Jurnal Indonesia : Manajemen Informatika dan Komunikasi
www.journal.amikindonesia.ac.id/jimik/ Vol 4 No 3, September (2023)
E-ISSN: 2723-7079, P-ISSN: 2776-8074

ADOPSI *DEVSECOPS* UNTUK MENDUKUNG METODE *AGILE* MENGGUNAKAN *TRIVY* SEBAGAI *SECURITY SCANNER DOCKER IMAGE* DAN *DOCKERFILE*

Panca Rizki Perkasa [1*], Evangs Mailoa [2].

[1*,2] Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Kota Salatiga, Provinsi Jawa Tengah, Indonesia.

Email: rizkypanca386@gmail.com [*1], evangs.mailoa@uksw.edu [2]

Histori Artikel:
Dikirim 25 Mei 2023; *Diterima dalam bentuk revisi* 10 Juni 2023; *Diterima* 1 Juli 2023; *Diterbitkan* 10 September 2023.

Abstrak

Kepatuhan terhadap undang-undang perlindungan data pribadi mewajibkan penyelenggara sistem elektronik lebih memperhatikan security dalam aplikasi. Security testing yang biasa dilakukan diakhir SDLC membuat prinsip Agile tidak sesuai dengan tujuannya yang mengutamakan percepatan, adaptif, dan responsif terhadap perubahan. Implementasi DevSecOps menggunakan Trivy akan menyisipkan proses security scanner terhadap aplikasi yang di deploy dalam bentuk kontainerisasi. Proses security scanner yang di intergrasikan dalam CI/CD yang sifatnya continuous akan meningkatkan kesadaran dari developer dalam hal keamanan aplikasi, sehingga developer akan lebih cepat untuk memperbaiki masalah tersebut dan menghindari penumpukan security issues diakhir SDLC.

Kata Kunci: Agile; CI/CD; Security; DevSecOps; Kontainer.

Abstract

Compliance with personal data protection laws requires electronic system operators to pay more attention to security in applications. Security testing which is usually done at the end of the SDLC makes Agile principles incompatible with advantages that prioritize acceleration, adaptability and responsiveness to change. DevSecOps implementation using Trivy will insert a security scanner process for applications that are deployed in containerized form. The continuous process of security scanning integrated in CI/CD will increase the awareness of developers in terms of application security, so that developers will more quickly fix these problems and avoid security problems at the end of the SDLC.

Keyword: Agile; CI/CD; Security; DevSecOps; Container.

1. Pendahuluan

Dalam perancangan sebuah aplikasi maupun sistem, dibutuhkannya kerangka yang disebut dengan SDLC (*Software Development Life Cycle*), untuk menggambarkan proses pembuatan suatu aplikasi dari awal hingga aplikasi dapat digunakan oleh pengguna [1]. *Agile* merupakan metode yang digunakan untuk mendukung penggunaan model SDLC, dan mempunyai sifat adaptif serta responsif terhadap perubahan [2]. Ketidaksiapan SDM serta ketidakcocokan nilai *Agile* dengan budaya perusahaan, menjadi tantangan dan penyebab kegagalan penerapan metodologi ini di perusahaan [3]. Terhambatnya implementasi dari *Agile*, sering diakibatkan oleh ketidaksesuaian keinginan antara pihak *developer* dan *operation*, walaupun kedua belah pihak tersebut memiliki tujuan yang sama agar aplikasi bisa cepat untuk dirilis [4]. Budaya DevOps hadir sebagai solusi untuk mempersatukan *developer* dan *operation* menjadi DevOps, dengan tujuan menciptakan kolaborasi, interaksi, dan empati yang sangat diperlukan dalam proses SDLC [5]. Pengaplikasian budaya DevOps bisa dilakukan dengan otomatisasi menggunakan tools yang biasa di kenal dengan proses CI/CD, dan bertujuan menghindari human error serta mengimplementasikan nilai dari Agile, untuk pengembangan aplikasi yang berkesinambungan dan cepat untuk bisa di deliver kepada pengguna [6].

Awalnya arsitektur aplikasi menggunakan *monolith*, namun terdapat perubahan signifikan dalam dunia IT dimana arsitektur banyak berpindah menjadi *microservices*, arsitektur ini memiliki tujuan yaitu untuk memecah satu aplikasi besar menjadi *service-service* kecil [7]. Pengimplemetasian arsitektur *microservices* dianggap sangat sesuai dengan budaya DevOps, hal ini dikarenakan *microservices* mempermudah *developer* dalam pembuatan aplikasi sesuai dengan kemampuannya, serta meringankan *operation* melakukan *monitoring* aplikasi yang telah berbentuk *service-service* kecil [8]. Penerapan arsitektur *microservices* dengan menggunakan teknologi kontainerisasi, akan membantu mengisolasi sebuah proses dari proses-proses lainnya, serta menciptakan suatu lingkungan yang sesuai dibutuhkan oleh *services* seperti *dependency*, *library*, dan OS *packages* [9]. Kontainerisasi akan membantu skalabilitas pengembangan aplikasi, serta dapat menekan biaya pengembangan secara tradisional, dengan cara memanfaatkan sumber daya (CPU, memori, dan swap) secara maksimal [10]. Hadirnya kontainerisasi tentu juga perlu untuk memperhatikan mengenai keamanan serta risiko yang akan timbul dari penggunaan teknologi tersebut [11].

