LAPORAN UJIAN AKHIR SEMESTER
IMPLEMENTASI DEVSECOPS PADA APLIKASI OWASP JUICE SHOP

Mata Kuliah:
Development Security Operation

Oleh:
Jovita Nadya Artista 88032023017

PROGRAM STUDI REKAYASA KEAMANAN SIBER
POLITEKNIK BHAKTI SEMESTA
SALATIGA
2026

I. PENDAHULUAN

Perkembangan aplikasi berbasis web yang semakin pesat juga diiringi dengan meningkatnya risiko keamanan siber. Banyak aplikasi yang memiliki celah keamanan akibat kurangnya pengujian dan pengamanan sejak tahap awal pengembangan. Oleh karena itu, penerapan pendekatan DevSecOps menjadi sangat penting agar aspek keamanan dapat terintegrasi ke dalam seluruh siklus pengembangan aplikasi.

Pada laporan ini, digunakan aplikasi OWASP Juice Shop sebagai studi kasus karena aplikasi ini memang dirancang secara sengaja memiliki berbagai celah keamanan. Aplikasi tersebut sangat cocok digunakan untuk pembelajaran dan simulasi pengujian keamanan, mulai dari Static Application Security Testing (SAST), Software Composition Analysis (SCA), Dynamic Application Security Testing (DAST), hingga Container Security dan Monitoring & Incident Handling.

Melalui penerapan pipeline DevSecOps, pengujian keamanan dilakukan secara bertahap dan terstruktur untuk mengidentifikasi potensi kerentanan, menganalisis dampaknya, serta memahami bagaimana kerentanan tersebut dapat dimitigasi.

II. TUJUAN

Adapun tujuan dari penyusunan laporan dan pelaksanaan proyek ini adalah sebagai berikut:

a. Menerapkan konsep DevSecOps pada aplikasi web menggunakan OWASP Juice Shop.
b. Mengidentifikasi kerentanan keamanan aplikasi melalui:
    - SAST (npm audit)
    - SCA & Container Security (Trivy)
    - DAST (OWASP ZAP)
c. Menganalisis dampak dari kerentanan yang ditemukan terhadap keamanan sistem.
d. Menerapkan praktik container hardening untuk meningkatkan keamanan image Docker.
e. Mengimplementasikan monitoring dan observability untuk mendeteksi error dan aktivitas mencurigakan.
f. Mensimulasikan proses incident handling berdasarkan log dan temuan keamanan.
g. Memberikan pemahaman praktis mengenai pentingnya keamanan aplikasi sejak tahap pengembangan.

III. ARSITEKTUR

Arsitektur sistem pada proyek ini menggunakan pendekatan container-based dan DevSecOps pipeline. Aplikasi OWASP Juice Shop dijalankan di dalam container Docker dan diintegrasikan dengan berbagai tools keamanan untuk melakukan pengujian dan monitoring.

3.1 Komponen Arsitektur

a. OWASP Juice Shop

   Aplikasi web berbasis Node.js yang menjadi objek pengujian keamanan.

b. Docker

   Digunakan untuk menjalankan aplikasi dalam bentuk container, termasuk versi insecure dan hardened.

c. SAST Tool (npm audit)

   Digunakan untuk mendeteksi kerentanan pada dependency JavaScript secara statis.

d. SCA / Container Security (Trivy)

   Digunakan untuk melakukan scanning terhadap image Docker dan dependency yang digunakan aplikasi.

e. DAST Tool (OWASP ZAP)

   Digunakan untuk melakukan pengujian keamanan secara dinamis dengan mensimulasikan serangan ke aplikasi yang sedang berjalan.

f. Monitoring & Logging

   Logging aplikasi dilakukan menggunakan log bawaan container Docker dan dianalisis untuk mendeteksi error serta aktivitas mencurigakan.

g. Incident Handling

   Proses penanganan insiden dilakukan berdasarkan log error dan hasil monitoring untuk mensimulasikan tahapan respons insiden.

3.2 Alur Arsitektur Sistem

a. Aplikasi OWASP Juice Shop dijalankan di dalam container Docker.

b. Kode sumber aplikasi dianalisis menggunakan SAST untuk mendeteksi kerentanan statis.

c. Image Docker discan menggunakan Trivy untuk mengidentifikasi kerentanan pada dependency dan sistem.

d. Aplikasi yang berjalan diuji menggunakan OWASP ZAP untuk menemukan celah keamanan secara dinamis.

e. Log aplikasi dikumpulkan dan dianalisis untuk mendeteksi error dan indikasi serangan.

f. Hasil temuan digunakan sebagai dasar untuk proses incident handling dan evaluasi keamanan sistem.

IV. TAHAP 1 — SETUP APLIKASI & REPOSITORY

Tahap ini bertujuan untuk menyiapkan aplikasi OWASP Juice Shop, menghubungkannya dengan repository GitHub pribadi, serta memastikan aplikasi berhasil dijalankan menggunakan Docker.

4.1 Clone OWASP Juice Shop

Pada langkah ini, repository resmi OWASP Juice Shop diambil dari GitHub ke komputer lokal menggunakan perintah git clone.

```
git clone https://github.com/juice-shop/juice-shop.git
cd juice-shop
```

Penjelasan:

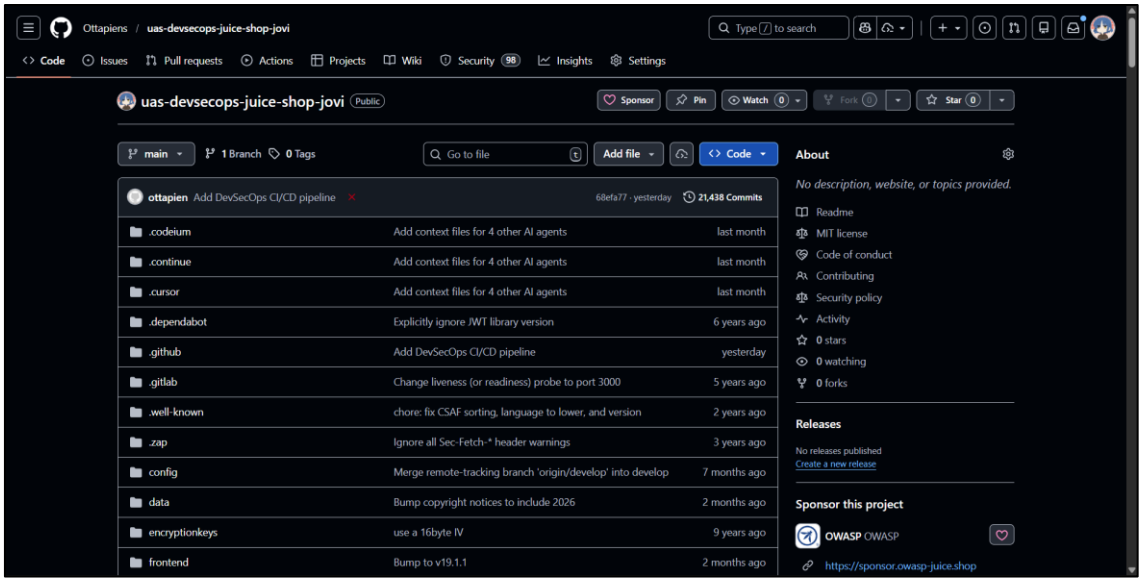
- git clone digunakan untuk menyalin source code OWASP Juice Shop dari GitHub.
- cd juice-shop digunakan untuk masuk ke folder project agar dapat melakukan konfigurasi dan pengembangan lebih lanjut.

