Journal of Applied Informatics and Computing (JAIC)
Vol.x, No.x, Juli 201x, pp. xx~xx
e-ISSN: 2548-6861

Analisis Kerentanan dan *Hardening* Keamanan pada *Endpoint* API Transaksi DWANPOS Menggunakan OWASP API Security Top 10:2023

Adam Rizky Nugroho [1]*, Nur Cahyono Kushardianto [2]**
*Rekayasa Keamanan Siber, Jurusan Teknik Informatika, Politeknik Negeri Batam
**Teknik Komputer, Jurusan Teknik Informatika, Politeknik Negeri Batam
ad.adam992@gmail.com [1], anung@polibatam.ac.id [2]

Article Info

Article history:

Received …
Revised …
Accepted …

Keyword:

API Security, BOLA, Multi-Tenant, SaaS, Web Security.

ABSTRACT

The increasing adoption of Software as a Service (SaaS) applications introduces security challenges in cross-tenant data isolation and access control. This study analyzes and hardens transaction API endpoints in DWANPOS, a multi-tenant Point of Sale (POS) SaaS platform, together with the shared JWT authentication middleware that protects them. The research employs white-box security testing based on NIST SP 800-115 and the OWASP Web Security Testing Guide (WSTG v4.2). Vulnerabilities are classified using Common Weakness Enumeration (CWE) and mapped to OWASP API Security Top 10:2023 and OWASP Top 10:2021. Static analysis using Gosec and Semgrep with a custom rule is combined with manual code review, while dynamic testing is performed manually using Postman, following a test matrix of six test cases and two authenticated tenant accounts. The assessment identifies a Critical authentication bypass (CVSS v3.1 9.8, CWE-287/CWE-347) caused by hardcoded demo-token logic and two High-severity Broken Object Level Authorization (BOLA) vulnerabilities (CVSS v3.1 7.6 and 7.5, CWE-639 and CWE-862) affecting transaction status modification and history retrieval. Gosec did not detect either class of flaw; they were identified through a custom Semgrep rule (BOLA) and manual code review. Mitigation controls include removing the demo-token bypass, enforcing HMAC-SHA256 signature verification, and binding authenticated user identifiers to GORM queries. Retesting with the same scenarios, together with positive-control tests, shows that cross-tenant access is rejected while legitimate owners retain access to their own data. The findings highlight the importance of combining static analysis with manual business-logic review and positive-control testing to strengthen API authorization.

This is an *open* access article under the CC–BY-SA license.

I. PENDAHULUAN

Transformasi digital mendorong penggunaan aplikasi berbasis web dalam berbagai aktivitas bisnis, termasuk pengelolaan transaksi dan operasional usaha melalui sistem *Point of Sale* (POS). Salah satu bentuk penerapannya adalah POS berbasis *Software as a Service* (SaaS), yaitu aplikasi yang menyediakan layanan secara terpusat dan dapat digunakan oleh beberapa pengguna melalui jaringan. Model SaaS memberikan kemudahan dalam pengelolaan aplikasi, tetapi juga menimbulkan kebutuhan untuk menjaga pemisahan dan keamanan data antar pengguna maupun pelanggan yang menggunakan layanan tersebut [1].

Keamanan aplikasi web menjadi aspek penting karena aplikasi POS mengelola informasi yang berkaitan dengan akun pengguna, transaksi, dan aktivitas operasional. Berbagai penelitian terdahulu telah memanfaatkan kerangka *Open Web Application Security Project* (OWASP) Top 10 untuk menganalisis risiko dan kerentanan keamanan pada aplikasi berbasis web [2], [3]. Standar OWASP Top 10:2021 sendiri menyediakan klasifikasi risiko keamanan mutakhir yang umum digunakan dalam evaluasi aplikasi web [4]. Pada aplikasi yang berkomunikasi melalui *Application Programming Interface* (API), mekanisme autentikasi dan otorisasi juga menjadi bagian penting karena API secara

e-ISSN: 2548-6861

langsung menyediakan akses terhadap fungsi dan objek data aplikasi [5].