Dengan berkembangnya teknologi informasi mengiringi peningkatan kejahatan di dunia digital atau dikenal dengan istilah cybercrime [12]. Adanya permasalahan yang muncul membuat Pemerintah Indonesia membentuk undang-undang yang mengatur Perlindungan Data Pribadi yang terdapat pada UU No.27 Tahun 2022. Undang-undang ini dibuat untuk menghindari adanya penyalahgunaan serta kebocoran data pribadi oleh penyelenggara sistem elektronik [13]. Adapun beberapa faktor penyebab kurangnya keamanan dalam suatu aplikasi, seperti terdapat kesalahan penulisan kode program serta *misconfiguration*, adanya hal tersebut dimanfaatkan oleh penyerang untuk mengirim serangan [14]. Munculnya masalah tersebut, maka peneliti melakukan penelitian dengan judul "Adopsi DevSecOps Untuk Mendukung Metode Agile Menggunakan Trivy Sebagai Security Scanner Docker Image Dan Dockerfile", diharapkan penelitian ini dapat mengintegrasikan antara *DevOps* dan *security*, guna untuk mencegah terjadinya kerentanan dalam pengembangan aplikasi berbentuk kontainer sejak dini, dengan begitu aplikasi mampu di *deliver* kepada pengguna dengan cepat sesuai dengan nilai yang terdapat pada *Agile*. Penelitian ini akan berfokus pada *scanning tools* menggunakan Trivy, untuk menampilkan semua *misconfiguration* Dockerfile dan *vulnerability* Docker *image* yang dibuat menggunakan *container engine* Docker. Implementasi *DevSecOps* ini akan dilakukan dengan menggunakan *tools* CI/CD Jenkins. Adapun batasan masalah penelitian ini yaitu hanya akan menampilkan hasil laporan *vulnerability* berdasarkan *database vulnerability* yang dimiliki Trivy.

Pada penelitian yang berjudul Implementasi *Static Application Security Testing* Menggunakan Jenkins CI/CD Berbasis Docker *Container* Pada PT. Emporia Digital Raya, membahas mengenai implementasi *DevSecOps* dengan pendekatan SAST menggunakan SonarQube untuk melakukan *vulnerability assessment* terhadap *source code* aplikasi, dan dijalankan menggunakan CI/CD *tools* Jenkins. Hasil dari penelitian tersebut menunjukkan bahwa pengembangan aplikasi yang menggunakan CI/CD akan mengotomatisasi setiap proses ke dalam *stage*, sehingga apabila terdapat *stage* yang tidak

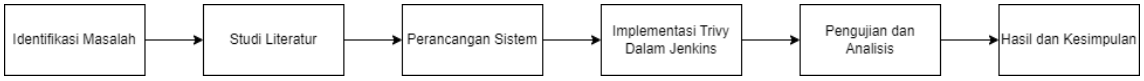
Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 4 No 3, September (2023), pp. 856-863
DOI : https://doi.org/10.35870/jimik.v4i3.291

sesuai dengan kondisi diinginkan, maka akan mengagalkan *pipeline* dan tidak akan dilanjutkan ke *stages* berikutnya. Sehingga, pada penelitian tersebut dapat diasumsikan dengan adanya *stage* yang melakukan SAST akan mempermudah pendeteksian *vulnerability* aplikasi sejak dini, sehingga tidak mempersulit untuk memperbaiki *vulnerability* aplikasi yang sudah berjumlah besar setelah aplikasi hendak dirilis pada *production* [15]. Penelitian tersebut memiliki keselarasan dengan penelitian ini, dimana penelitian tersebut melakukan SAST di dalam Jenkins *pipeline*. Namun, penelitian tersebut melakukan *source code* analisis menggunakan SonarQube untuk mendeteksi *bugs* dan *code smell*, berbeda dengan penelitian ini yang melakukan pendeteksian *vulnerability* pada Docker *images* menggunakan Trivy. Selain itu, penelitian tersebut memiliki keselarasan untuk melakukan SAST dalam Fase *Continuous Integration*. Namun, penelitian tersebut melakukan *vulnerability scanner* pada tahapan *code* aplikasi, sedangkan penelitian ini melakukan *vulnerability scanner* setelah tahapan *code* aplikasi.