Hasil dari langkah ini adalah source code OWASP Juice Shop tersedia di lingkungan lokal.

4.2 Membuat Repository GitHub Pribadi

Setelah source code tersedia di lokal, dibuat repository GitHub pribadi dengan nama uas-devsecops-juice-shop-jovi yang digunakan untuk menyimpan seluruh hasil pengerjaan proyek. Selanjutnya repository lokal dihubungkan ke GitHub menggunakan perintah berikut:

```
git init
git remote add origin https://github.com/Ottapiens/uas-devsecops-juice-shop-jovi.git
git branch -M main
git push -u origin main
```



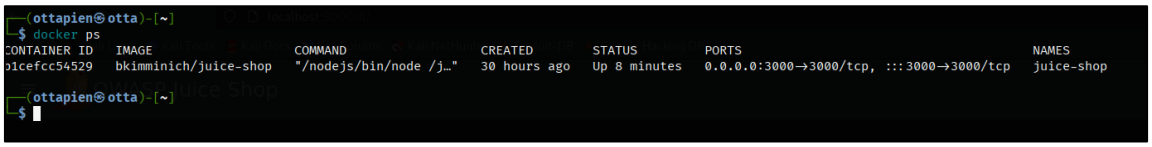
4.3 Menjalankan OWASP Juice Shop Menggunakan Docker

Aplikasi OWASP Juice Shop dijalankan menggunakan Docker agar lebih mudah dan konsisten. Menggunakan perintah berikut:

```
docker run -d -p 3000:3000 --name juice-shop bkimminich/juice-shop
```

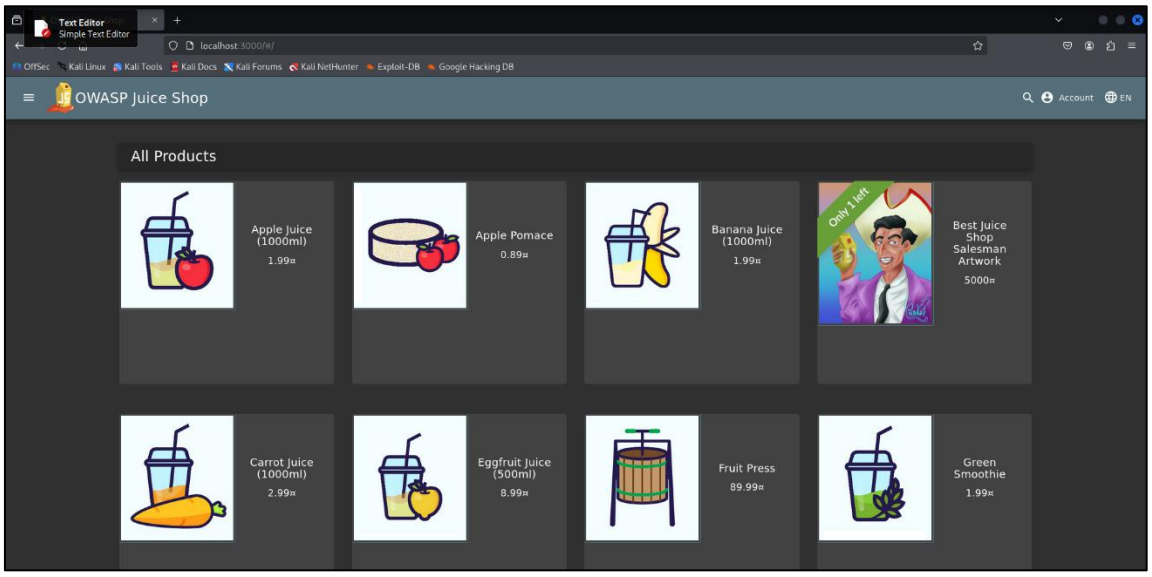
4.4 Verifikasi Container Berjalan

Untuk memastikan container berjalan dengan baik, digunakan perintah:



4.5 Akses Aplikasi melalui Browser

Aplikasi OWASP Juice Shop dapat diakses melalui browser dengan alamat http://localhost:3000



V. TAHAP 2 — SECURE SDLC & THREAT MODELING

5.1 Diagram Arsitektur Juice Shop

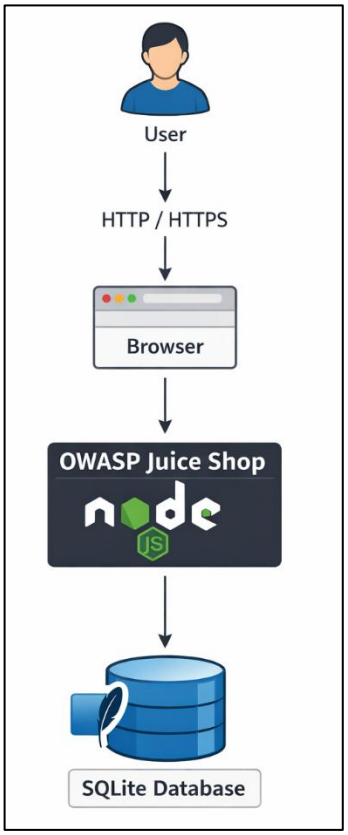


Diagram arsitektur ini menggambarkan alur kerja sistem OWASP Juice Shop secara sederhana, mulai dari pengguna hingga penyimpanan data. User merupakan pihak yang menggunakan aplikasi OWASP Juice Shop.

User dapat berupa pengunjung atau pengguna terdaftar yang melakukan berbagai aktivitas seperti melihat produk, melakukan login, atau mengirimkan data melalui aplikasi.

Browser berfungsi sebagai media penghubung antara user dan aplikasi OWASP Juice Shop. User mengakses aplikasi menggunakan browser seperti Google Chrome atau Mozilla Firefox. Komunikasi antara browser dan aplikasi dilakukan menggunakan protokol HTTP atau HTTPS, di mana browser mengirimkan request dan menerima response dari server.

