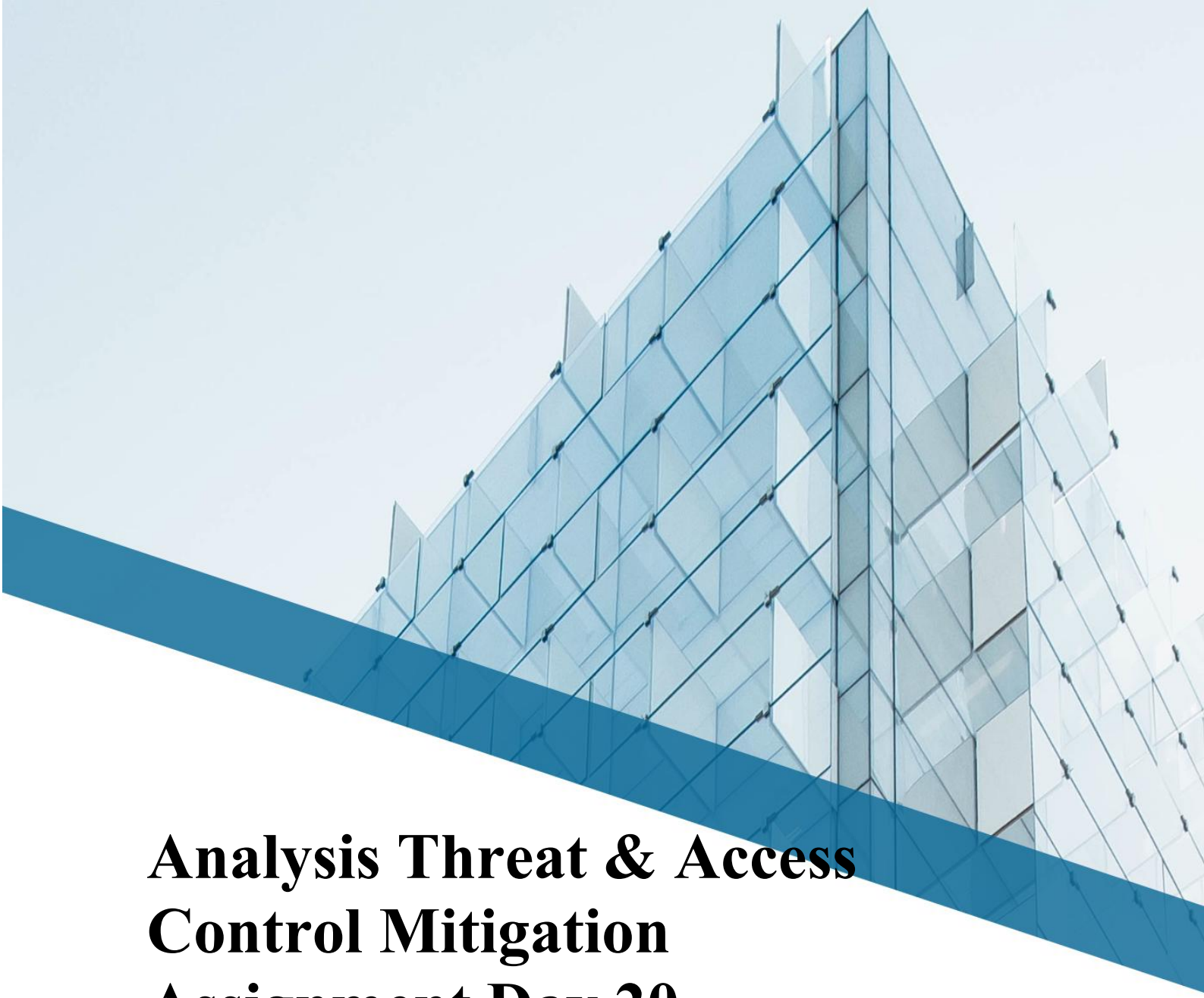


Analysis Threat & Access Control Mitigation Assignment Day 20

Bootcamp Cyber Security Batch 4

Disusun oleh: Sabilillah Ramaniya Widodo

!!Catatan: Diharapkan seluruh pengerjaan Assignment tidak sepenuhnya mengandalkan penggunaan AI!!