Pada penelitian yang berjudul *Vulnerability Management* Pada *Vulnerable* Docker Menggunakan Clair Scanner Dan Joomscan Berdasarkan Standar GSA CIO-IT Security-17-80, membahas mengenai implementasi Clair Scanner dan Joomscan sebagai *vulnerability scanner* pada aplikasi Joomla yang berbasis CMS, dan aplikasi dibangun menggunakan *container engine* Docker. Hasil dari penelitian tersebut menunjukkan bahwa *scanning* yang dilakukan setelah *update* pada Joomla serta menaikkan versi Docker images yang dipakai, maka jumlah *vulnerability* yang didapat lebih sedikit dibandingkan dengan sebelum di lakukannya *update*. Sehingga diasumsikan bahwa Clair Scanner belum melakukan update pada *database vulnerability* yang mereka miliki [16]. Penelitian tersebut memiliki keselarasan dengan penelitian ini, dimana penelitian tersebut melakukan pendeteksian *vulnerability* pada Docker *images*. Namun, *tools* yang digunakan dalam penelitian ini memiliki perbedaan dalam penggunaan *database vulnerability*. Clair Scanner melakukan pendeteksian *vulnerability* berdasarkan *database* CVE yang diluncurkan oleh MITRE Corporation, berbeda dengan Trivy yang menggunakan *database* NVD yang di luncurkan oleh National Institute of Standards and Technology (NIST) Computer Security Division. Penelitian ini melakukan pembuktian apakah Trivy memiliki kelemahan yang sama seperti Clair Scanner dalam *database vulnerability* yang kurang up to date.

2. Metode Penelitian

Dalam suatu penelitian diperlukan sebuah tahapan-tahapan yang berguna untuk menjadi acuan dalam melakukan penelitian. Berikut merupakan metodologi penelitian yang dilakukan.



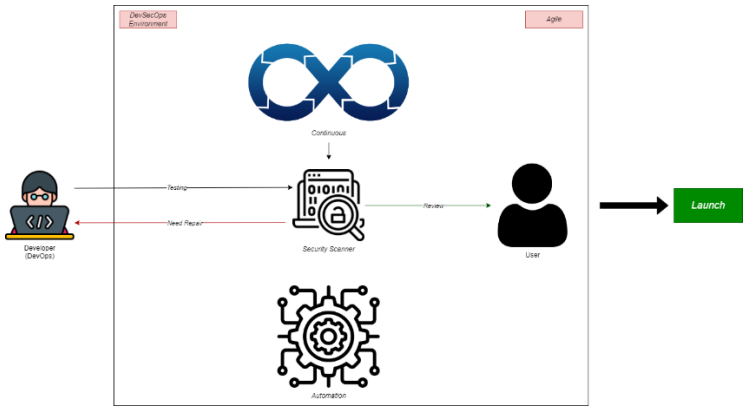
Gambar 1. Metodologi Penelitian

Tahapan awal yang dilakukan untuk mencapai tujuan dari penelitiannya adalah melakukan identifikasi masalah, dimana masalah yang didentifikasi adalah mengapa perlu menerapkan budaya *DevSecOps* didalam *Agile* SDLC. Kemudian, dilanjutkan ke tahapan studi literatur, pada tahapan ini akan dilakukannya riset mengenai permasalahan yang telah diidentifikasi sebelumnya serta mencari solusi yang dapat dilakukan untuk menyelesaikan masalah tersebut berdasarkan literatur yang sesuai serta didukung oleh penelitian-penelitian serupa yang pernah dilakukan sebelumnya. Apabila studi literatur yang dilakukan oleh sudah memadai dan telah menemukan solusi yang tepat untuk menyelesaikan masalah yang telah diidentifikasi, maka akan dilanjutkan ke dalam tahapan perancangan arsitektur sistem. Setelah semua persiapan yang dibutuhkan untuk melakukan penelitian ini telah dilakukan, maka tahapan selanjutnya yaitu melakukan implementasi dari perancangan sistem yang telah dibuat agar penerapan dari budaya *DevSecOps* ini bisa dilakukan secara otomatisasi.

Setelah implementasi telah dilakukan, akan dilakukannya pengujian dan analisis hasil dari perbandingan dua proyek yaitu proyek yang menggunakan Trivy sebagai *security scanner* Dockerfile dan Docker *image* dengan proyek yang tidak mengimplementasikan Trivy. Tahapan terakhir yang

dilakukan oleh adalah mengangkat kesimpulan apakah penggunaan Trivy dalam CI/CD *pipeline* itu tepat untuk diterapkan.