OWASP Juice Shop adalah aplikasi web yang berjalan menggunakan Node.js. Komponen ini berperan sebagai server aplikasi (backend sekaligus frontend) yang:

- Menerima request dari browser
- Memproses logika aplikasi
- Mengelola autentikasi dan otorisasi pengguna
- Mengakses dan mengelola data di database

SQLite Database digunakan sebagai media penyimpanan data aplikasi. Database ini menyimpan berbagai informasi penting seperti data user, produk, transaksi, dan data lainnya. OWASP Juice Shop berkomunikasi langsung dengan database untuk melakukan proses Create, Read, Update, dan Delete (CRUD) data.

5.2 Identifikasi Aset Penting (Minimal 5)

Identifikasi aset penting dilakukan untuk menentukan komponen sistem yang memiliki nilai tinggi dan perlu dilindungi karena berpotensi menjadi target serangan.

No.	Aset	Deskripsi
1.	User Account	Data akun pengguna
2.	Authentication	Login & session
3.	Database	Menyimpan data user & produk
4.	API Endpoint	Endpoint backend
5.	Session Token	Autentikasi user

5.3 Threat Modeling — STRIDE

Metode STRIDE digunakan untuk mengidentifikasi jenis ancaman keamanan yang mungkin terjadi pada setiap aset sistem.

Aset	S	T	R	I	D	E	Contoh Ancaman
User Account	✓		✓	✓		✓	Account takeover
API Endpoint		✓		✓	✓	✓	API abuse
Database		✓		✓	✓	✓	SQL Injection

a. User Account – Account Takeover

Ancaman ini terjadi ketika penyerang berhasil mengambil alih akun pengguna dengan cara mencuri kredensial, melakukan brute force, atau memanfaatkan kelemahan autentikasi. Dampaknya adalah akses tidak sah terhadap data dan fitur aplikasi.

- Spoofing: Penyerang menyamar sebagai user sah
- Repudiation: Aktivitas tidak dapat dilacak ke pelaku sebenarnya
- Information Disclosure: Data pribadi pengguna bocor
- Elevation of Privilege: Penyerang mendapatkan hak akses lebih tinggi

b. API Endpoint – API Abuse

API endpoint yang tidak memiliki validasi dan kontrol akses yang baik dapat dieksploitasi oleh penyerang untuk mengirim request berbahaya secara berulang atau mengakses data yang tidak seharusnya.

- Tampering: Manipulasi parameter request
- Information Disclosure: Kebocoran data melalui API
- Denial of Service: API dibanjiri request berlebih
- Elevation of Privilege: Akses fungsi admin tanpa izin

c. Database – SQL Injection

SQL Injection terjadi ketika input dari user tidak divalidasi dengan baik dan langsung digunakan dalam query database. Penyerang dapat membaca, mengubah, atau menghapus data di database.

- Tampering: Manipulasi data di database
- Information Disclosure: Kebocoran data sensitif
- Denial of Service: Kerusakan database menyebabkan layanan terganggu
- Elevation of Privilege: Akses database tanpa hak

5.4 Risk Analysis — DREAD

Metode DREAD digunakan untuk mengukur tingkat risiko setiap ancaman berdasarkan lima faktor penilaian.

Threat	Damage	Repro	Exploit	Affected	Discover	Total
SQL Injection	9	8	9	9	8	43
XSS	7	8	7	8	8	38
Brute Force	6	9	8	7	8	38

a. SQL Injection (Total: 43 – Risiko Tinggi)

SQL Injection memiliki tingkat risiko paling tinggi karena dapat menyebabkan kebocoran data besar, manipulasi database, hingga penghapusan data penting. Serangan ini relatif mudah direproduksi dan berdampak pada seluruh pengguna sistem.

b. Cross-Site Scripting (XSS) (Total: 38 – Risiko Sedang–Tinggi)

XSS memungkinkan penyerang menyisipkan skrip berbahaya ke aplikasi yang kemudian dijalankan di browser pengguna lain. Dampaknya meliputi pencurian session token dan manipulasi tampilan halaman.

c. Brute Force (Total: 38 – Risiko Sedang–Tinggi)

Serangan brute force dilakukan dengan mencoba berbagai kombinasi username dan password secara berulang. Jika tidak ada pembatasan login, serangan ini mudah dilakukan dan dapat menyebabkan pengambilalihan akun pengguna.

VI. TAHAP 3 — CI/CD PIPELINE (DEVSECOPS)

Pada repository proyek ini terdapat beberapa workflow GitHub Actions yang berjalan secara bersamaan. Setiap workflow didefinisikan dalam file konfigurasi berekstensi .yml yang berada di dalam direktori .github/workflows. GitHub Actions secara otomatis memperlakukan setiap file workflow tersebut sebagai satu pipeline yang berdiri sendiri. Adapun workflow yang terdapat pada repository ini antara lain:

- CodeQL Scan
- Let me lint:fix that for you
- CI/CD Pipeline
- DevSecOps Pipeline

Keberadaan beberapa pipeline ini disebabkan oleh penggunaan template bawaan GitHub serta penambahan pipeline secara bertahap selama proses pengembangan.

Beberapa pipeline berhasil dijalankan tanpa kendala, yaitu CodeQL Scan dan Let me lint:fix that for you. Pipeline tersebut menggunakan konfigurasi standar GitHub Actions dan tidak memerlukan proses build atau deployment aplikasi secara penuh. Pipeline CodeQL berfungsi untuk melakukan analisis kode statis (Static Application Security Testing) untuk mendeteksi potensi kerentanan pada source code. Sementara itu, pipeline linting bertujuan untuk memastikan konsistensi gaya penulisan kode dan melakukan perbaikan otomatis terhadap kesalahan awal.

Pipeline CI/CD Pipeline dan DevSecOps Pipeline mengalami kegagalan pada beberapa tahap eksekusi. Kegagalan tersebut tidak disebabkan oleh kesalahan konfigurasi, melainkan oleh beberapa faktor teknis.

Pertama, terdapat ketidaksesuaian konfigurasi environment khususnya versi Node.js yang digunakan pada beberapa pipeline lama, sehingga tidak sepenuhnya kompatibel dengan versi terbaru OWASP Juice Shop. Kondisi ini menyebabkan proses instalasi dan build aplikasi tidak berjalan dengan sempurna.