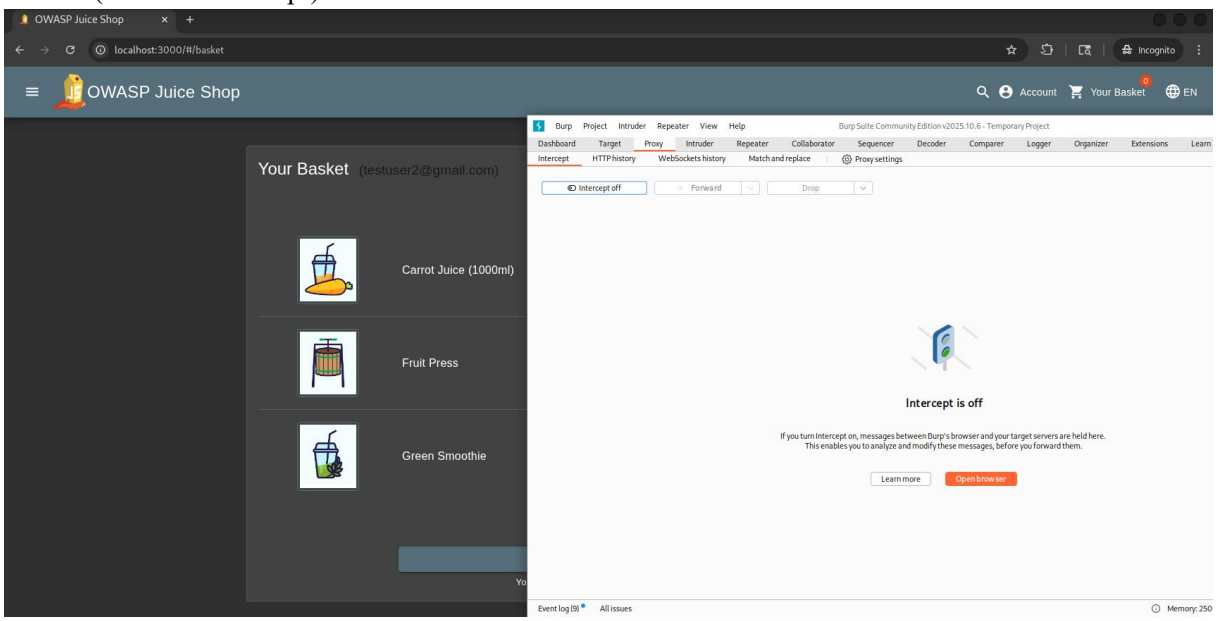
“Proses belajar ibarat menanam pohon. Jika hanya mengandalkan AI tanpa memahami esensinya, yang berkembang bukan kompetensimu, melainkan ketergantungan yang melemahkan.”
- Learning Design Dibimbing

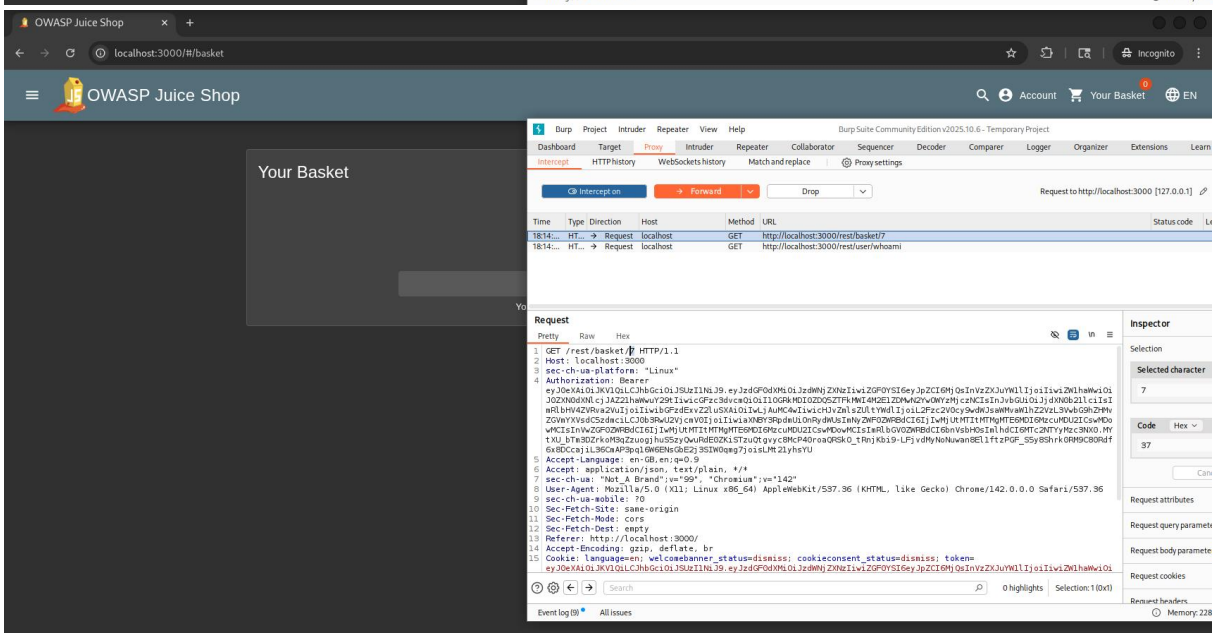
Temuan #1: Insecure Direct Object References (IDOR) pada Basket