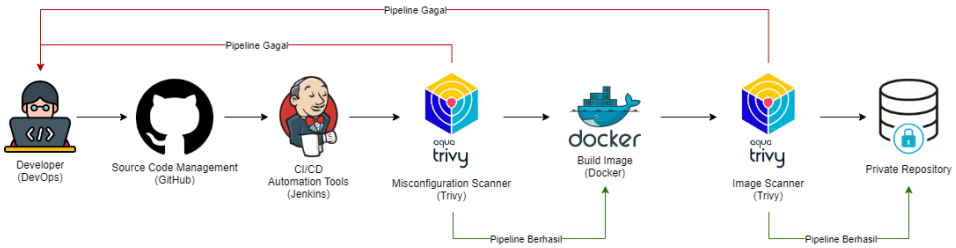
3. Hasil dan Pembahasan



Gambar 2. Alur dan Perancangan Sistem

Dengan perancangan lingkungan pengembangan aplikasi seperti pada gambar diatas, maka proses pengembangan aplikasi akan menjadi lebih efektif dan efisien. *Developer* akan sangat terbantu karena pekerjaan yang dilakukan secara berulang-ulang dapat dilakukan secara otomatisasi. Dalam arsitektur ini pula menyisipkan proses *security scanner*. Sehingga, saat *developer* ingin melakukan *build* kode program yang telah dibuat untuk menjadi sebuah *image*, nantinya akan dilakukan pengecekan terlebih dahulu apakah konfigurasi tersebut memiliki kerentanan yang akan berdampak kedepannya terhadap aplikasi. Apabila terdapat kerentanan, maka *developer* dapat segera untuk memperbaiki konfigurasi tersebut hingga proses *security scanner* bisa dilewati. Begitu juga dengan proses *image scanner*, yang nanti akan digunakan untuk men-*deploy* aplikasi tersebut secara kontainerisasi.

Alur pengembangan aplikasi seperti ini sangat mendukung metode *Agile*. Dengan penyisipan *security scanner* pada tiap-tiap proses yang ada pada pengembangan aplikasi, diharapkan akan mendapatkan hasil *security issues* yang lebih minim pada tahapan akhir pengembangan aplikasi. Sehingga, aplikasi dapat di *review* oleh *user* secepatnya, untuk menentukan apakah aplikasi tersebut telah sesuai dan bisa untuk segera di rilis. Berikut merupakan alur dari implementasi DevSecOps pada pipeline.



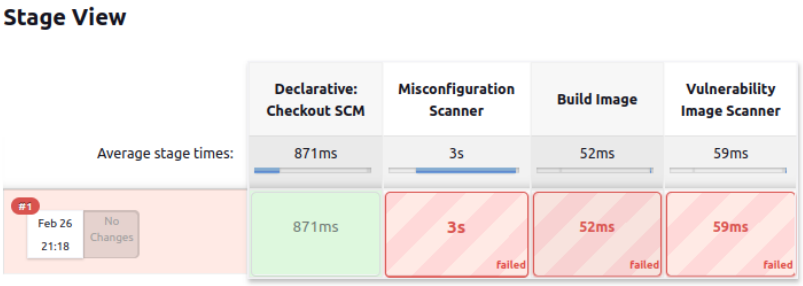
Gambar 3. Arsitektur Sistem

Proses dimulai dengan *developer* yang melakukan pembuatan kode program dari masing-masing lokal komputer dan di lakukan *push* untuk menyimpan kode program pada kode repositori GitHub. Jenkins akan melakukan penarikan kode program yang ada pada Github untuk dilakukan *misconfiguration scanner* pada Dockerfile yang telah dibuat oleh *developer*. Apabila dalam proses pemindaian ini ditemukan *security issues*, maka Jenkins akan mengagalkan *pipeline*, agar *developer* bisa memperbaiki Dockerfile tersebut. Setelah tahapan *misconfiguration scanner* telah dilewati, maka akan

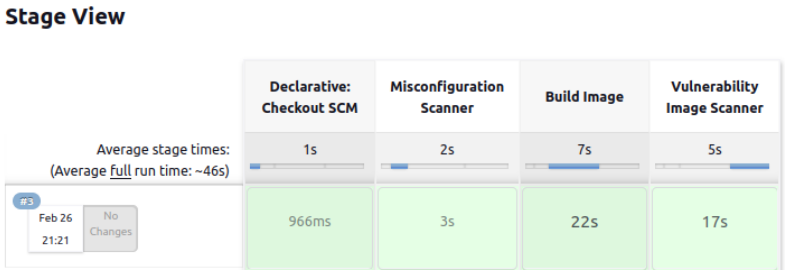
dilanjutkan pada tahapan *build image* dari Dockerfile dan kode program yang telah dibuat oleh *developer* untuk menjadi sebuah *image* menggunakan Docker.