DWANPOS merupakan sistem POS berbasis SaaS yang menyediakan fungsi pengelolaan produk, pengguna, dan transaksi melalui API. Sistem menggunakan *JSON Web Token* (JWT) sebagai mekanisme autentikasi pengguna. JWT banyak digunakan pada layanan berbasis REST karena memungkinkan informasi autentikasi dibawa dalam setiap permintaan tanpa memerlukan penyimpanan sesi secara langsung pada server [6], serta mampu menjaga integritas klaim pengguna melalui verifikasi tanda tangan kriptografis [7]. Namun, keberhasilan autentikasi tidak secara otomatis menjamin bahwa pengguna hanya dapat mengakses objek yang menjadi haknya. Pemeriksaan otorisasi terhadap objek tetap diperlukan pada setiap operasi yang mengakses atau memodifikasi data.

Salah satu kerentanan yang relevan pada kondisi tersebut adalah *Broken Object Level Authorization* (BOLA), yaitu kegagalan aplikasi dalam menerapkan pemeriksaan otorisasi pada objek yang diminta oleh pengguna [5]. Kerentanan ini dapat terjadi ketika aplikasi hanya memvalidasi bahwa pengguna telah terautentikasi, tetapi tidak memastikan bahwa pengguna tersebut memiliki hak untuk mengakses atau mengubah objek tertentu. Pada sistem transaksi, kelemahan tersebut dapat berdampak terhadap kerahasiaan maupun integritas data karena pengguna dapat memperoleh akses terhadap transaksi yang bukan menjadi kewenangannya.

Beberapa penelitian sebelumnya telah membahas kerentanan yang berkaitan dengan IDOR dan *broken access control*. Kusnana dkk. [8] menganalisis kerentanan IDOR pada aplikasi invoice dan melakukan penilaian tingkat keparahan menggunakan CVSS. Prasetyo dkk. [9] juga mengidentifikasi kerentanan IDOR pada aplikasi pendaftaran sidang. Kedua penelitian tersebut menunjukkan bahwa manipulasi *identifier* dapat digunakan untuk menguji kelemahan pengendalian akses terhadap objek. Sementara itu, penelitian Aziza dan Nugraha [1] membahas pemisahan data pada arsitektur SaaS multi-tenant, tetapi berfokus pada kajian arsitektur dan belum menguji pengendalian akses objek secara langsung pada aplikasi yang menjadi objek penelitian.

Berdasarkan penelitian terdahulu yang ditinjau, pengujian kerentanan IDOR atau *broken access control* masih banyak berfokus pada tahap identifikasi dan penilaian kerentanan. Aspek penerapan perbaikan teknis dan verifikasi melalui pengujian ulang belum menjadi fokus utama pada penelitian yang digunakan sebagai rujukan dalam penelitian ini [8], [9]. Selain itu, kajian mengenai keamanan pemisahan data pada SaaS masih perlu dikaitkan dengan pengujian pengendalian akses pada objek aplikasi secara langsung [1]. Kondisi tersebut menunjukkan perlunya rangkaian pengujian yang tidak hanya mengidentifikasi kerentanan, tetapi juga menerapkan *mitigation control* dan memverifikasi efektivitas perbaikan melalui pengujian ulang.

Berdasarkan permasalahan tersebut, penelitian ini berfokus pada analisis keamanan *endpoint* API transaksi pada DWANPOS. Pengujian dilakukan dengan pendekatan *white-box* melalui analisis implementasi aplikasi, pemindaian kode statis, dan pengujian permintaan API menggunakan Postman. Temuan yang terbukti dapat dieksploitasi diklasifikasikan menggunakan *Common Weakness Enumeration* (CWE) sebagai sistem acuan utama, dipetakan ke OWASP API Security Top 10 2023 dan OWASP Top 10:2021, serta dipetakan dengan teknik pengujian yang relevan pada OWASP *Web Security Testing Guide* (WSTG). Tingkat keparahan kerentanan dinilai menggunakan CVSS, kemudian dilakukan penerapan *mitigation control* dalam bentuk penguatan pada implementasi API yang relevan. Setelah perbaikan diterapkan, pengujian ulang dilakukan menggunakan skenario yang sama, disertai pengujian kontrol positif, untuk membandingkan kondisi keamanan sebelum dan sesudah perbaikan.