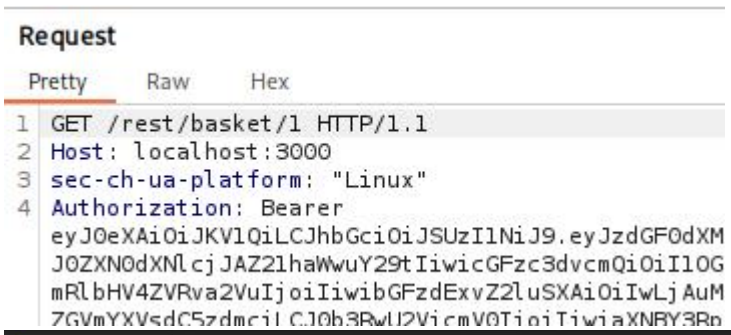
Kategori risiko: HIGH

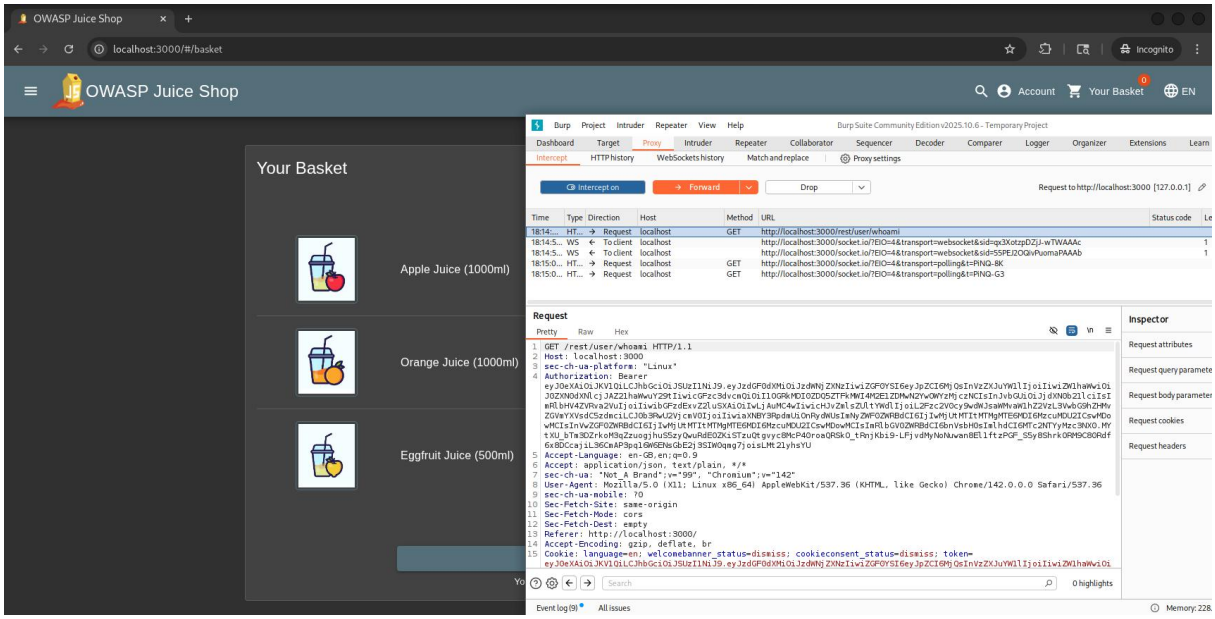
- Deskripsi: ditemukan IDOR vulnerability pada endpoint API REST untuk basket, dimana aplikasi tidak memvalidasi dengan benar apakah user yang sedang login punya otorisasi untuk mengakses ID keranjang (basketId) yang direquest. User bisa saja memanipulasi id parameter dalam URL request untuk lihat isis keranjang milik pengguna lain.