Proses akan dilanjutkan pada tahapan *image scanner* untuk melakukan pemindaian apakah *base image* yang digunakan oleh *developer* untuk membuat *image* tersebut memiliki *security issues*. Apabila dalam proses pemindaian ini ditemukan *security issues*, maka Jenkins akan mengagalkan *pipeline* untuk *developer* bisa mengganti *base image* yang digunakan untuk membuat *image* tersebut. Setelah tahapan *image scanner* ini telah dilewati, maka proses akan dilanjutkan dengan menyimpan *image* yang telah melewati *security scanner* tersebut untuk disimpan pada *private repository*.

Pada CI/CD *tools* yang dalam penelitian ini menggunakan Jenkins, akan terdapat *pipeline* yang memiliki banyak *stages* atau tahapan yang perlu untuk di lewati. Berikut adalah pipeline yang terdapat stages untuk melakukan misconfiguration scanner dan vulnerability image scanner.



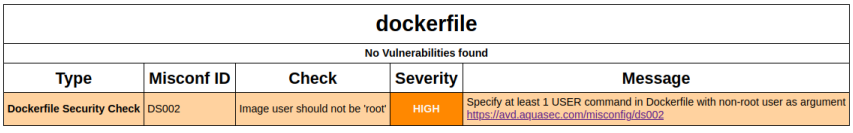
Gambar 4. *Pipeline* Gagal



Gambar 5. *Pipeline* Berhasil

Penjelasan dari gambar 4 menunjukkan bahwa *pipeline* yang gagal melewati proses *misconfiguration scanner* dan *vulnerability image scanner*. Oleh karena itu, *developer* bisa mengetahui dan memperbaiki *security issues* yang ditemukan untuk mendapatkan hasil *pipeline* yang berhasil seperti pada gambar 5.

Temuan dari *misconfiguration scanner* dan *vulnerability image scanner* dapat dilihat dalam bentuk format HTML. Hal ini akan membantu *developer* lebih mudah dalam membaca hasil *security issues*. Hasil yang diperoleh oleh Trivy akan menujukkan alasan mengapa *pipeline* Jenkins menjadi gagal dan berhasil.

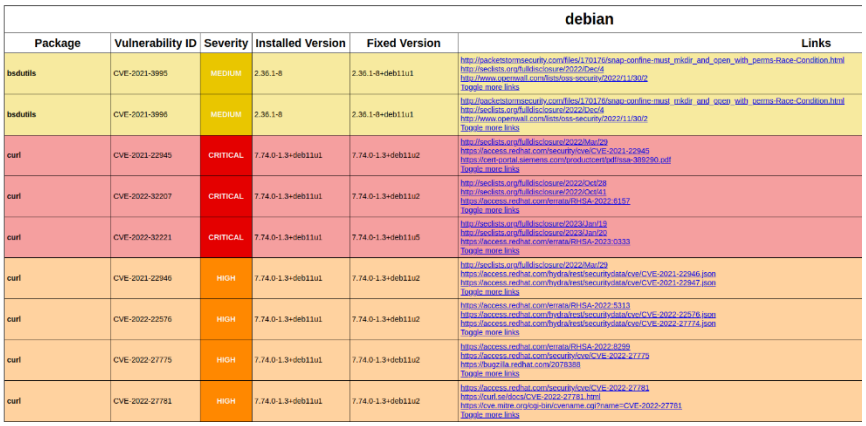


dockerfile

No Vulnerabilities found

Type	Misconf ID	Check	Severity	Message
Dockerfile Security Check	DS002	Image user should not be 'root'	HIGH	Specify at least 1 USER command in Dockerfile with non-root user as argument https://avd.aquasec.com/misconfig/ds002

