# Analisis Kerentanan Pada *Vulnerable Docker* Menggunakan *Scanner* Openvas Dan Docker Scout Dengan Acuan Standar PTES

1st Dimas Ismunandar
*Program Studi Teknik Informatika*
*Universitas Telkom Purwokerto*
Banyumas, Jawa Tengah, Indonesia
[1]dimasismunandar@students.telkomuniversity.ac.id

2nd Alon Jala Tirta Segara S.Kom., M.Kom.
*Program Studi Teknik Informatika*
*Universitas Telkom Purwokerto*
Banyumas, Jawa Tengah, Indonesia
[2]alonhs@telkomuinversity.ac.id

**Abstrak ~ *Administrator*, pengembang, dan pengembang proyek open source dapat menggunakan Docker untuk membuat, mengemas, dan menjalankan aplikasi dalam container dalam berbagai lingkungan. Container memungkinkan penggunaan terpisah dan bergantung pada lingkungan, tetapi mereka juga menimbulkan risiko keamanan jika tidak dikelola dengan baik. Mengetahui dan mengantisipasi berbagai kerentanan pada gambar Docker sangat penting untuk keamanan sistem. Jumlah container yang digunakan dalam proses produksi menunjukkan bahwa ancaman terhadap keamanan teknologi ini semakin meningkat. Pemindaian kerentanan adalah prosedur penting untuk memastikan bahwa gambar yang digunakan tidak mengandung komponen berbahaya atau celah keamanan. Karena beberapa alat pemindai kerentanan tidak efektif dalam mendeteksi risiko, alat pemindai kerentanan yang paling umum harus dievaluasi. Studi ini menyelidiki masalah keamanan *image* Docker melalui analisis dan perbandingan alat pemindai kerentanan Docker Scout dan OpenVAS. Selama proses pemindaian, standar PTES (Pelaksanaan Pengujian Penetrasi) digunakan sebagai standar pelaksanaan pengujian penetrasi. Setiap aplikasi diuji pada sejumlah gambar Docker untuk mengidentifikasi jenis kerentanan.Hasil penelitian menunjukkan bahwa setiap alat memiliki kemampuan yang lebih baik untuk menemukan jenis kerentanan tertentu. Sementara Docker Scout berfokus pada kerentanan *image* spesifik, OpenVAS lebih baik dalam menemukan kerentanan sistem secara keseluruhan. Untuk meningkatkan keamanan container Docker, saran keamanan yang dihasilkan oleh kedua alat ini dapat digunakan saat membuat rencana mitigasi.**

***Kata Kunci* — Docker, PTES, OpenVAS, Docker Scout, Kerentanan.**

## I. PENDAHULUAN

Internet dan teknologi informasi kini menjadi bagian penting dalam mendukung aktivitas bisnis dan komunikasi jarak jauh. Namun, kebutuhan infrastruktur teknologi seperti server berspesifikasi tinggi menjadi kendala, terutama bagi usaha kecil dan startup karena tingginya biaya. Salah satu solusi untuk mengatasi masalah ini adalah penerapan teknologi virtualisasi berbasis container seperti Docker, yang memungkinkan efisiensi dalam deployment aplikasi dan pengelolaan infrastruktur[1].

Meskipun Docker memberikan banyak kemudahan, platform ini tetap memiliki potensi kerentanan, khususnya pada container dan image yang digunakan.[2] Kerentanan ini dapat dimanfaatkan oleh pihak tidak bertanggung jawab dan mengancam data penting seperti informasi pelanggan dan strategi bisnis. Oleh karena itu, diperlukan proses pengujian keamanan sistem yang sistematis.[3]

Penelitian ini bertujuan untuk menganalisis efektivitas tools OpenVAS dan Docker Scout dalam mendeteksi celah keamanan pada Docker, serta mengimplementasikan standar *Penetration Testing Execution Standard* (PTES) sebagai kerangka kerja dalam proses pengujian penetrasi. Hasilnya diharapkan dapat memberikan rekomendasi mitigasi untuk meningkatkan keamanan sistem berbasis container.