- Bukti (Proof-of-Concept):









Attacker bisa mengubah angka 7 menjadi id lain (misalnya 1, 2, 3, …) untuk mengakses data basket user lain tanpa validasi kepemilikan

- Dampak

1. Melanggar privasi: attacker bisa melihat item apa saja yang dibeli oleh user lain
2. Manipulasi data: kalau method HTTP diubah jadi POST di endpoint yang sama, attacker mungkin saja bisa memanipulasi isi basket user lain

- Rekomendasi mitigasi:

1. Server-side validation: mitigasi bisa dilakukan dengan menerapkan pemeriksaan access control yang ketat di setiap API request. Pastikan session atau token user yang login cocok dengan user_id pemilik basket (keranjang) yang diminta
2. Menggunakan indirect reference: hal ini bisa dilakukan dengan menghindari penggunaan id database secara sequence (seperti 1, 2, 3, dst.) secara langsung di url. Bisa gunakan randomized token atau uuid yang sulit ditebak
3. Contoh pseudocode:

```
FUNCTION getBasketDetails(requested_basket_id, current_user_token):
      user_id = decodeToken(current_user_token)
      basket = database.query("SELECT * FROM baskets WHERE id = ?",
      requested_basket_id)
IF basket.owner_id != user_id:
      LogSecurityEvent("IDOR Attempt detected by user " + user_id)
RETURN HTTP_403_FORBIDDEN("Anda tidak memiliki akses ke
keranjang ini.")
```

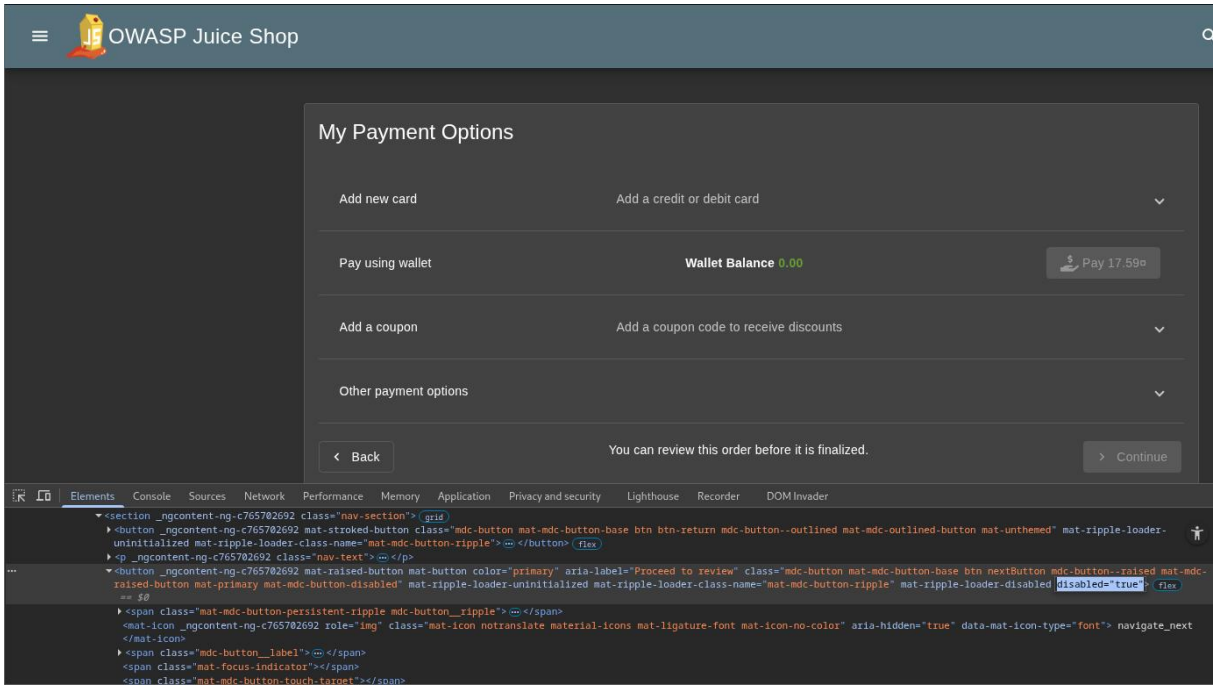
RETURN HTTP_200_OK(basket)