Dengan demikian, tujuan penelitian ini adalah mengidentifikasi dan memvalidasi kerentanan pada *endpoint* API transaksi DWANPOS, menentukan tingkat keparahan dan klasifikasi kerentanan, menerapkan *mitigation control* melalui *API hardening*, serta mengevaluasi efektivitas perbaikan berdasarkan hasil pengujian ulang. Pendekatan tersebut diharapkan dapat memberikan gambaran yang lebih terukur mengenai perubahan kondisi keamanan *endpoint* API setelah dilakukan perbaikan.

II. METODE

A. Pendekatan Penelitian

Penelitian ini menggunakan pendekatan kualitatif deskriptif dengan studi kasus pada aplikasi DWANPOS. Penelitian bertujuan untuk mengidentifikasi dan memvalidasi kerentanan keamanan pada *endpoint* API transaksi, menerapkan *mitigation control* terhadap kerentanan yang ditemukan, serta melakukan pengujian ulang untuk mengevaluasi efektivitas perbaikan.

Pendekatan pengujian yang digunakan adalah *white-box security testing*, karena peneliti memiliki akses terhadap *source code*, struktur *endpoint*, mekanisme autentikasi, serta implementasi logika aplikasi. Pengujian dilakukan melalui dua aktivitas utama, yaitu analisis terhadap *source code* (pemindaian statis dan tinjauan manual) untuk mengidentifikasi potensi kelemahan, dan pengujian dinamis menggunakan permintaan HTTP untuk memvalidasi apakah kelemahan tersebut dapat dieksploitasi.

Tahapan pengujian mengacu pada NIST SP 800-115 sebagai pedoman umum dalam pelaksanaan pengujian keamanan teknis [10], yang relevan diterapkan dalam pengujian penetrasi keamanan sistem informasi [11]. Teknik pengujian aplikasi web dan API mengacu pada OWASP *Web Security Testing Guide* (WSTG) [12]. Klasifikasi temuan menggunakan CWE sebagai sistem acuan utama [13], yang dipetakan ke OWASP API Security Top 10 2023 [5] karena

lebih relevan untuk objek pengujian berupa API, serta ke OWASP Top 10:2021 [4] sebagai pemetaan pendamping.

Tahapan penelitian secara sistematis terdiri atas persiapan lingkungan, identifikasi *endpoint*, analisis *source code*, pengujian keamanan, klasifikasi dan penilaian kerentanan, penerapan *mitigation control*, *API hardening*, pengujian ulang (*retest*), dan evaluasi hasil pengujian, sebagaimana disajikan pada Gambar 1.

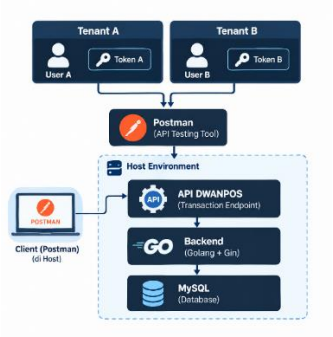


Gambar 1. Tahapan Penelitian

B. Lingkungan Penelitian

Penelitian dilakukan pada lingkungan pengembangan aplikasi DWANPOS yang dijalankan secara lokal menggunakan *virtual machine*. Sistem yang diuji terdiri atas aplikasi *backend* berbasis Golang dan basis data MySQL sebagai penyimpanan data aplikasi. Pengujian dilakukan terhadap mekanisme komunikasi antara pengguna dan *backend* melalui API. Dalam konteks multi-tenant, pengujian melibatkan Tenant A dan Tenant B, masing-masing dengan akun dan token JWT hasil autentikasi sungguhan, untuk mengevaluasi apakah pengguna dari suatu tenant dapat mengakses atau memodifikasi objek transaksi yang dimiliki oleh tenant lain. Topologi dan komponen lingkungan penelitian ini digambarkan pada Gambar 2.