## II. KAJIAN TEORI

### A. Docker Container

Docker adalah platform kontainerisasi yang semakin populer untuk web hosting, terutama karena kemampuannya dalam mendukung penyebaran aplikasi web dan lingkungan pendukung seperti web server, database, dan dependensi lainnya. Docker mempermudah proses pemaketan, deployment, dan distribusi aplikasi, sehingga dapat mengatasi berbagai masalah seperti perbedaan versi software, kekurangan file, serta konfigurasi yang tidak konsisten.[4]

### B. OpenVAS

OpenVAS adalah pemindai kerentanan yang lengkap, dengan kemampuan melakukan pengujian baik yang diautentikasi maupun tidak, mendukung berbagai protokol internet dan industri, serta dapat disesuaikan untuk pemindaian skala besar. Alat ini menggunakan bahasa pemrograman internal yang kuat untuk mendeteksi berbagai jenis kerentanan, dan memperoleh tes dari feed yang diperbarui setiap hari serta memiliki riwayat panjang.[1]

### C. Docker Scout

Docker Scout adalah alat bawaan yang memungkinkan pemindaian otomatis terhadap gambar Docker untuk mendeteksi kerentanan, dengan mengandalkan database yang harus selalu diperbarui agar hasilnya akurat. Alat ini memiliki tiga fitur utama, yaitu pemindaian dasar pada repositori Docker Hub, pemindaian otomatis terhadap *image* Docker, serta pemindaian melalui pengunggahan

*image* ke Docker Hub untuk melihat laporan kerentanan. Selain itu, Docker Scout juga memberikan informasi terkini mengenai kerentanan beserta langkah-langkah remediasi guna meningkatkan keamanan sistem.[1]

D. PTES (*Penetration Testing Execution Standart*)

PTES, standar tahun 2010, dapat digunakan untuk melakukan analisis dan audit sistem keamanan website. Memberikan hasil pengujian celah keamanan server adalah tujuan pembuatan standar ini. PTES memiliki 7 tahapan implementasi, mulai dari *pre-engagement interaction* hingga *reporting*. Tahapan-tahapan ini adalah sebagai berikut: *pre-engagement interactions, gathering information, threat modeling, vulnerabilities analysis, exploitation, post-exploitation, and reporting*.[5]

E. Kerentanan System (System Vulnerabilities)

Kerentanan sistem adalah kelemahan dalam konfigurasi, perangkat lunak, atau arsitektur sistem yang memungkinkan penyerang mendapatkan akses tidak sah atau mengeksekusi perintah berbahaya. Dalam konteks Docker, kerentanan dapat muncul dari image yang tidak aman, konfigurasi volume yang lemah, hingga Docker daemon yang diekspos secara langsung. Identifikasi kerentanan ini penting untuk mencegah potensi eksploitasi.[6].

F. SQL Injection

SQL Injection adalah teknik serangan keamanan yang dilakukan dengan menyisipkan perintah SQL berbahaya ke dalam input aplikasi, seperti formulir login atau URL. Serangan ini memanfaatkan kelemahan dalam validasi input dan dapat digunakan untuk membaca, mengubah, atau menghapus data dalam database yang tidak seharusnya diakses. Contohnya, penyerang dapat menggunakan input seperti `' OR '1'='1` untuk melewati proses autentikasi dan mendapatkan akses ilegal ke sistem.[7].

## III. METODE

A. Topologi Perancangan

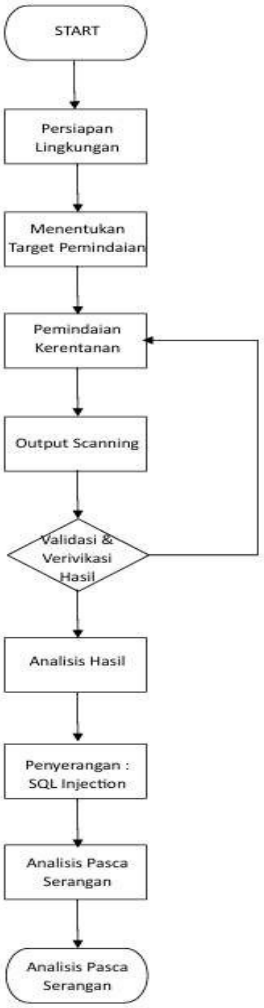
Topologi Perancangan adalah gambaran sistem yang akan dibangun dalam proses penelitian. Penelitian ini menggunakan pendekatan virtualisasi, di mana sistem utama (host) menjalankan beberapa sistem operasi virtual (guest) menggunakan VirtualBox. Setiap sistem operasi virtual memiliki peran masing-masing dalam mendukung proses pengujian kerentanan.