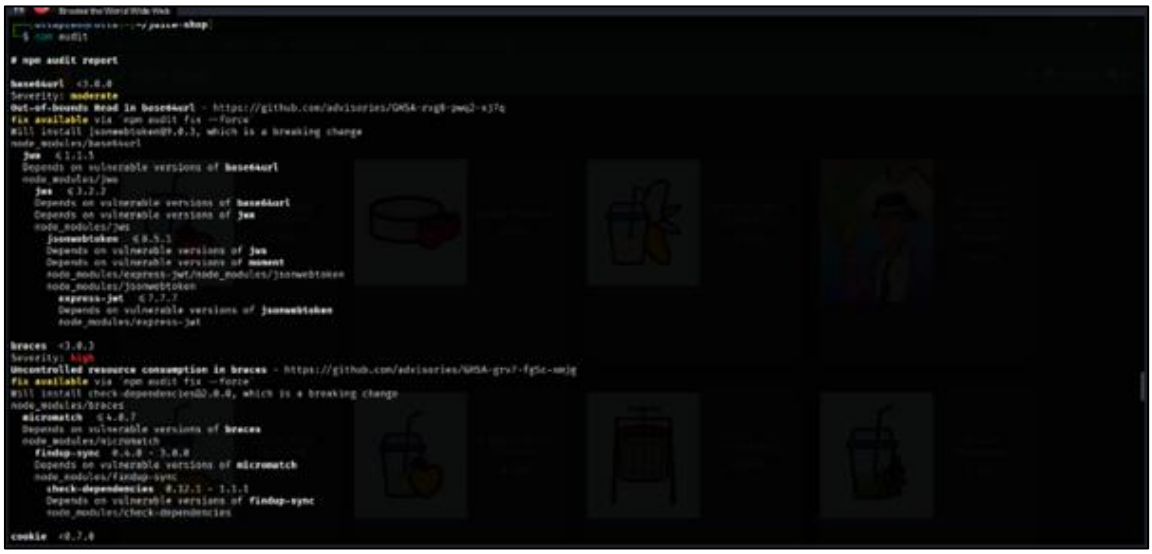
Kedua, aplikasi OWASP Juice Shop merupakan aplikasi yang secara sengaja dirancang memiliki kerentanan keamanan. Oleh karena itu, pada tahap pengujian keamanan seperti vulnerability scanning, beberapa tools mendeteksi temuan berisiko tinggi yang dapat menyebabkan pipeline ditandai sebagai gagal.

Ketiga, pipeline keamanan menerapkan aturan yang cukup ketat. Tools seperti Trivy dan OWASP ZAP dapat menghentikan pipeline ketika menemukan celah keamanan tertentu.

VII. TAHAP 4 — SECURITY TESTING OTOMATIS

7.1 SAST – npm audit

Static Application Security Testing (SAST) dilakukan untuk mengidentifikasi kerentanan keamanan pada kode sumber dan dependency aplikasi tanpa menjalankan aplikasi secara langsung. Pada tahap ini, tools yang digunakan adalah npm audit yang berfungsi untuk memeriksa library pihak ketiga (dependency) terhadap database kerentanan keamanan publik (CVE).



Hasil pemindaian menunjukkan bahwa aplikasi masih memiliki beberapa celah keamanan, terutama dari library pihak ketiga yang digunakan.

Severity	Jumlah
Critical	7
High	17
Moderate	18
Low	1
Total	**43**

Dari seluruh hasil scan, beberapa vulnerability dengan tingkat Critical dipilih untuk dibahas karena memiliki dampak paling berbahaya.

a. vm2 – Critical

Sandbox Escape

Library vm2 digunakan untuk menjalankan kode JavaScript dalam sandbox. Namun, kerentanan ini memungkinkan penyerang keluar dari sandbox dan menjalankan kode di sistem host. Risikonya yaitu penyerang dapat menjalankan kode berbahaya di server.

Dampak:

- Remote Code Execution (RCE), yang berpotensi menyebabkan:
- Pengambilalihan server
- Akses tidak sah ke sistem operasi
- Manipulasi data aplikasi

b. crypto-js - Critical

Cryptographic Function

Kerentanan pada crypto-js disebabkan oleh penggunaan algoritma kriptografi yang sudah dianggap lemah dan tidak lagi direkomendasikan untuk melindungi data sensitif. Risikonya yaitu mekanisme hashing atau enkripsi dapat dengan mudah dipecahkan oleh penyerang.

Dampak:

- Kebocoran data sensitif
- Password compromise
- Pelanggaran kerahasiaan informasi pengguna

c. Lodash – Critical

Prototype Pollution & Command Injection

Library lodash memiliki celah yang memungkinkan manipulasi prototype objek JavaScript, yang dapat dimanfaatkan untuk mengubah perilaku aplikasi secara tidak sah. Risikonya yaitu penyerang penyerang bisa mengubah cara kerja aplikasi..

Dampak:

- Eksekusi perintah berbahaya
- Pengambilalihan logika aplikasi
- Potensi Remote Code Execution

d. Marsdb – Critical

Kerentanan ini memungkinkan penyerang menyisipkan perintah sistem melalui input yang tidak tervalidasi dengan baik. Risikonya yaitu penyerang dapat menjalankan perintah berbahaya di server.

Dampak:

- Server dapat sepenuhnya dikuasai
- Data bisa dihapus atau dicuri
- Sistem menjadi tidak aman

7.2 SCA & Container Scan – Trivy

Software Composition Analysis (SCA) adalah proses untuk menganalisis library dan dependency yang digunakan oleh aplikasi. Pada tahap ini digunakan Trivy untuk memindai image Docker OWASP Juice Shop dan mencari celah keamanan yang berasal dari:

- Library Node.js
- Dependency pihak ketiga
- Komponen di dalam container

Berdasarkan hasil pemindaian menggunakan Trivy, ditemukan beberapa vulnerability dengan tingkat keparahan tinggi.

Severity	Jumlah
Critical	8
High	23
Total	**31**

Dari seluruh hasil scan, beberapa vulnerability dengan tingkat Critical dipilih untuk dianalisis karena memiliki dampak paling berbahaya terhadap sistem.

a. vm2 - Critical

Sandbox Escape

Library vm2 digunakan untuk menjalankan kode secara terisolasi (sandbox). Namun, celah keamanan ini memungkinkan kode berbahaya keluar dari sandbox dan mengakses sistem utama. Risikonya yaitu penyerang dapat menjalankan kode berbahaya langsung di server.

Dampak:

- Remote Code Execution (RCE)
- Server dapat dikendalikan oleh penyerang
- Aplikasi kehilangan kontrol keamanan

b. crypto-js - Critical

Weak Cryptographic Function

Library crypto-js menggunakan algoritma enkripsi atau hashing yang sudah tidak aman dan terlalu lemah untuk standar keamanan saat ini. Risikonya yaitu password dan data yang dienkripsi bisa dengan mudah dipecahkan.

Dampak:

- Password pengguna dapat dicuri
- Data sensitif bocor
- Keamanan sistem secara keseluruhan menurun

c. lodash – Critical

Prototype Pollution

Celah ini memungkinkan penyerang memanipulasi struktur objek JavaScript di dalam aplikasi. Risikonya yaitu penyerang dapat mengubah perilaku aplikasi tanpa izin.