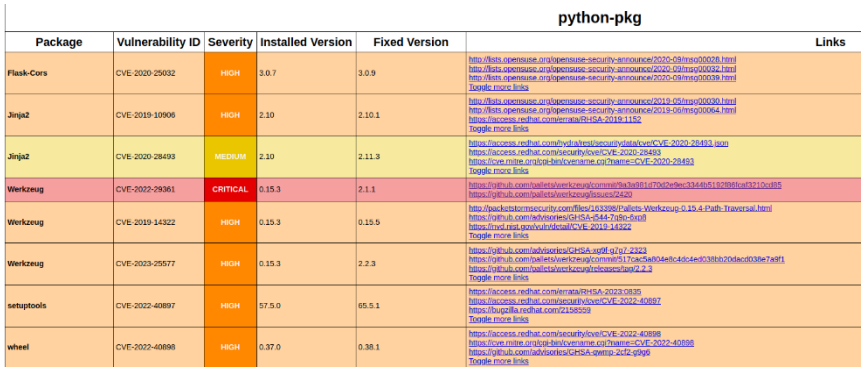
Gambar 6. *Misconfiguration* Dockerfile



debian

Package	Vulnerability ID	Severity	Installed Version	Fixed Version	Links
bsdutils	CVE-2021-3995	MEDIUM	2.36.1-8	2.36.1-8+deb11u1	http://packetstormsecurity.com/files/170176/snap-confine-must_mkdir_and_open_with_perms-Race-Condition.html http://seclists.org/fulldisclosure/2022/Dec/4 http://www.openwall.com/lists/oss-security/2022/11/30/2 Toggle more links
bsdutils	CVE-2021-3996	MEDIUM	2.36.1-8	2.36.1-8+deb11u1	http://packetstormsecurity.com/files/170176/snap-confine-must_mkdir_and_open_with_perms-Race-Condition.html http://seclists.org/fulldisclosure/2022/Dec/4 http://www.openwall.com/lists/oss-security/2022/11/30/2 Toggle more links
curl	CVE-2021-22945	CRITICAL	7.74.0-1.3+deb11u1	7.74.0-1.3+deb11u2	http://seclists.org/fulldisclosure/2022/Mar/29 https://access.redhat.com/security/cve/CVE-2021-22945 https://cert-portal.siemens.com/productcert/pdf/ssa-389290.pdf Toggle more links
curl	CVE-2022-32207	CRITICAL	7.74.0-1.3+deb11u1	7.74.0-1.3+deb11u2	http://seclists.org/fulldisclosure/2022/Oct/28 http://seclists.org/fulldisclosure/2022/Dec/41 https://access.redhat.com/errata/RHSA-2022:6157 Toggle more links
curl	CVE-2022-32221	CRITICAL	7.74.0-1.3+deb11u1	7.74.0-1.3+deb11u5	http://seclists.org/fulldisclosure/2023/Jan/19 http://seclists.org/fulldisclosure/2023/Jan/20 https://access.redhat.com/errata/RHSA-2023:0333 Toggle more links
curl	CVE-2021-22946	HIGH	7.74.0-1.3+deb11u1	7.74.0-1.3+deb11u2	http://seclists.org/fulldisclosure/2022/Mar/29 https://access.redhat.com/hydra/rest/securitydata/cve/CVE-2021-22946.json https://access.redhat.com/hydra/rest/securitydata/cve/CVE-2021-22947.json Toggle more links
curl	CVE-2022-22576	HIGH	7.74.0-1.3+deb11u1	7.74.0-1.3+deb11u2	https://access.redhat.com/errata/RHSA-2022:5313 https://access.redhat.com/hydra/rest/securitydata/cve/CVE-2022-22576.json https://access.redhat.com/hydra/rest/securitydata/cve/CVE-2022-27774.json Toggle more links
curl	CVE-2022-27775	HIGH	7.74.0-1.3+deb11u1	7.74.0-1.3+deb11u2	https://access.redhat.com/errata/RHSA-2022:8299 https://access.redhat.com/security/cve/CVE-2022-27775 https://bugzilla.redhat.com/2078388 Toggle more links
curl	CVE-2022-27781	HIGH	7.74.0-1.3+deb11u1	7.74.0-1.3+deb11u2	https://access.redhat.com/security/cve/CVE-2022-27781 https://curl.se/docs/CVE-2022-27781.html https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-27781 Toggle more links

Gambar 7. *Vulnerability Image* Tingkat OS



python-pkg

Package	Vulnerability ID	Severity	Installed Version	Fixed Version	Links
Flask-Cors	CVE-2020-25032	HIGH	3.0.7	3.0.9	http://lists.opensuse.org/opensuse-security-announce/2020-09/msg00028.html http://lists.opensuse.org/opensuse-security-announce/2020-09/msg00032.html http://lists.opensuse.org/opensuse-security-announce/2020-09/msg00039.html Toggle more links
Jinja2	CVE-2019-10906	HIGH	2.10	2.10.1	http://lists.opensuse.org/opensuse-security-announce/2019-05/msg00030.html http://lists.opensuse.org/opensuse-security-announce/2019-06/msg00064.html https://access.redhat.com/errata/RHSA-2019:1152 Toggle more links
Jinja2	CVE-2020-28493	MEDIUM	2.10	2.11.3	https://access.redhat.com/hydra/rest/securitydata/cve/CVE-2020-28493.json https://access.redhat.com/security/cve/CVE-2020-28493 https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-28493 Toggle more links
Werkzeug	CVE-2022-29361	CRITICAL	0.15.3	2.1.1	https://github.com/pallets/werkzeug/commit/5a3a981d70d2e5ec3344b5192f86fcaf3210cd85 https://github.com/pallets/werkzeug/issues/2420
Werkzeug	CVE-2019-14322	HIGH	0.15.3	0.15.5	http://packetstormsecurity.com/files/163398/Pallets-Werkzeug-0.15.4-Path-Traversal.html https://github.com/advisories/GHSA-044-7q8c-6xpf https://nvd.nist.gov/vuln/detail/CVE-2019-14322 Toggle more links
Werkzeug	CVE-2023-25577	HIGH	0.15.3	2.2.3	https://github.com/advisories/GHSA-xg9f-g7g7-2323 https://github.com/pallets/werkzeug/commit/517ca5a604ef1c4dc4ed038bb20dacd03fe7a9f1 https://github.com/pallets/werkzeug/releases/tag/2.2.3 Toggle more links
setuptools	CVE-2022-40897	HIGH	57.5.0	65.5.1	https://access.redhat.com/errata/RHSA-2023:0835 https://access.redhat.com/security/cve/CVE-2022-40897 https://bugzilla.redhat.com/2158359 Toggle more links
wheel	CVE-2022-40898	HIGH	0.37.0	0.38.1	https://access.redhat.com/security/cve/CVE-2022-40898 https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-40898 https://github.com/advisories/GHSA-qwmp-2cf2-g9g9 Toggle more links