GAMBAR 1.
TOPOLOGI PERANCANGAN

Pada gambar 1 diatas topologi ini menggambarkan hubungan antara dua mesin virtual, yaitu Kali Linux yang berfungsi sebagai mesin pemindai dengan menggunakan tools OpenVAS. Ubuntu yang dijalankan sebagai mesin target, di mana container Docker yang rentan (vulnerable Docker) berada dan dijadikan objek uji. Komunikasi antara Kali Linux dan Ubuntu dilakukan dalam satu jaringan yang sama (internal network). Kali Linux mengirimkan permintaan pemindaian ke Ubuntu, dan OpenVAS kemudian menganalisis layanan dan sistem yang berjalan pada container Docker.

B. Diagram Alir

Diagram Alir adalah diagram alir penelitian menjelaskan langkah-langkah sistematis yang dilakukan dalam proses pengujian kerentanan pada lingkungan Docker menggunakan dua alat bantu, yaitu OpenVAS dan Docker Scout.




GAMBAR 2.
DIAGRAM ALIR

Setiap tahapan bertujuan untuk memastikan bahwa proses pemindaian berjalan secara terstruktur dan menghasilkan data yang valid untuk dianalisis lebih lanjut. Berikut adalah tahapan diagram alir

1. Persiapan Lingkungan

Persiapan lingkungan merupakan tahap awal dalam proses pengujian kerentanan yang bertujuan untuk membangun infrastruktur pengujian yang sesuai dan mendukung kebutuhan penelitian. Lingkungan pengujian

dibangun dengan menggunakan perangkat lunak virtualisasi VirtualBox, yang memungkinkan sistem utama (host) menjalankan dua sistem operasi virtual (guest).

2. Memasukan Target

Pada tahapan ini, target pemindaian kerentanan ditentukan. Target tersebut berupa Docker Image yang telah dikonfigurasi secara sengaja untuk memiliki celah keamanan (vulnerable). Docker container tersebut berjalan di atas sistem operasi Ubuntu dan memiliki IP address statis yang digunakan sebagai acuan dalam proses pemindaian. Identifikasi target ini penting untuk memastikan bahwa proses scanning berlangsung pada sistem yang benar dan sesuai tujuan penelitian.

3. Pemindaian Kerentanan

Setelah target ditentukan, dilakukan proses pemindaian kerentanan menggunakan dua alat utama, yaitu OpenVAS dan Docker Scout. OpenVAS digunakan sebagai platform pemindaian yang mendeteksi kerentanan umum pada sistem berbasis jaringan dan aplikasi, sedangkan Docker Scout difokuskan pada analisis kerentanan pada level image Docker, seperti dependency yang rentan dan konfigurasi yang tidak aman.

4. Output Scanning

Tahap ini menghasilkan data hasil pemindaian yang berisi daftar kerentanan yang ditemukan, tingkat keparahan (severity), serta deskripsi teknis dari masing-masing kerentanan. Informasi yang dihasilkan dapat mencakup CVE (Common Vulnerabilities and Exposures), risiko eksploitasi, dan rekomendasi awal dari tools yang digunakan.

5. Analisis Hasil

Setelah hasil pemindaian diperoleh, dilakukan proses verifikasi untuk memastikan validitas dan akurasi data yang dihasilkan. Jika hasil dianggap kurang akurat atau terdapat kesalahan konfigurasi, maka proses pemindaian akan diulang. Namun jika hasil dinyatakan valid, yaitu sesuai dengan ekspektasi dan target pengujian, maka data pemindaian digunakan sebagai dasar untuk tahap analisis lebih lanjut. Proses ini penting agar penelitian tidak menghasilkan data yang bias atau tidak representatif.

6. Penyerangan SQL Injection

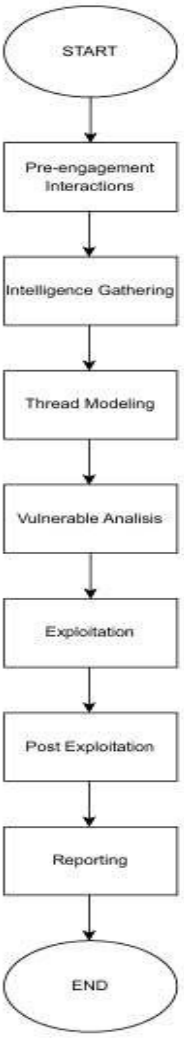
Serangan ini dilakukan untuk menguji sejauh mana celah keamanan tersebut dapat dimanfaatkan oleh pihak yang tidak berwenang. SQL Injection merupakan teknik serangan yang menyisipkan perintah SQL berbahaya ke dalam input pengguna dengan tujuan untuk memanipulasi basis data.