Dampak:

- Aplikasi berjalan tidak sesuai fungsinya
- Data bisa dimanipulasi
- Potensi pengambilalihan aplikasi

d. marsdb – Critical

Command Injection

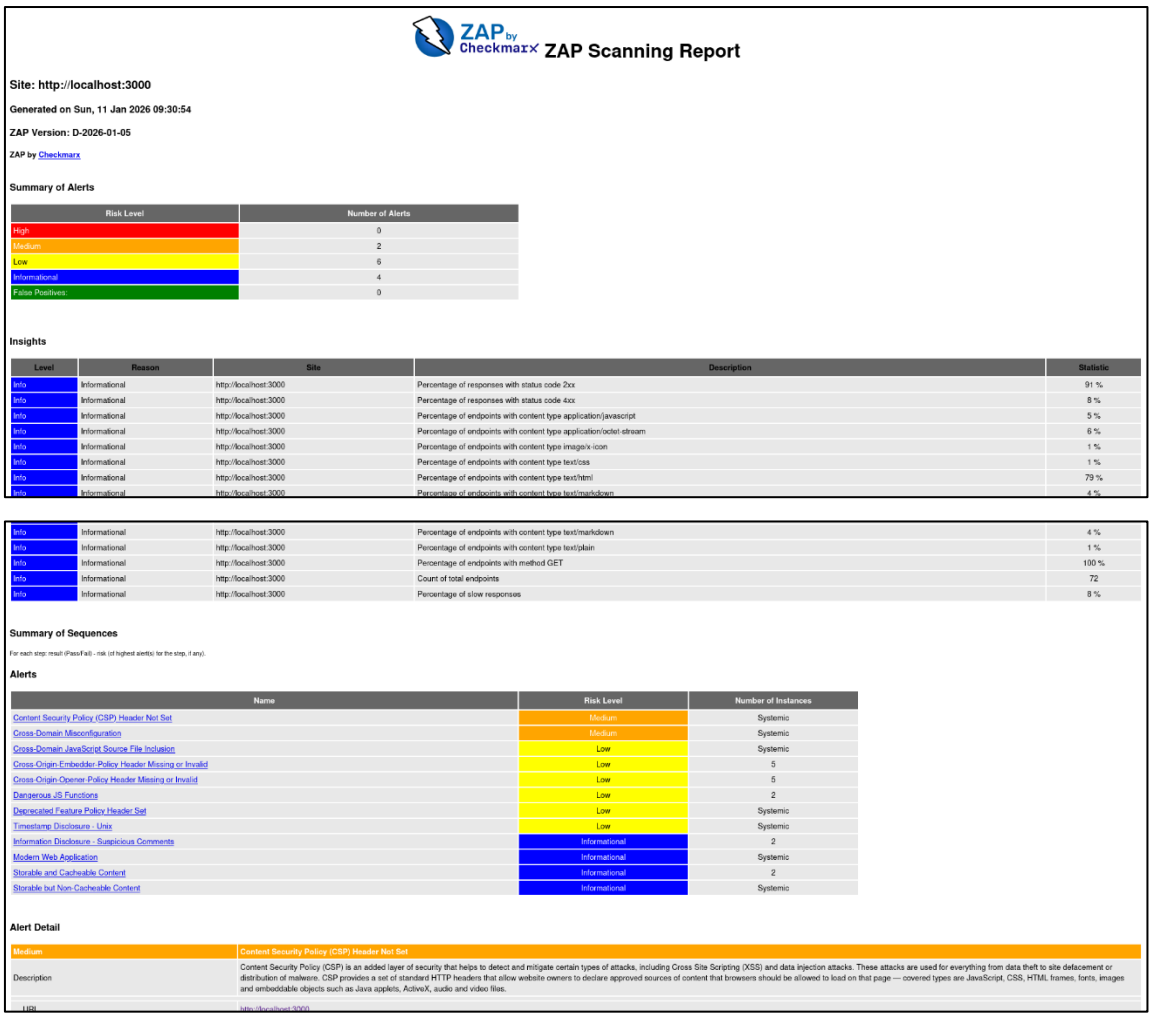
Library marsdb memungkinkan input berbahaya dijalankan sebagai perintah sistem. Risikonya yaitu penyerang bisa menjalankan perintah berbahaya di server.

Dampak:

- Server dapat dikuasai sepenuhnya
- Data bisa dihapus atau dicuri
- Sistem menjadi tidak aman

7.3 DAST – OWASP ZAP

Berdasarkan hasil pemindaian DAST menggunakan OWASP ZAP terhadap aplikasi OWASP Juice Shop, ditemukan beberapa kerentanan dengan tingkat High dan Critical. Temuan ini menunjukkan bahwa aplikasi memiliki celah keamanan yang dapat dieksploitasi secara langsung saat aplikasi sedang berjalan.



ZAP by Checkmarx ZAP Scanning Report

Site: http://localhost:3000

Generated on Sun, 11 Jan 2026 09:30:54

ZAP Version: D-2026-01-05

ZAP by Checkmarx

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	2
Low	6
Informational	4
False Positives:	0

Insights

Level	Reason	Site	Description	Statistic
Info	Informational	http://localhost:3000	Percentage of responses with status code 2xx	91 %
Info	Informational	http://localhost:3000	Percentage of responses with status code 4xx	8 %
Info	Informational	http://localhost:3000	Percentage of endpoints with content type application/javascript	5 %
Info	Informational	http://localhost:3000	Percentage of endpoints with content type application/octet-stream	6 %
Info	Informational	http://localhost:3000	Percentage of endpoints with content type image/x-icon	1 %
Info	Informational	http://localhost:3000	Percentage of endpoints with content type text/css	1 %
Info	Informational	http://localhost:3000	Percentage of endpoints with content type text/html	79 %
Info	Informational	http://localhost:3000	Percentage of endpoints with content type text/markdown	4 %

Info	Informational	http://localhost:3000	Percentage of endpoints with content type text/markdown	4 %
Info	Informational	http://localhost:3000	Percentage of endpoints with content type text/plain	1 %
Info	Informational	http://localhost:3000	Percentage of endpoints with method GET	100 %
Info	Informational	http://localhost:3000	Count of total endpoints	72
Info	Informational	http://localhost:3000	Percentage of slow responses	8 %

Summary of Sequences

For each step: result (Pass/Fail) - risk (of highest alert(s) for the step, if any).

Alerts

Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	Systemic
Cross-Domain Misconfiguration	Medium	Systemic
Cross-Domain JavaScript Source File Inclusion	Low	Systemic
Cross-Origin-Embedder-Policy Header Missing or Invalid	Low	5
Cross-Origin-Opener-Policy Header Missing or Invalid	Low	5
Dangerous JS Functions	Low	2
Deprecated Feature Policy Header Set	Low	Systemic
Timestamp Disclosure - Unix	Low	Systemic
Information Disclosure - Suspicious Comments	Informational	2
Modern Web Application	Informational	Systemic
Storable and Cacheable Content	Informational	2
Storable but Non-Cacheable Content	Informational	Systemic

Alert Detail

Medium	Content Security Policy (CSP) Header Not Set
Description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
URL	http://localhost:3000

a. Cross-Site Scripting (XSS) — High

Aplikasi menerima input dari pengguna tanpa validasi yang aman, sehingga penyerang bisa menyisipkan script berbahaya (JavaScript). Risikonya yaitu penyerang dapat mencuri cookie atau session pengguna dan akun pengguna bisa diambil alih.