Gambar 8. *Vulnerability Image* Tingkat Aplikasi

Ketiga gambar diatas adalah informasi yang berisikan hasil temuan *security issues* dan juga informasi untuk mengatasi *issues* tersebut. Hasil temuan akan mempengaruhi *pipeline* Jenkins yang ada pada gambar 4 dan 5. Pada gambar 6 menginformasikan bahwa kontainer harus di jalankan oleh *user non root* untuk meminimalisir terjadinya eksploitasi oleh penyerang yang telah berhasil mendapatkan akses ke dalam kontainer. Hal ini akan membatasi langkah penyerang dalam mengakses data sensitif milik host system yang hanya bisa dilakukan oleh user root.

Pada gambar 7 sesuai dengan CVE-2022-32221 menginformasikan bahwa *package* curl *version* 7.74.0-1.3+deb11u1 memiliki kerentanan saat melakukan HTTP(S) *transfer*, libcurl terkadang keliru ketika menggunakan read callback (CURLOPT_READFUNCTION) saat meminta data untuk dikirim. Kerentanan ini dapat menyebabkan aplikasi berperilaku buruk dan mengirimkan data yang salah. Pada gambar 8 sesuai dengan CVE-2022-29361 menginformasikan bahwa *package* Werkzeug *version* 0.15.3 memiliki kerentanan terhadap HTTP *request smuggling*, yang disebabkan oleh penguraian HTTP *request* yang tidak tepat. Dengan mengirimkan HTTP *request* yang dibuat khusus, penyerang dapat mengeksploitasi kerentanan ini untuk meracuni *web cache*, melewati *web application firewall*, dan melakukan serangan XSS.

4. Kesimpulan dan Saran

Dari penelitian ini dapat disimpulkan bahwa implementasi *DevSecOps* menggunakan Trivy yang di intergrasikan pada Jenkins akan membentuk SDLC yang bersifat *continuous* sesuai dengan prinsip *Agile*. Sifatnya yang *continuous* dapat memberikan *feedback* yang lebih cepat sehingga *developer* dapat mengetahui dan memperbaiki *security issues* yang ditemukan untuk menghindari penumpukan temuan diakhir SDLC. Hasil temuan dari Trivy yang sudah dalam bentuk format HTML serta sudah tertera

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 4 No 3, September (2023), pp. 856-863
DOI : https://doi.org/10.35870/jimik.v4i3.291

informasi untuk memperbaiki *security issues* tersebut akan memudahkan *developer* dalam membaca dan menganalisa perbaikan yang harus dilakukan. Adapun saran dari penelitian ini adalah dapat mengembangkan tingkat kedewasaan *DevSecOps* ini menjadi lebih baik lagi seperti menambahkan *source code review* untuk mengetahui *code smell* dan *bugs* yang bisa terjadi pada aplikasi. Tahapan DAST juga baik di tambahkan dalam *pipeline* untuk menirukan perilaku Pentester dalam melakukan *penetration testing* secara black box.

5. Daftar Pustaka

[1] Permana, A. Y., & Romadlon, P., (2019). Perancangan Sistem Informasi Penjualan Perumahan Mengunakan Metode SDLC Pada PT. Mandiri Land Proseperous Berbasis Mobile. Jurnal Teknologi Pelita Bangsa, 84(10), pp. 1511–1518. DOI: https://doi.org/10.1134/s0320972519100129

[2] Murdiani, D., Yudhana, A., & Sunardi, S., (2020). Implementasi Agile Method dalam Pengembangan Jurnal Elektronik di Lembaga Penelitian Non Pemerintahan (NGO). Jurnal Teknologi Informasi Dan Ilmu Komputer, 7(4), pp. 709. DOI: https://doi.org/10.25126/jtiik.2020741839