7. Analisis Pasca Penyerangan

Setelah serangan SQL Injection berhasil dilakukan, tahap selanjutnya adalah analisis pasca penyerangan. Tahap ini bertujuan untuk mengevaluasi dampak eksploitasi terhadap sistem target serta mengidentifikasi potensi risiko yang mungkin timbul.

C. Metode Analisis Kerentanan PTES

Selanjutnya adalah tahapan dalam analisa kerentanan PTES digunakan untuk memastikan bahwa seluruh tahapan analisis dilakukan secara komprehensif dan berstandar profesional. PTES membagi proses pengujian menjadi enam tahapan utama, namun dalam konteks penelitian ini, tahapan yang relevan dan diadopsi secara adaptif seperti gambar 3 dibawah ini




GAMBAR 3.
ALUR METODE PTES

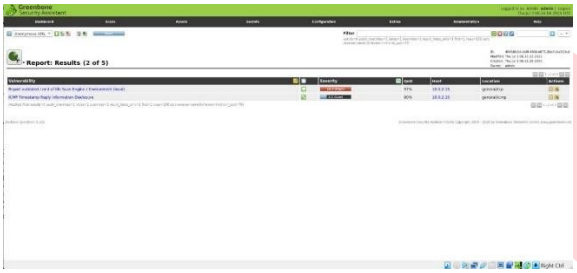
Gambar diatas menunjukkan alur metode PTES (*Penetration Testing Execution Standard*) yang terdiri dari tujuh tahapan utama secara berurutan, dimulai dari START hingga STOP. Proses diawali dengan *Pre-engagement Interactions* yang merupakan tahap kesepakatan awal antara penguji dan pemilik sistem, dilanjutkan dengan *Intelligence Gathering* untuk mengumpulkan informasi teknis tentang target. Tahap berikutnya adalah *Threat Modeling* untuk mengidentifikasi potensi ancaman berdasarkan informasi yang diperoleh, kemudian *Exploitation* untuk menguji kerentanan yang ditemukan. Setelah itu dilakukan *Post Exploitation* guna menganalisis dampak eksploitasi, dan

diakhiri dengan *Reporting* yang berisi dokumentasi seluruh hasil pengujian sebelum proses ditutup. Diagram ini menggambarkan pendekatan sistematis dan profesional dalam proses analisis kerentanan.

## IV. HASIL DAN PEMBAHASAN

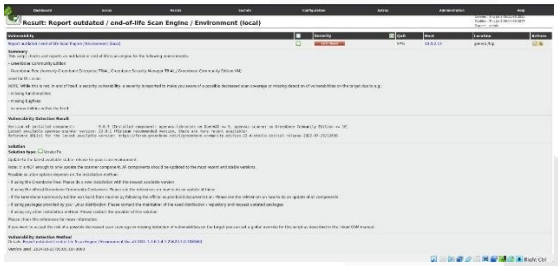
Proses ini dilakukan dengan dua alat yaitu OpenVAS dan Docker Scout Analisis dilakukan dengan mengacu pada standar PTES (*Penetration Testing Execution Standard*) sebagai kerangka kerja pengujian keamanan.

### A. Hasil Scan dengan OpenVas



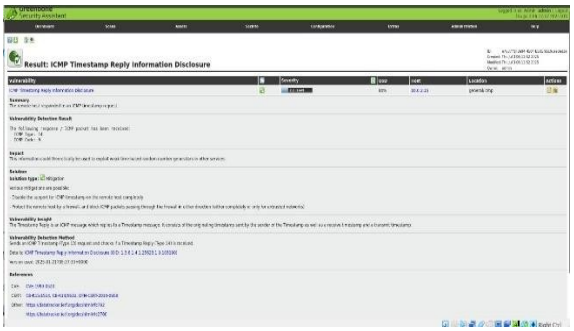

GAMBAR 4.
HASIL SCAN

Pada gambar 4. hasil scan OpenVAS setelah container DVWA dijalankan menjelaskan terdapat 2 kerentanan yang terdeteksi yaitu kerentanan 1, Report outdated/end-of-life Scan Engine/Environment (local) dan kerentanan 2, ICMP Timestamp Reply Information Disclosure. Kerentanan 1 menunjukan severity 10.0(high), Qod 97%, dengan host 10.0.2.15. Untuk kerentanan 2 menunjukan severity 2.1(low), Qod 80% dengan host 10.0.2.15.