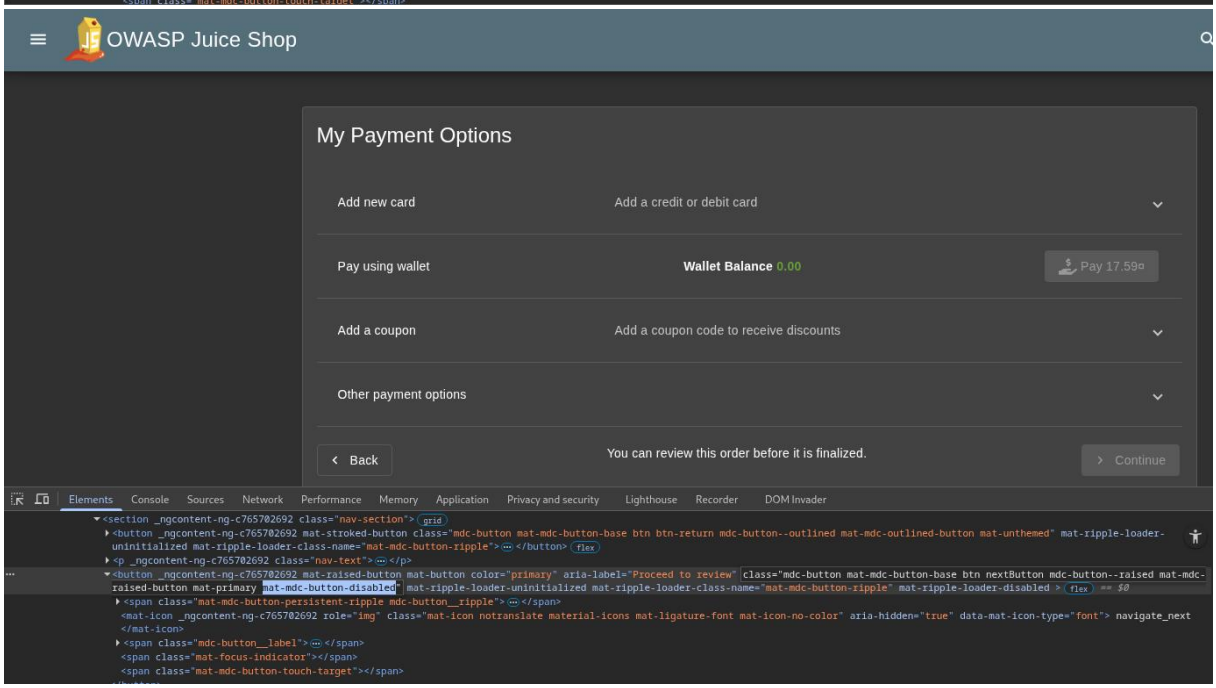
Temuan #2: Client-side Access Control Bypass di Opsi Pembayaran