Dampak:

- Pencurian data pengguna
- Session hijacking
- Kerusakan kepercayaan pengguna terhadap aplikasi

b. SQL Injection — High / Critical

   Aplikasi tidak memfilter input dengan baik saat berkomunikasi dengan database, sehingga penyerang bisa menjalankan query SQL berbahaya.Risikonya yaitu data database dapat dibaca, diubah, atau dihapus dan bypass login tanpa password.

   Dampak:

   - Kebocoran data sensitif
   - Kerusakan atau penghapusan database
   - Akses ilegal ke sistem backend

c. Broken Authentication / Session Management — High

   Pengelolaan login dan session pengguna tidak aman. Risikonya yaitu session bisa ditebak atau dicuri dan login bisa dilewati tanpa autentikasi yang valid.

   Dampak:

   - Pengambilalihan akun pengguna
   - Akses ilegal ke fitur admin

d. Sensitive Data Exposure — High

   Data sensitif seperti token, cookie, atau informasi penting dikirim atau disimpan tanpa perlindungan yang cukup. Risikonya yaitu data bisa disadap oleh pihak tidak berwenang.

   Dampak:

   - Kebocoran data pribadi
   - Pelanggaran privasi pengguna

e. Security Misconfiguration — High

   Konfigurasi server dan aplikasi masih menggunakan pengaturan default atau tidak aman. Risikonya yaitu informasi sistem mudah diketahui penyerang dan endpoint sensitif terbuka.

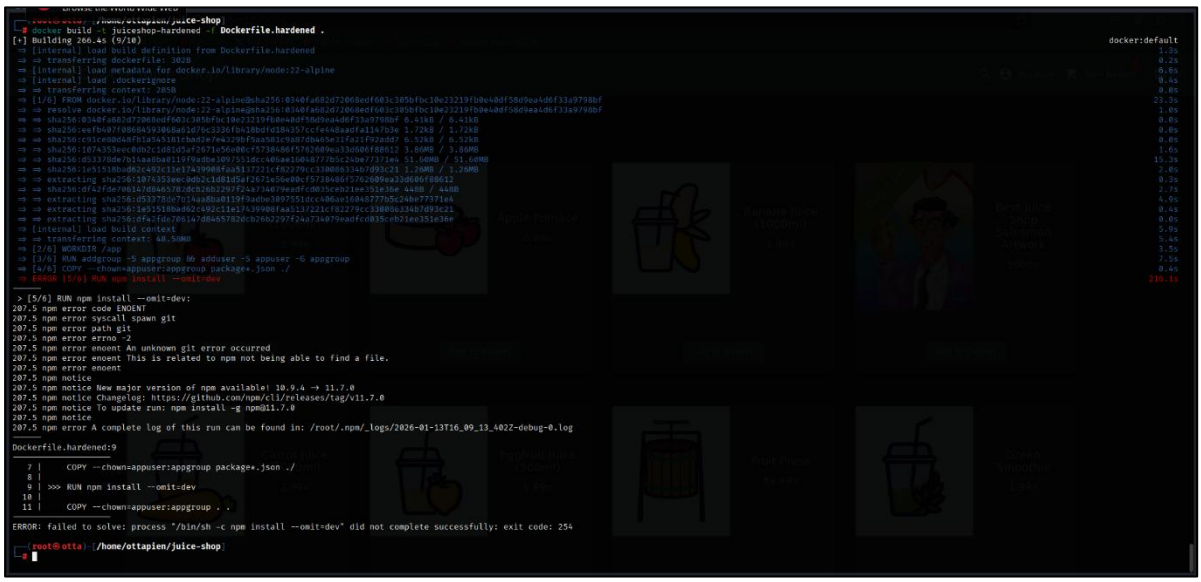
   Dampak:

   - Mempermudah eksploitasi celah keamanan lain
   - Meningkatkan risiko serangan lanjutan

VIII. TAHAP 5 — DOCKER SECURITY

Pada tahap Docker Security, dibuat dua jenis Dockerfile, yaitu Dockerfile insecure dan Dockerfile hardened. Dockerfile insecure menggambarkan kondisi container yang belum aman. Pada versi ini, aplikasi dijalankan dengan konfigurasi standar tanpa pengamanan tambahan, seperti masih menggunakan user root dan belum membatasi akses atau resource. Konfigurasi ini berisiko karena jika terjadi serangan, penyerang bisa memiliki akses yang lebih luas ke dalam container.

Selanjutnya, Dockerfile hardened dibuat sebagai versi yang lebih aman dari Dockerfile insecure. Pada versi ini, dilakukan beberapa perbaikan keamanan seperti menjalankan aplikasi menggunakan user non-root, menggunakan base image yang lebih ringan, serta mengurangi komponen yang tidak diperlukan. Tujuan dari Dockerfile hardened adalah untuk mengurangi risiko keamanan dan membatasi dampak jika terjadi serangan pada aplikasi.



Dalam proses pengerjaan, kedua Dockerfile tersebut mengalami kegagalan saat proses build. Hal ini terjadi karena aplikasi OWASP Juice Shop memiliki proses build yang cukup kompleks dan membutuhkan banyak dependensi, terutama pada bagian frontend. Akibatnya, image tidak berhasil dibuat secara lokal menggunakan Dockerfile yang disusun.

IX. TAHAP 6 — MONITORING & OBSERVABILITY

Pada tahap Monitoring & Observability, direncanakan penggunaan metrics menggunakan Prometheus dan Grafana untuk memantau performa aplikasi seperti CPU usage, memory usage, dan request rate.

Namun, pada implementasi ini metrics belum berhasil diaktifkan, dengan alasan sebagai berikut:

- Aplikasi OWASP Juice Shop tidak menyediakan endpoint metrics secara default
- Tidak terdapat integrasi langsung dengan Prometheus
- Grafana tidak menerima data metrics karena tidak ada exporter yang berjalan

Oleh karena itu, monitoring pada tahap ini fokus pada logging aplikasi sebagai bentuk observability dasar.