GAMBAR 5.
HASIL REPORT OUTDATED

Pada gambar 5. menjelaskan berdasarkan hasil pemindaian yang dilakukan menggunakan *Greenbone Community Edition*, ditemukan dua jenis kerentanan pada host dengan alamat IP 10.0.2.15. Kerentanan pertama adalah penggunaan versi yang sudah usang atau mencapai akhir masa pakai (end-of-life) dari mesin pemindaian (scan engine), yaitu OpenVAS versi 9.0.3. Kerentanan ini diklasifikasikan dengan tingkat keparahan tinggi (severity 10.0) karena versi tersebut tidak lagi menerima pembaruan keamanan dan fungsionalitas, sehingga dapat menyebabkan deteksi kerentanan yang tidak akurat atau bahkan gagal mendeteksi celah keamanan yang ada. Hal ini berisiko besar terhadap keamanan jaringan karena perangkat lunak keamanan yang usang dapat memberi rasa aman yang palsu dan membuka peluang serangan tidak terdeteksi.




GAMBAR 6.
HASIL ICMP

Pada gambar 6. menjelaskan kerentanan kedua yang ditemukan adalah ICMP Timestamp Reply Information Disclosure, yang termasuk dalam kategori informasi sensitif terbuka. Tingkat keparahan dari kerentanan ini tergolong rendah (severity 2.1), namun tetap relevan dalam konteks keamanan karena memberikan informasi waktu sistem yang dapat dimanfaatkan oleh penyerang untuk menghitung selisih waktu atau melakukan fingerprinting terhadap sistem target. Hasil pemindaian menunjukkan bahwa host merespons permintaan ICMP Timestamp, yang menunjukkan sistem target masih mengizinkan jenis komunikasi tersebut. Rekomendasi mitigasi meliputi penonaktifan dukungan ICMP Timestamp secara langsung di sistem target atau memblokirnya melalui firewall, terutama jika sistem berada di jaringan terbuka atau tidak dipercaya.

Pemilik sistem disarankan untuk segera memperbarui semua komponen dari Greenbone ke versi terbaru dan memperkuat konfigurasi firewall agar tidak membocorkan informasi sistem yang tidak perlu.

### B. Hasil Scan dengan Docker Scout

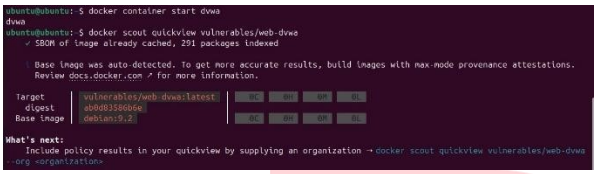

GAMBAR 7.
PEMINDAIAN IMAGE DOCKER

Gambar 7. diatas adalah tahap threat modeling dilakukan dengan menjalankan perintah docker scout quickview vulnerables/web-dvwa. Perintah ini memicu pemindaian cepat terhadap image dan menampilkan ringkasan keamanan, termasuk jumlah kerentanan yang terdeteksi (jika ada), serta saran pembaruan image.


GAMBAR 8. .
PEMINDAIAN IMAGE DOCKER SEBELUM RUN

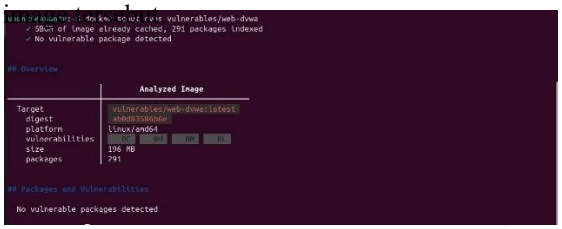
Pada gambar 8. image docker yang belum dijalankan dengan hasil pemindaian dengan Docker Scout terhadap image yang vulnerables/web-dvwa

mengindikasikan potensi kerentanan karena penggunaan base image lama (debian:9.2) dan keberadaan banyak paket lama. Meskipun informasi jumlah kerentanannya belum ditampilkan secara eksplisit, pengguna disarankan untuk melakukan audit lebih lanjut terhadap image ini dan segera memperbarui komponen-komponennya untuk menurunkan risiko eksploitasi. Proses ini dilakukan pada waktu 1 july 2025 sampai 8 juli 2025.