[3] Sunardi, S., & Fadli, S., (2018). IDENTIFIKASI MASALAH PENERAPAN METODE AGILE (SCRUM) PADA PENGEMBANGAN PERANGKAT LUNAK DI PERGURUAN TINGGI (Studi Kasus Universitas Nahdlatul Ulama Nusa Tenggara Barat). Jurnal Manajemen Informatika Dan Sistem Informasi, 1(2), pp. 14. DOI: https://doi.org/10.36595/misi.v1i2.37

[4] Hemon, A., Lyonnet, B., Rowe, F., & Fitzgerald, B., (2020). From Agile to DevOps: Smart Skills and Collaborations. Information Systems Frontiers, 22(4), pp. 927–945. DOI: https://doi.org/10.1007/s10796-019-09905-1

[5] Tohirin, T., Utami, S. F., Widianto, S. R., & Mauludyansah, W. Al., (2020). Implementasi DevOps Pada Pengembangan Aplikasi e-Skrining Covid-19. Multinetics, 6(1), pp. 15–20. DOI: https://doi.org/10.32722/multinetics.v6i1.2764

[6] Melgar, A. S., & Osores, J. J. V., (2021). DevOps as a culture of interaction and deployment in an insurance company. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(4), pp. 1701–1708. DOI: https://doi.org/10.17762/turcomat.v12i4.1429

[7] Saputra, M. H. K., & Nabil, L. M., (2021). Penerapan Arsitektur Microservice Pada Sistem Tata Kelola Matakuliah Proyek Politeknik Pos Indonesia. Teknik Informatika, 13(3), pp. 22–28.

[8] Khalyly, B. El, Belangour, A., Erraissi, A., & Banane, M., (2020). Devops and Microservices Based Internet of Things Meta-Model. International Journal of Emerging Trends in Engineering Research, 8(9), pp. 6254–6266. DOI: https://doi.org/10.30534/ijeter/2020/217892020

[9] Putra, R. A., (2018). Analisa Implementasi Arsitektur Microservoces Berbasis Kontainer Pada Komunitas Pengembang Perangkat Lunak Sumber Terbuka (OpenDayLight DevOps Community). Jurnal Sistem Infomasi Teknologi Informasi Dan Komputer (Just It) Universitas Bina Nusantara Magister Manajemen Sistem Informasi Jakarta, pp. 150–162.

[10] Ariadi, F., Iswahyudi, C., Nurnawati, E., Informatika, J., & Akprind, I., (2020). Penerapan Docker Container Sebagai Teknologi Ramah Skalabilitas Dibanding Teknik Virtualisasi Untuk Membangun Website Di Ubuntu 18.04.4 Lts. Jurnal JARKOM, 8(2), pp. 47–57.

[11] Hanifah, F., Budiyono, A., & Widjajarto, A., (2021). Analisa Kerentanan Pada Vulnerable Docker Menggunakan Alienvault Dan Docker Bench For Security Dengan Acuan Framework CIS Control. E-Proceeding of Engineering, 8(5), pp. 8879–8885. DOI: https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/15914

[12] Palinggi, S., Palelleng, S., & Allolinggi, L. R., (2020). Peningkatan Rasio Kejahatan Cyber Dengan Pola Interaksi Sosio Engineering Pada Periode Akhir Era Society 4.0 Di Indonesia. Jurnal Ilmiah Dinamika Sosial, 4(1), pp. 145. DOI: https://doi.org/10.38043/jids.v4i1.2314

[13] Elis, E., & Hamimah, S., (2022). Urgensi Undang-Undang Perlindungan Data Pribadi Dalam Menjamin Keamanan Data Pribadi Sebagai Pemenuhan Hak Atas Privasi Masyarakat Indonesia. Jurnal Rechten : Riset Hukum Dan Hak Asasi Manusia, 3(2), pp. 1–6. DOI: https://doi.org/10.52005/rechten.v3i2.34

[14] Edy Listartha, I. M., Premana Mitha, I. M. A., Aditya Arta, M. W., & Yuda Arimika, I. K. W., (2022). Analisis Kerentanan Website SMA Negeri 2 Amlapura Menggunakan Metode OWASP (Open Web Application Security Project). Simkom, 7(1), pp. 23–27. DOI: https://doi.org/10.51717/simkom.v7i1.63

[15] Shama, A. M., & W. Chandra, D., (2021). Implementasi Static Application Security Testing Menggunakan Jenkins Ci/Cd Berbasis Docker Container Pada Pt. Emporia Digital Raya. Jurnal Ilmiah Informatika, 9(02), pp. 95–99. DOI: https://doi.org/10.33884/jif.v9i02.3769

[16] Ramadhan, R. S., Widjajarto, A., & Almaarif, A., (2022). Vulnerability Management Pada Vulnerable Docker Menggunakan Clair Scanner Dan Joomscan Berdasarkan Standar GSA CIO-IT Security -17-80. 4(1), pp. 85–93. DOI: https://doi.org/10.30865/json.v4i1.4789.