Postman digunakan sebagai alat untuk mengirim dan memodifikasi permintaan HTTP selama pengujian dinamis. Gosec [15] dan Semgrep [16] digunakan untuk pemindaian kode statis. Permintaan HTTP dikirim dari Postman pada host ke aplikasi yang berjalan di virtual machine, dengan alamat akses localhost:8080 atau alamat IP jaringan lokal (192.168.1.28:8080) sesuai konfigurasi jaringan saat pengujian; kedua alamat tersebut merujuk pada instans backend DWANPOS yang sama. Sementara itu, *source code* aplikasi digunakan dalam tahap analisis *white-box* untuk menelusuri implementasi autentikasi, identitas pengguna, parameter objek, serta mekanisme pemeriksaan otorisasi.



Gambar 2. Lingkungan Penelitian

C. Identifikasi dan Pemilihan Endpoint

Identifikasi *endpoint* dilakukan dengan meninjau dokumentasi API dan *source code* aplikasi DWANPOS. Peninjauan difokuskan pada *route*, HTTP *method*, *handler*, mekanisme autentikasi, parameter identitas objek, serta fungsi yang berhubungan dengan data transaksi.

Dari hasil identifikasi tersebut dipilih dua *endpoint* transaksi dan satu endpoint pendukung sebagai objek pengujian. Pemilihan *endpoint* didasarkan pada beberapa kriteria, yaitu:

1) *endpoint* berhubungan dengan fungsi transaksi;
2) *endpoint* memiliki mekanisme autentikasi atau berhubungan dengan kontrol akses pengguna;
3) *endpoint* menerima atau menggunakan parameter yang memengaruhi objek atau proses transaksi;
4) *endpoint* memiliki mekanisme pengambilan, pembuatan, atau perubahan data;
5) terdapat aspek keamanan yang dapat diuji pada *endpoint*; dan
6) kegagalan kontrol keamanan dapat memberikan dampak terhadap kerahasiaan, integritas, atau ketersediaan data/proses transaksi.

Setiap *endpoint* kemudian didokumentasikan berdasarkan metode HTTP, fungsi *endpoint*, parameter yang dapat dimanipulasi, kondisi autentikasi, konteks tenant, serta skenario pengujiannya. Ringkasan kriteria pemilihan *endpoint* transaksi tersebut disajikan pada Tabel 1.

Selain endpoint transaksi, pengujian autentikasi dilakukan pada satu endpoint pendukung, yaitu GET /api/profile. Endpoint ini tidak dipilih berdasarkan kriteria pada Tabel 1, tetapi digunakan sebagai representasi route terproteksi karena memakai middleware JWT yang sama dengan endpoint transaksi, sehingga kelemahan pada middleware tersebut berdampak pada endpoint transaksi yang memerlukan autentikasi.

Tabel 1. Kriteria Pemilihan Endpoint

No	Kriteria	Deskripsi
1	Fungsi transaksi	*Endpoint* berkaitan dengan proses atau data transaksi DWANPOS

e-ISSN: 2548-6861

No	Kriteria	Deskripsi
2	Autentikasi dan kontrol akses	*Endpoint* memiliki atau memerlukan mekanisme autentikasi, atau berhubungan dengan kontrol akses pengguna
3	Parameter transaksi	*Endpoint* menerima parameter yang memengaruhi proses atau objek transaksi
4	Operasi data	*Endpoint* melakukan pengambilan, pembuatan, atau perubahan data
5	Aspek keamanan dapat diuji	Terdapat aspek keamanan yang dapat diuji pada *endpoint*
6	Dampak keamanan	Kegagalan kontrol keamanan berdampak pada kerahasiaan, integritas, atau ketersediaan data/proses transaksi