9.1 Implementasi Monitoring

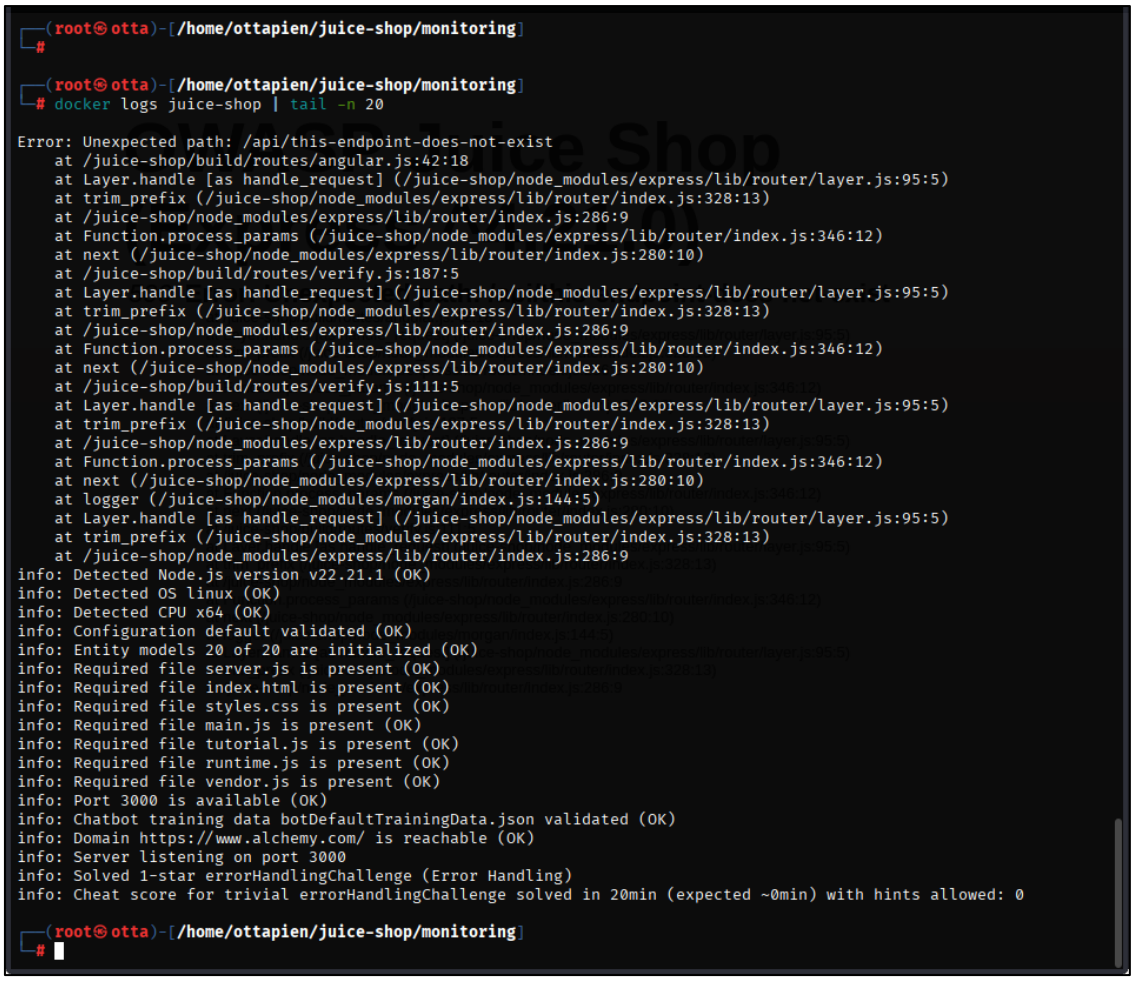
Aplikasi OWASP Juice Shop secara default telah memiliki mekanisme logging berbasis console log yang dapat diakses melalui Docker.

Log aplikasi diambil menggunakan perintah:

```
docker logs juice-shop
```

9.2 Skenario Pengujian

Logging ini digunakan untuk mencatat error aplikasi, mendeteksi request yang tidak valid, dan membantu analisis insiden. Untuk menguji kemampuan logging, dilakukan simulasi error aplikasi dengan cara mengakses endpoint API yang tidak tersedia pada aplikasi dengan /api/this-endpoint-does-not-exist. Aksi ini bertujuan untuk memicu error pada aplikasi sehingga dapat diamati apakah error tersebut tercatat pada log.



Setelah skenario dijalankan, dilakukan pengecekan log aplikasi dan dari hasil log ditemukan pesan error Error: Unexpected path: /api/this-endpoint-does-not-exist.

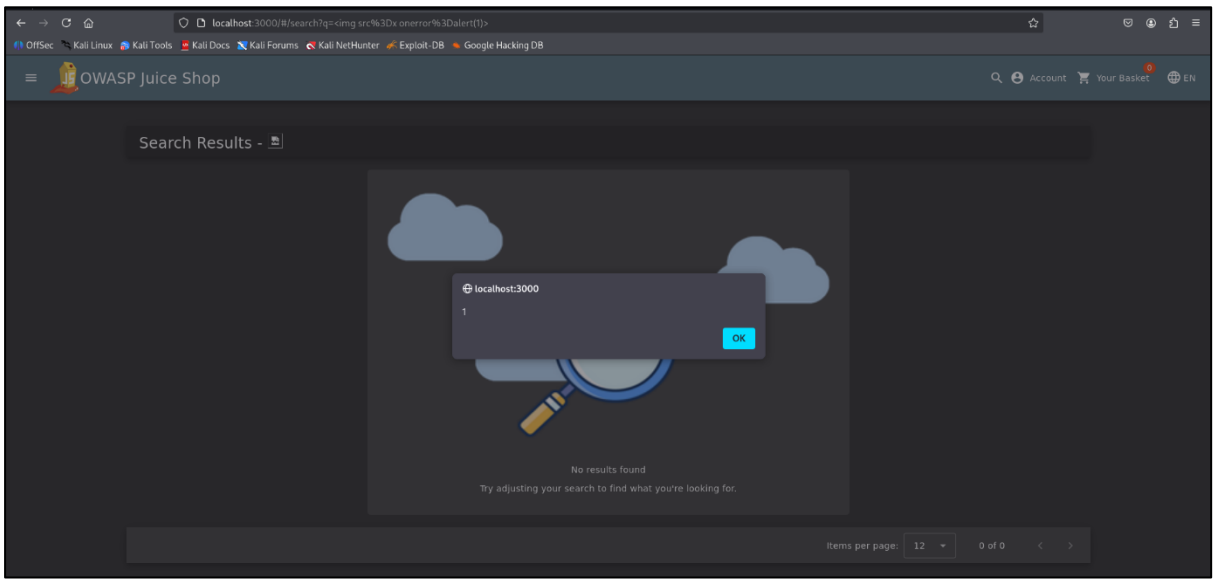
Berdasarkan hasil pengujian, dapat disimpulkan bahwa:

- Aplikasi berhasil mendeteksi akses ke endpoint yang tidak valid
- Error aplikasi tercatat secara otomatis di dalam log
- Informasi error yang ditampilkan cukup lengkap untuk keperluan analisis dan debugging

Mekanisme logging berjalan dengan baik dan dapat digunakan untuk monitoring dasar Logging ini sangat membantu dalam mengidentifikasi kesalahan aplikasi, menganalisis potensi serangan, dan memproses incident handling di tahap selanjutnya.

