuan untuk mengetahui tanggal berapa suatu kerentanan itu di *publish* dan tanggal berapa *patch* dirilis untuk menanggulangi kerentanan tersebut. Kemudian berapa lama waktu yang di butuhkan dari munculnya kerentanan hingga di rilisnya *patch* untuk kerentanan tersebut. Sehingga dapat dikategorikan waktu remediasinya termasuk ke kategori mana berdasarkan acuan dari standar GSA CIO-IT *Security*-17-80.

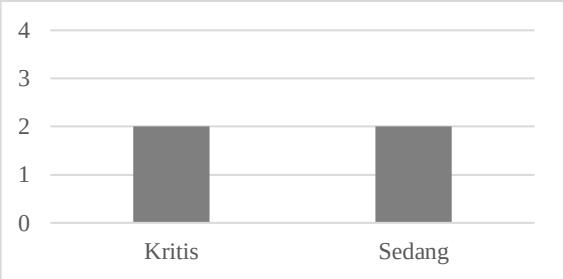
3.4.1 Docker Images



Gambar 5. Diagram Perbandingan Kategori Docker Images

Berdasarkan data dari diagram diatas didapatkan hasil dari 13 data kerentanan, terdapat 8 kerentanan pada kategori kritis, 3 kerentanan pada kategori sedang, dan 2 kerentanan pada kategori rendah.

3.4.2 Joomla



Gambar 6 Diagram Perbandingan Kategori Joomla

Berdasarkan data dari diagram diatas didapatkan hasil dari 4 data kerentanan, terdapat 2 kerentanan pada kategori kritis dan 2 kerentanan pada kategori sedang.

3.5 Re-Classification of Known Vulnerabilities : GSA CIO-IT Security-17-80

Tahapan analisis ini bertujuan untuk melakukan validasi terhadap laporan yang telah didapatkan sebelumnya. Validasi ini dilakukan dengan cara menjelaskan secara detail kerentanan mana saja yang memiliki status kritis dan bagaimana cara perbaikannya. Berikut merupakan analisis kerentanan dengan status kritis berdasarkan perhitungan *Remediation Verification*.

1. openssl (CVE-2016-6303)
   Pada kerentanan ini telah ditemukan *Integer overfLow* sebelum versi 1.1.0 memungkinkan penyerang jarak jauh menyebabkan penolakan layanan atau mungkin memiliki dampak lain yang tidak ditentukan melalui vektor yang tidak dikenal. Solusi yang dilakukan adalah melakukan *upgrade* versi openssl.
2. openssl (CVE-2016-2182)
   Pada kerentanan ini telah ditemukan suatu fungsi di Openssl sebelum versi 1.1.0 tidak memvalidasi hasil pembagian dengan benar, yang memungkinkan penyerang jarak jauh menyebabkan penolakan layanan. Solusi yang digunakan pada kerentanan ini adalah dengan melakukan pemeriksaan pada fungsi tersebut.
3. curl (CVE-2016-9953)

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 4, Nomor 1, September 2022
e-ISSN 2685-998X
DOI 10.30865/json.v4i1.4789

Pada kerentanan ini telah ditemukan pada libcurl versi 7.30.0 hingga 7.51.0, yang memungkinkan penyerang jarak jauh untuk mendapatkan informasi sensitif, menyebabkan penolakan layanan (*crash*). Solusi yang digunakan pada kerentanan ini adalah melakukan pemeriksaan sertifikat yang diubah untuk menggunakan fungsi verifikasi sertifikat pada libcurl.

4. curl (CVE-2016-8618)
   Pada kerentanan ini telah ditemukan fungsi API libcurl yang disebut sebelum versi 7.51.0 yang dapat diakali untuk melakukan *double-free* pada sistem. Solusi yang digunakan pada kerentanan ini adalah melakukan *update* curl ke versi 7.51.0
5. busybox (CVE-2022-28391)
   Pada kerentanan ini telah ditemukan bahwa BusyBox versi 1.35.0 memungkinkan penyerang jarak jauh untuk mengeksekusi kode *arbitrary*. Solusi yang digunakan pada kerentanan ini adalah dengan memastikan *host* hanya berisi karakter yang dapat dicetak.
6. zlib (CVE-2018-25032)
   Pada kerentanan ini telah ditemukan bahwa zlib sebelum versi 1.2.12 memungkinkan kerusakan memori saat mengempis yaitu, saat melakukan kompresi pada *input* yang memiliki banyak kecocokan. Solusi yang digunakan pada kerentanan ini adalah dengan menggabungkan buffer jarak dan panjang menjadi buffer simbol tunggal.
7. busybox (CVE-2021-28831)
   Pada kerentanan ini telah ditemukan BusyBox versi 1.32.1 salah menangani *error* bit sehingga menyebabkan salah format. Solusi yang digunakan pada kerentanan ini adalah dengan melakukan pembaruan pada busybox yang dapat diinstal dengan program pembaruan "dnf".
8. apk-*tools* (CVE-2021-36159)
   Pada kerentanan ini telah ditemukan bahwa pada libfetch, seperti yang digunakan dalam apk-*tools* salah menangani *string numerik* untuk protokol FTP dan HTTP. Implementasi mode *pasif* FTP memungkinkan pembacaan di luar sistem. Solusi yang digunakan pada kerentanan ini adalah dengan menghapus variable tersebut pada libfetch.
9. PHPMailer (CVE-2016-10033)
   Pada kerentanan ini telah ditemukan bahwa pada fungsi mailSend dalam *transport* isMail di PHPMailer sebelum versi 5.2.18 memungkinkan penyerang jarak jauh untuk mengirimkan parameter tambahan ke perintah *mail*. Solusi yang digunakan pada kerentanan ini adalah dengan melakukan dengan memperbaiki *address* PHPMailer.
10. PHPMailer (CVE-2016-10045)
    Pada kerentanan ini telah ditemukan bahwa pada *transport* isMail di PHPMailer sebelum versi 5.2.20 memungkinkan penyerang jarak jauh untuk meneruskan parameter tambahan ke perintah *email* dan membuatnya bisa mengeksekusi kode *arbitrer*. Solusi yang digunakan pada kerentanan ini adalah dengan menggunakan *transport* SMTP PHPMailer.