GAMBAR 9.
PEMINDAIAN IMAGE DOCKER SETELAH RUN

Pada gambar 9. image docker yang sudah dijalankan menunjukkan bahwa Docker image vulnerables/web-dvwa telah berhasil dipindai oleh Docker Scout dan mengandung 291 package yang siap dianalisis untuk kerentanan. Penggunaan base image debian:9.2 menandakan adanya potensi besar kerentanan karena sistem operasi tersebut sudah usang. Langkah mitigasi lanjutan sangat disarankan, termasuk mengganti base image dan memperbarui package dalam




GAMBAR 10.
PEMINDAIAN IMAGE DOCKER BERBASIS CVE SEBELUM RUN

Pada gambar 10. memindai image docker berbasis CVE sebelum dijalankan menunjukkan bahwa image vulnerables/web-dvwa telah dianalisis menggunakan Docker Scout dan tidak ditemukan kerentanan aktif berdasarkan database CVE yang tersedia. Meskipun image ini dikenal sebagai image pengujian yang rentan, dalam konteks CVE yang dianalisis oleh Docker Scout, seluruh paket dalam image dianggap aman secara struktural.


GAMBAR 11.
PEMINDAIAN IMAGE DOCKER BERBASIS CVE

Pada gambar 11. pemindaian image docker berbasis CVE setelah dijalankan menunjukan Image vulnerables/web-dvwa tidak memiliki kerentanan aktif pada tingkat paket. Meskipun image tersebut dikenal sebagai image yang digunakan untuk pengujian keamanan (vulnerable intentionally), dari sisi struktur paket bawaan, tidak ditemukan kelemahan langsung yang terdaftar secara publik.

C. Perbedaan Scan OpenVas dan Docker Scout

Setelah melakukan pemindaian kerentanan menggunakan OpenVAS dan Docker Scout, diperoleh perbedaan yang cukup signifikan dari kedua tools pada tabel dibawah ini:

TABEL 1.
PERBANDINGAN TOOLS

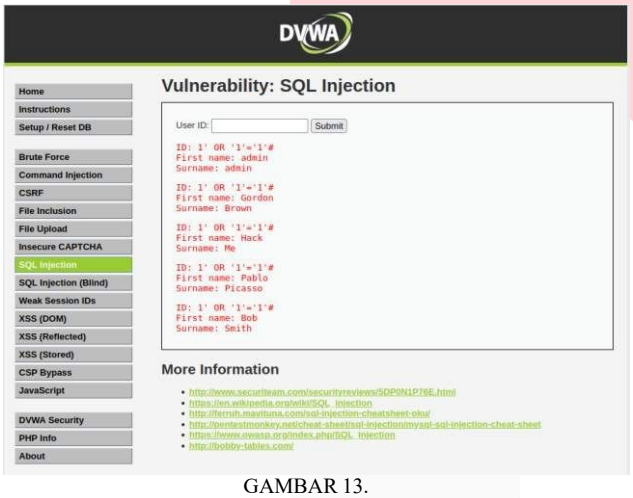
| Aspek | OpenVAS | Docker Scout |
|---|---|---|
| Fokus Pemindaian | Menyasar sistem, layanan, dan konfigurasi secara menyeluruh (*host-based scanning*) | Fokus pada image Docker dan dependensinya (*image-based scanning*) |
| Tipe Kerentanan yang Dideteksi | Kerentanan jaringan, service aktif, protokol, konfigurasi sistem | Kerentanan dalam layer image, package CVE, library yang outdated |
| Lingkungan Operasi | Diinstal di Kali Linux, digunakan dengan antarmuka web (*Greenbone Security Assistant*) | Diinstall di Ubuntu, CLI tools, terintegrasi dengan Docker Hub |
| Database CVE | *Greenbone Vulnerability Management Feed* (dengan update harian) | Database CVE Docker Hub, berbasis metadata image |
| Visualisasi Hasil | Laporan mendetail dalam antarmyka web (*status, sverity, Qod*) | Tampilan CLI dengan ringkasan CVE per level risiko |
| Rekomendasi Mitigasi | Tersedia secara manual dalam laporan (berdasarkan *severity* dan *Qod*) | Tersedia secara otomatis dan terintegrasi dengan saran updtae base image |
| Kelebihan | Deteksi menyeluruh pada Container, termasuk layanan aktif dan konfigurasi jaringan | Cepat dan ringan, cocok untuk CI/CD pipeine, fokus spesifik pada image docker |
| Kelemahan | Konfigurasi dan runtime yang kompleks, memerlukan waktu pemindaian yang lama | Kurang endeteksi ancaman pada level sistem atau layanan berjalan |
| Contoh Hasil | Menemukan 2 kerentanan : *outdated scan engine* (*severity 10.0*) dan ICMP *timestamp leak* (*secerity 2.1*) | Menampilkan kerentanan image seperti 1 CVE dengan level medium dan 24 CVE dengan level low saat membandingkan base image |
| Kesesuaian PTES | Cocok untuk fase *Intelligence Gathering, Thread Modeling,* dan *Vulnerability Analysis* | Cocok untuk tahap awal *Intelligence Gathering* berbasisi image |
| Waktu yang Didibutuhkan | Cukup lama, beberapa menit | Hanya memerlukan waktu sedikit, beberapa detik |

D. Exploitasi

Setelah melakukan Scan Vulnerability menggunakan OpenVAS dan Docker Scout. Tahap berikutnya melakukan *Exploitation*. *Exploitation* adalah fase di mana penyerang secara aktif mencoba mengeksploitasi kerentanan yang telah teridentifikasi untuk mendapatkan akses yang tidak sah atau informasi sensitif. Bisa dilihat pada gambar 12. Untuk menu SQL Injection pada DVWA.