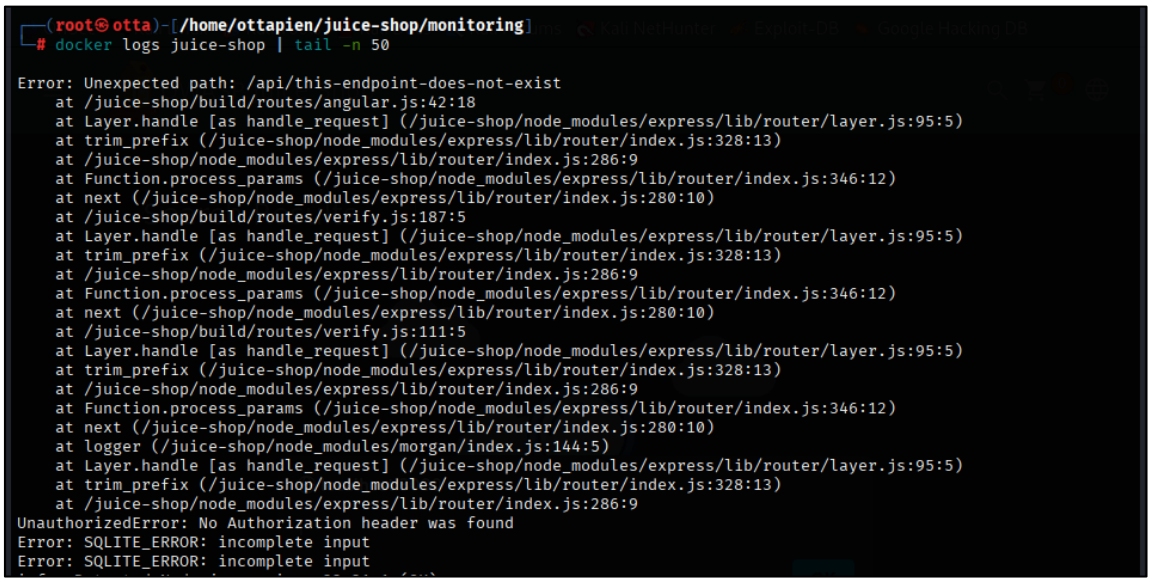
X. TAHAP 7 — INCIDENT HANDLING

Pada tahap ini dilakukan simulasi serangan SQL Injection terhadap aplikasi OWASP Juice Shop. Serangan dilakukan dengan memasukkan input tidak valid pada parameter URL dan endpoint API yang berhubungan dengan proses pencarian dan autentikasi. Aplikasi berhasil mendeteksi aktivitas tersebut melalui log backend, yang menunjukkan adanya kesalahan query pada database SQLite.



10.1 Identification (Identifikasi Insiden)

Insiden teridentifikasi melalui log aplikasi Docker dengan munculnya pesan error berikut:



Pesan error ini menandakan bahwa query SQL yang dijalankan oleh aplikasi menjadi tidak lengkap atau rusak akibat input yang tidak sesuai format. Kondisi ini merupakan ciri umum dari percobaan SQL Injection, di mana penyerang mencoba memanipulasi perintah SQL melalui input pengguna.

10.2 Containment (Penanganan Sementara)

Pada tahap ini, aplikasi berhasil menahan dampak serangan secara otomatis karena query SQL yang tidak valid tidak berhasil dieksekusi oleh sistem. Database menolak query tersebut dan hanya menampilkan pesan error, sehingga serangan tidak berkembang menjadi eksploitasi penuh. Tidak terjadi kebocoran maupun perubahan data, serta tidak ditemukan adanya akses tidak sah ke dalam sistem. Query berbahaya gagal dijalankan, aplikasi tetap beroperasi secara normal, dan tidak diperlukan tindakan pemutusan layanan karena stabilitas serta keamanan sistem tetap terjaga.

10.3 Eradication (Penghapusan Akar Masalah)

Untuk mencegah serangan serupa di masa depan, langkah perbaikan yang direkomendasikan adalah:

- Menggunakan parameterized query / prepared statement
- Melakukan validasi dan sanitasi input pengguna
- Menghindari penggunaan query SQL yang dibangun langsung dari input user
- Menyembunyikan detail error database dari tampilan pengguna

Langkah ini bertujuan untuk menghilangkan celah yang dapat dimanfaatkan oleh penyerang.

10.4 Recovery (Pemulihan Sistem)

Setelah insiden terdeteksi, aplikasi tetap berjalan dengan normal tanpa mengalami gangguan operasional. Tidak diperlukan tindakan pemulihan seperti restart container maupun proses restore database karena tidak ada dampak terhadap sistem. Kondisi layanan tetap stabil dan dapat digunakan sebagaimana mestinya, sementara proses monitoring terus dilakukan untuk memastikan tidak adanya percobaan serangan lanjutan serta menjaga keamanan sistem secara berkelanjutan.

10.5 Lessons Learned (Pembelajaran dari Insiden)

Dari insiden ini dapat diambil beberapa pembelajaran penting:

- Input pengguna tanpa validasi dapat menyebabkan risiko SQL Injection
- Log backend sangat penting untuk mendeteksi serangan eksploitasi
- Error database yang terlalu detail dapat membantu penyerang
- Monitoring aplikasi harus dilakukan secara terus-menerus
- Pendekatan DevSecOps membantu mendeteksi insiden lebih awal

Insiden ini menunjukkan pentingnya penerapan keamanan sejak tahap pengembangan aplikasi.

XI. KESIMPULAN

Berdasarkan praktik yang telah dilakukan, dapat disimpulkan bahwa penerapan konsep DevSecOps pada aplikasi OWASP Juice Shop berhasil memberikan gambaran menyeluruh mengenai bagaimana keamanan dapat diintegrasikan ke dalam seluruh siklus pengembangan aplikasi. Mulai dari tahap pengembangan, pengujian, deployment, hingga monitoring, keamanan tidak lagi ditempatkan di akhir proses, tetapi menjadi bagian yang berkelanjutan.

Secara keseluruhan, praktik ini membuktikan bahwa DevSecOps mampu meningkatkan kesadaran keamanan, membantu mendeteksi potensi kerentanan sejak dini, serta memberikan pendekatan yang lebih sistematis dalam menjaga keamanan aplikasi. Penerapan DevSecOps menjadi langkah penting untuk membangun aplikasi yang tidak hanya fungsional, tetapi juga aman.