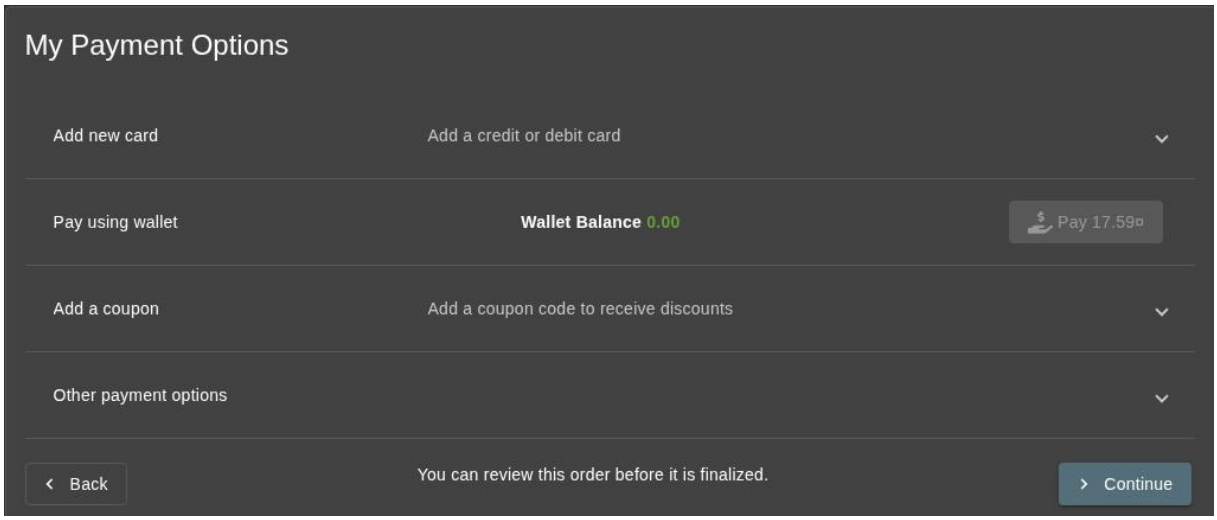
Kategori risiko: Medium

- Deskripsi: Aplikasi dibuat dengan access control yang tidak aman dengan hanya mengandalkan validasi di client-side (browser) untuk menonaktifkan tombol pembayaran. Tombol “Continue” didisable secara visual ketika saldo 0 (tidak mencukupi). Tapi mekanisme ini bisa dibypass dengan memodif elemen HTML lewat Developer Tools browser

- Bukti (Proof-of-Concept)







Di awal, tombol “Continue” punya atribut disabled. Attacker bisa menginspect elemen dan menghapus atribut itu secara manual. Tombol kemudian menjadi aktif dan bisa diklik meski saldo tidak mencukupi.

- Dampak

1. Kerugian bisnis: user mungkin bisa memproses pesanan tanpa pembayaran yang sah kalau server tidak verifikasi ulang saldo sebelum memproses transaksi
2. Data Inconsistency: bisa menyebabkan error di logika pemrosesan pesanan di backend

- Rekomendasi Mitigasi

1. Server-side validation: server harus selalu memeriksa apakah saldo pengguna mencukupi saat request pembayaran diterima, terlepas dari status tombol di UI
2. Logic Integrity: memastikan logika bisnis yang critical dieksekusi di backend, bukan frontend
3. Contoh pseudocode:

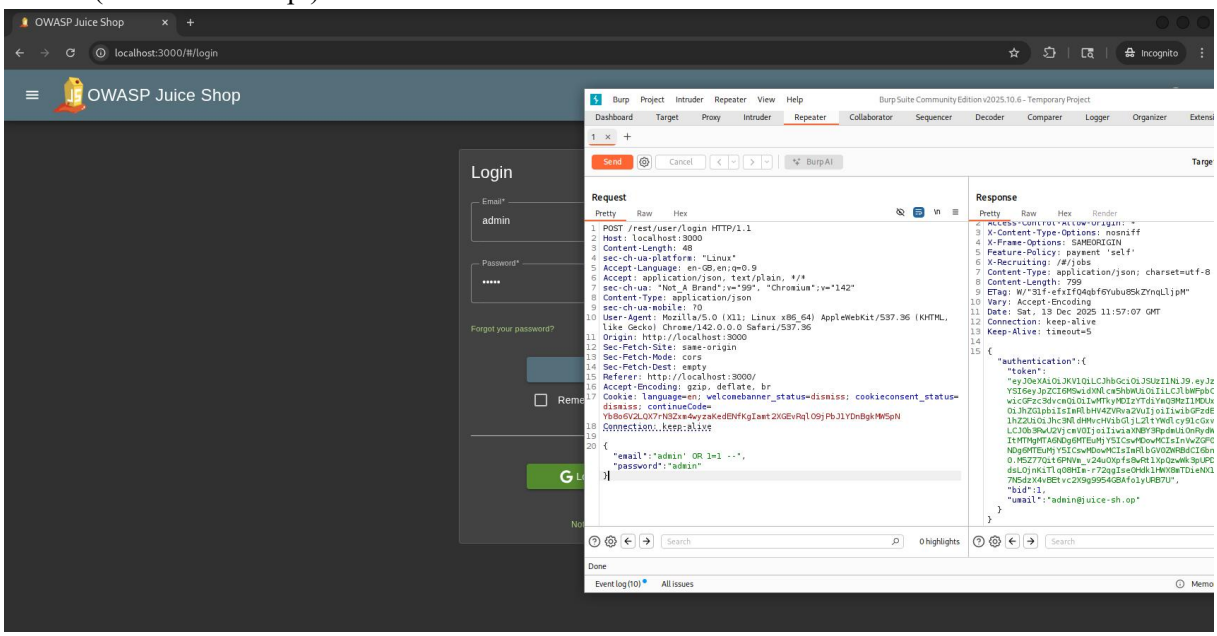
```
FUNCTION processPayment(user_id, order_total):
      current_balance = database.query(“SELECT balance FROM wallets WHERE user_id = ?”,     user_id)

      IF current_balance < order_total:
            RETURN HTTP_400_BAD_REQUEST(“Saldo tidak mencukupi. Transaksi dibatalkan”)

      new_balance = current_balance - order_total
      database.execute(UPDATE wallets SET balance = ? WHERE user_id = ?, new_balance,           user_id)

      RETURN HTTP_200_OK(“Pembayaran berhasil”)
```

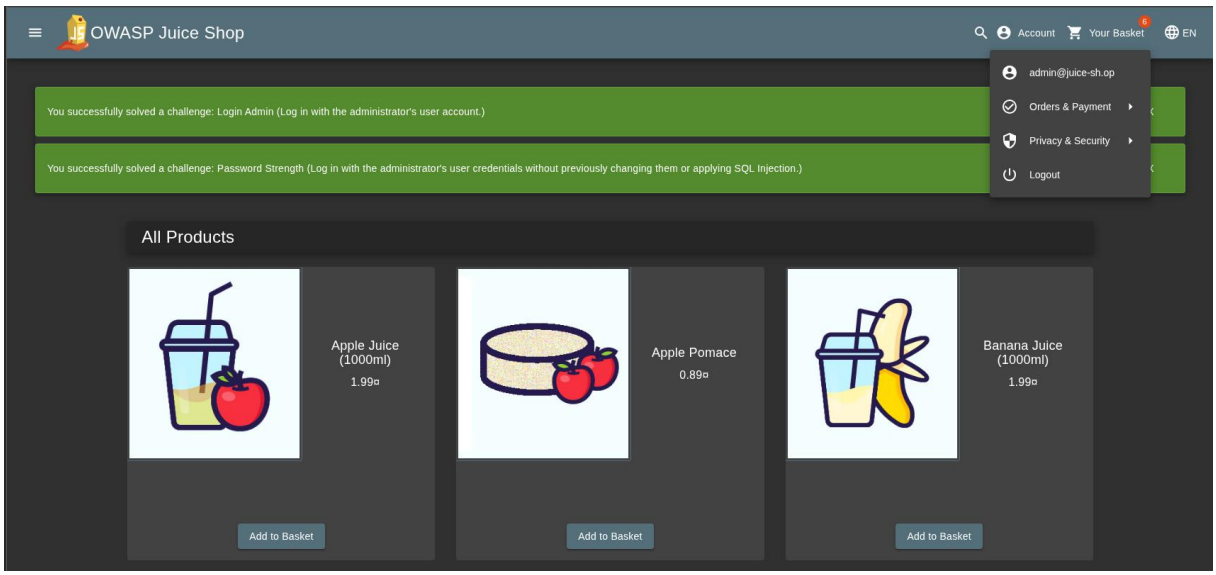
Temuan #3: Vertical Privilage Escalation (Admin Access) via Injeksi SQL