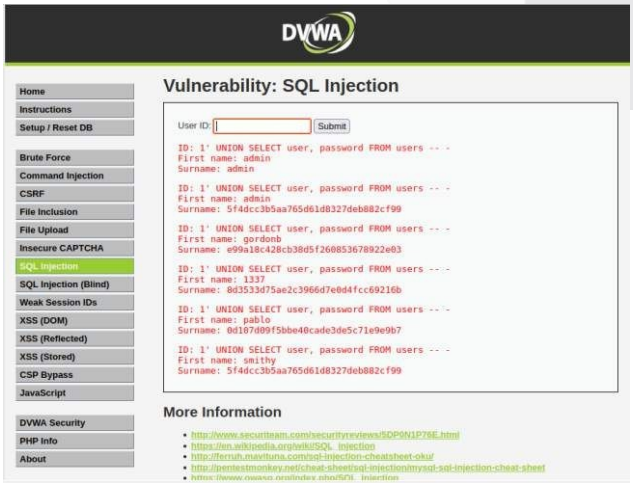

GAMBAR 12.
MENU SQL INJECTION

Pada gambar 13. dengan memanfaatkan payload seperti 1' OR '1'='1'#, attacker berhasil mem-bypass autentikasi dan menampilkan seluruh user yang tersimpan dalam database.




GAMBAR 13.
MENAMPILKAN SELURUH USER YANG TERSIMPAN

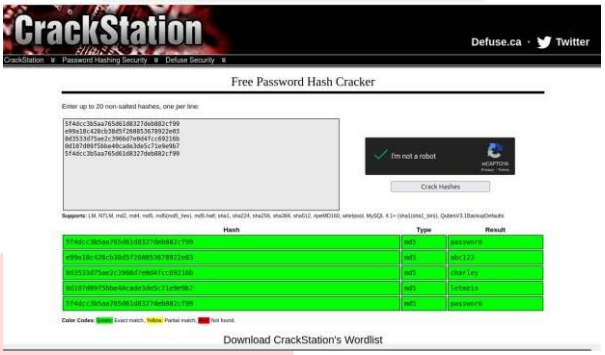
Pada gambar 14. menjelaskan attacker menggunakan payload lanjutan 1' UNION SELECT user, password FROM users -- - untuk memperoleh daftar user lengkap beserta hash password mereka. Ini menunjukkan bahwa celah keamanan pada DVWA dapat dengan mudah dieksploitasi dengan query sederhana yang melewati filter input.




GAMBAR 14.
MENAMPILKAN USERNAME DAN PASSWORD

E. Post Exploitation

Setelah berhasil melakukan serangan SQL Injection pada aplikasi DVWA, kamu mendapatkan hash password dari database. Tahap berikutnya adalah mencoba mendapatkan password asli dari hash yang diperoleh.




GAMBAR 15.
CRACKSTATION MEMECAHAN HASH PASSWORD

Gambar 15. menunjukkan hasil CrackStation yang menampilkan password plaintext, termasuk beberapa password sederhana seperti "password", "abc123", "charley", dan "letmein". Ini menunjukkan bahwa tahap Post-Exploitation berhasil dalam memperoleh informasi sensitif (password) dari database yang telah di-inject.

F. Reporting

Setelah attacker berhasil mendapatkan password plaintext, tahap selanjutnya adalah menyusun laporan yang mendokumentasikan seluruh langkah, metode, hasil, dan rekomendasi mitigasi. Laporan ini harus menyertakan bukti screenshot dari hasil CrackStation sebagai bukti bahwa hash password berhasil dipecahkan.