4. KESIMPULAN

Berdasarkan hasil analisis ada beberapa kesimpulan yang dapat diambil yaitu pada Docker *Images* dan Joomla setiap versi, versi – 1 lebih banyak memakan waktu dalam melakukan *Scanning* kerentanan dari pada versi – 2. Versi – 1 lebih banyak menghasilkan kerentanan dari pada versi – 2. Hal itu berarti bahwa total *Vulnerability* pada versi – 2 mengalami penurunan. Sehingga proses mitigasi yang dilakukan dapat dikatakan berhasil dengan menggunakan strategi menaikkan versi Docker *Images* dan Joomla. Terdapat perbedaan data severity level dari hasil scanning Clair *Scanner* dan JoomScan dengan perhitungan CVSS. Berdasarkan hal tersebut dapat dikatakan bahwa Clair *Scanner* belum melakukan *update database* kerentanan. Sedangkan Untuk JoomScan tidak dapat mengidentifikasi kerentanan CVSS versi 2. Berdasarkan hasil analisis pada *Documenting Report* Review dalam penelitian ini menghasilkan kategori *data* yaitu *Closed Vulnerability*, *Open Vulnerability*, dan *Newly Vulnerability*. Setelah didapatkan hasil analisis dari *Remediation Verification*. Pada Docker *Images* terdapat 8 kerentanan memiliki kategori kritis, 3 kerentanan memiliki kategori sedang, dan 2 kerentanan memiliki kategori rendah. Sedangkan pada Joomla terdapat 2 kerentanan memiliki kategori Kritis dan 2 kerentanan memiliki kategori Sedang.

REFERENCES

[1] R. Umar, “REVIEW TENTANG VIRTUALISASI,” 2013.
[2] M. Fadlulloh dan R. Bik, “IMPLEMENTASI DOCKER UNTUK PENGELOLAAN BANYAK APLIKASI WEB (Studi Kasus : Jurusan Teknik Informatika UNESA),” 2017.
[3] A. EFE, U. ASLAN, dan A. M. KARA, “Securing Vulnerabilities in Docker Images,” International Journal of Innovative Engineering Applications, vol. 4, no. 1, hlm. 31–39, Jun 2020, doi: 10.46460/ijiea.617181.
[4] D. C. Angir, A. Noertjahyana, dan J. Andjarwirawan, “Vulnerability Mapping pada Jaringan Komputer di Universitas X,” Jurnal Infra, vol. 3, no. 2, 2015.

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 4, Nomor 1, September 2022
e-ISSN 2685-998X
DOI 10.30865/json.v4i1.4789

[5] J. Srinivas, A. K. Das, dan N. Kumar, "Government regulations in cyber security: Framework, standards and recommendations," Future Generation Computer Systems, vol. 92, 2019, doi: 10.1016/j.future.2018.09.063.
[6] General Services Administration, "IT Security Procedural Guide: Vulnerability Management Process CIO-IT Security-17-80," 2021.
[7] T. Astriani, A. Budiyono, dan A. Widjajarto, "Analisa Kerentanan Pada Vulnerable Docker Menggunakan Scanner Openvas Dan Docker Scan Dengan Acuan Standar NIST 800-115," vol. 8, no. 4, 2021, [Daring]. Available: http://jurnal.mdp.ac.id
[8] F. Hanifah, A. Budiyono, dan A. Widjajarto, "ANALISA KERENTANAN PADA VULNERABLE DOCKER MENGGUNAKAN ALIENVAULT DAN DOCKER BENCH FOR SECURITY DENGAN ACUAN STANDAR FRAMEWORK CIS CONTROL (CSC)," 2021.
[9] S. Nanda, U. Ghugar, S. Associate, dan P. Scholar, "Approach to an Efficient Vulnerability Management Program International Journal of Innovative Research in Computer and Communication Engineering Approach to an Efficient Vulnerability Management Program," Article in International Journal of Innovative Research in Computer and Communication Engineering, 2017, doi: 10.15680/IJIRCCE.2017.
[10] Subhangani dan A. Chaudhary, "Vulnerability Scanning," 2022.
[11] M. Armes, D. Paepke, dan E. Alexander, "GSA Should Establish Goals and Performance Measures to Manage the Smart Buildings Program," 2018.
[12] B. Kaur, M. Dugré, A. Hanna, dan T. Glatard, "An analysis of security vulnerabilities in container images for scientific data analysis," Gigascience, vol. 10, no. 6, 2021, doi: 10.1093/gigascience/giab025.
[13] Y. N. Kunang, M. Fatoni, dan S. Sauda, "PENGUJIAN CELAH KEAMANAN PADA CMS (CONTENT MANAGEMENT SYSTEM)," Prosiding SeNAIK, hlm. 398–406, 2013.
[14] M. Muchson, Statistik Deskriptif. Bogor: Guepedia, 2017.
[15] FIRST, "Common Vulnerability Scoring System SIG," 2019. https://www.first.org/cvss/