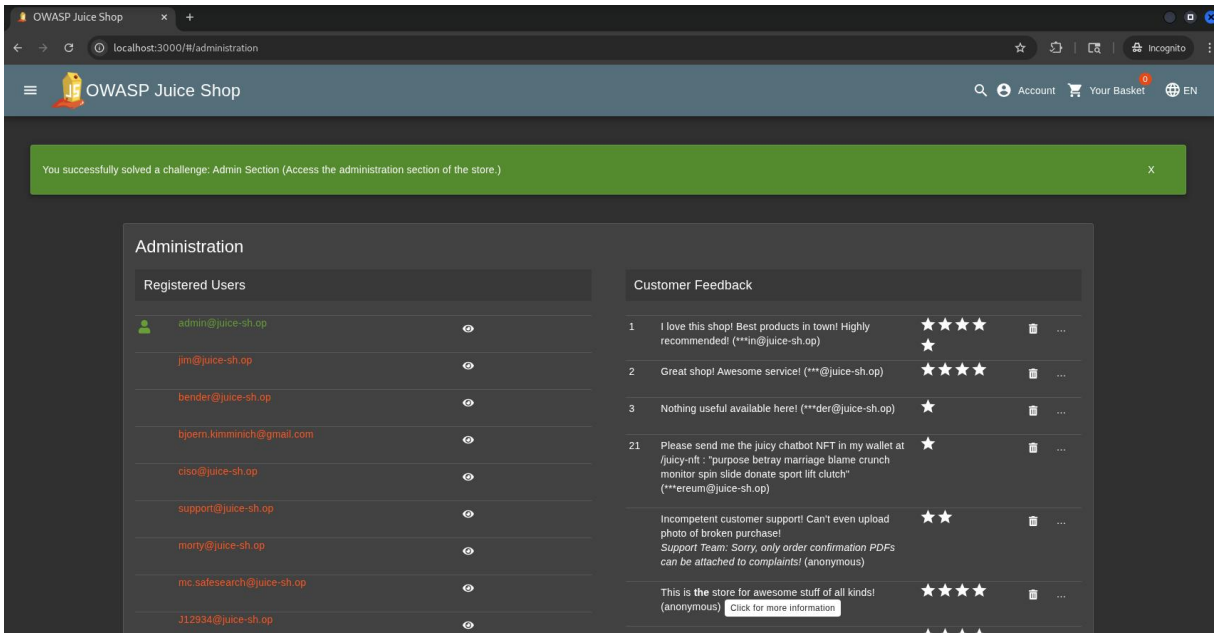
Kategori risiko: Critical

- Deskripsi: Kerentanan fatal ini ditemukan saat attacker bisa melewati mekanisme autentikasi dan mendapat akses admin menggunakan teknik SQL injection. Celah ini memungkinkan untuk mengakses data tanpa izin dan privilege escalation secara penuh

- Bukti (Proof-of-Concept):







Di page login, attacker bisa memasukkan payload SQLi di kolom email: ‘ OR 1=1 --. Payload ini memanipulasi query database untuk mereturn nilai true pada kondisi login, yang secara otomatis login sebagai pengguna pertama di database (admin). Hasilnya, attacker bisa akses halaman /administration yang seharusnya dibatasi hanya untuk role admin.

- Dampak

1. Pengambilalihan sistem penuh: attacke sekarang punya kontrol penuh atas aplikasi, termasuk kemampuan melihat semua user yang terdaftar, melihat feedback, dan data sensitif lainnya
2. Reputation & Policy: pelanggaran data skala besar yang bisa merusak kepercayaan user dan melanggar regulasi perlindungan data

- Rekomendasi Mitigasi

1. Parameterized queries: menggunakan prepared statements untuk semua interaksi database yang memisahkan kode SQL dari data input user, sehingga mencegah injeksi
2. Role-based Access Control: memastikan middleware otorisasi untuk memverifikasi role user secara ketat di setiap endpoint (dalam hal ini /administration)
3. Contoh pseudocode:

```
FUNCTION loginUser(email_input, password_input):
      Sql_query = “SELECT * FROM users WHERE email = ? AND password = ?”

      User = database.execute(sql_query, [email_input, password_input])

      IF user exists:
            generateSessionToken(user)
            RETURN HTTP_200_OK(“Login sukses”)
      ELSE:
            RETURN HTTP_401_UNAUTHORIZED(“Email atau password salah”)
```

Diagram (Security by Design)