1. Risiko

Kerentanan SQL Injection ini memiliki risiko yang sangat signifikan bagi organisasi. Dengan memanfaatkan kelemahan validasi input pada aplikasi DVWA, attacker dapat memperoleh akses ilegal ke database, termasuk username dan password yang seharusnya bersifat rahasia. Jika hash password berhasil dipecahkan, attacker dapat login ke akun user maupun administrator tanpa otorisasi. Hal ini berpotensi menyebabkan kebocoran data sensitif, kerusakan reputasi organisasi, serta risiko kerugian finansial akibat pencurian atau penyalahgunaan data pengguna.

2. Mitigasi

Untuk mengatasi risiko ini, organisasi harus menerapkan langkah-langkah mitigasi yang tepat. Pertama, gunakan prepared statements (parameterized queries) agar input pengguna tidak langsung dieksekusi sebagai bagian dari query SQL, sehingga mencegah serangan SQL Injection. Kedua, lakukan validasi input secara ketat untuk memastikan bahwa data yang diterima sesuai dengan format yang diharapkan. Ketiga, gunakan algoritma hashing yang lebih kuat, seperti bcrypt atau Argon2, dan tambahkan salt untuk memperkuat hash password agar tidak mudah dipecahkan. Keempat, lakukan patching secara rutin untuk menutup kerentanan yang dapat dieksploitasi oleh attacker. Terakhir,

lakukan uji penetrasi secara berkala untuk mendeteksi kerentanan baru sebelum dieksploitasi pihak tidak bertanggung jawab.

## V. KESIMPULAN

Berdasarkan hasil pengujian terhadap container Docker yang menjalankan DVWA menggunakan OpenVAS dan Docker Scout, dapat disimpulkan bahwa kedua alat memiliki keunggulan masing-masing. OpenVAS lebih efektif dalam mendeteksi kerentanan saat container berjalan (runtime), sementara Docker Scout unggul dalam analisis kerentanan pada level image sebelum container dijalankan.

Eksploitasi terhadap aplikasi menunjukkan bahwa meskipun image dianggap aman oleh Docker Scout, celah seperti SQL Injection tetap dapat ditemukan dan dimanfaatkan. Serangan ini bahkan memungkinkan perolehan hash password yang berhasil dipecahkan, mengindikasikan ancaman serius terhadap keamanan data.

Disarankan agar organisasi melakukan pemindaian keamanan secara rutin menggunakan kedua alat tersebut untuk cakupan yang lebih menyeluruh. Selain itu, praktik pengamanan seperti validasi input, penggunaan algoritma hashing yang kuat, pembaruan image yang selektif, serta pengendalian akses jaringan perlu diterapkan untuk meningkatkan keamanan container secara keseluruhan.

## REFERENSI


[1] T. Astriani, A. Budiyono, and A. Widjajarto,“Analisa Kerentanan Pada Vulnerable Docker Menggunakan Scanner Openvas Dan Docker Scan Dengan Acuan Standar NIST 800-115,” vol. 8, no. 4, 2021, [Online]. Available: http://jurnal.mdp.ac.id

[2] Dwiyatno, E. Rakhmat, and O. Gustiawan, “IMPLEMENTASI VIRTUALISASI SERVER BERBASIS DOCKER CONTAINER,” vol. 7, no. 2, 2020.

[3] D. N. Cunong, M. Saputra, and W. Puspitasari, “ANALYSIS OF OROS MODELER DATA REPORTING PROCESS TO SAP HANA IN ACTIVITY BASED COSTING FOR INDONESIA TELECOMMUNICATION INDUSTRY,” vol. 7, no. 1, 2020.

[4] F. Hanifah, A. Budiyono, and A. Widjajarto, “ANALISA KERENTANAN PADA VULNERABLE DOCKER MENGGUNAKAN ALIENVAULT DAN DOCKER BENCH FOR SECURITY DENGAN ACUAN FRAMEWORK CIS CONTROL.”

[5] S. Utoro *et al.*, “Analisis Keamanan Website E-Learning SMKN 1 Cibatu Menggunakan Metode Penetration Testing Execution Standard.” M. Oktaviana, A. Widjajarto, and A. Almaarif, “Analisis Vulnerability Management Pada Container Docker Menggunakan Opensource Scanner Berdasarkan Standar Cyber Resilience Review (CRR),” *Jurnal Sistem Komputer dan Informatika (JSON)*, vol. 4, no. 1, p. 77, Sep. 2022, doi: 10.30865/json.v4i1.4787.

[6] B. Wiguna *et al.*, “Wiguna, Implementasi Web Application Firewall dalam Mencegah Serangan SQL Injection pada Website Implementasi Web Application Firewall Dalam Mencegah Serangan SQL Injection Pada Website”, doi: 10.31849/digitalzone.v11i2.4867ICCS.