D. Analisis White-Box dan Security Testing

Analisis *white-box* dilakukan dengan menelusuri implementasi kode pada *endpoint* yang telah dipilih. Analisis difokuskan pada hubungan antara identitas pengguna yang diperoleh dari mekanisme autentikasi dengan objek transaksi yang diminta oleh pengguna. Beberapa bagian yang dianalisis meliputi mekanisme autentikasi, penggunaan identitas pengguna, parameter *object identifier*, proses pengambilan data dari *database*, serta pemeriksaan kepemilikan objek sebelum suatu objek dapat diakses atau dimodifikasi.

Analisis kode dilakukan dengan memadukan pemindaian otomatis dan tinjauan kode manual (*manual code review*). Gosec digunakan sebagai pemindaian dasar, sedangkan Semgrep dilengkapi aturan kustom untuk mendeteksi kueri objek tanpa penyaringan identitas pengguna (*user_id*), karena alat SAST generik umumnya lemah dalam mengidentifikasi cacat logika otorisasi seperti BOLA. Hasil analisis *source code* digunakan untuk mengidentifikasi potensi kelemahan. Potensi kelemahan tersebut tidak langsung dinyatakan sebagai kerentanan, tetapi divalidasi melalui pengujian dinamis.

Pengujian dinamis dilakukan menggunakan Postman dengan mengirimkan permintaan HTTP terhadap *endpoint* yang telah ditentukan. Salah satu skenario pengujian adalah ketika pengguna pada Tenant A mencoba mengakses atau memodifikasi objek transaksi yang dimiliki oleh Tenant B. Selain pengujian negatif tersebut, dilakukan pengujian kontrol positif (*positive control*) untuk memastikan pemilik sah tetap dapat mengakses objek miliknya. Dalam pengujian BOLA, kondisi pengujian dibandingkan dengan kondisi akses yang seharusnya. Apabila pengguna yang telah terautentikasi dapat mengakses atau memodifikasi objek yang bukan menjadi haknya, maka kondisi tersebut dinyatakan sebagai kerentanan *Broken Object Level Authorization*.

Data yang dicatat selama pengujian meliputi:

- *endpoint* dan HTTP *method*;
- kondisi autentikasi;
- identitas pengguna;
- *identifier* objek yang digunakan;
- parameter permintaan;
- HTTP *response*;
- isi respons API;
- status akses terhadap objek; dan
- keberhasilan atau kegagalan perubahan objek.

Skenario pengujian dinamis disusun dalam bentuk matriks pengujian (Tabel 2) yang diturunkan dari hasil analisis white-box. Setiap test case memuat endpoint, skenario pengujian (akun dan teknik manipulasi), dan hasil ekspektasi berdasarkan perilaku keamanan yang seharusnya, serta dipetakan ke OWASP WSTG [12] dan OWASP API Security Top 10 2023 [5] mengacu pada NIST SP 800-115 [10]. Pengujian menggunakan dua akun hasil autentikasi sungguhan melalui endpoint /auth/login, yaitu USR-A (Tenant A) sebagai penguji lintas tenant dan USR-B (Tenant B) sebagai pemilik objek target. Hasil aktual sebelum dan sesudah mitigasi disajikan pada Tabel 8.

Tabel 2. Matriks Pengujian Keamanan Endpoint API Transaksi DWANPOS

ID	Endpoint	Skenario Pengujian	Ekspektasi	Pemetaan
T-01	GET /api/transactions /history	Tanpa token; user_id=USR-B	401 Unauthorized	ATHZ-04 / API1:2023
T-02	GET /api/transactions /history	Token USR-A; manipulasi user_id=USR-B	Hanya data USR-A	ATHZ-04 / API1:2023
T-03	PATCH /api/transactions /:id/status	Token USR-A; akses transaksi Tenant B	Akses ditolak (403/404)	ATHZ-04 / API1:2023
T-04	GET /api/profile	Token demo demo_jwt_token_USR-B	401 Unauthorized	ATHN-04, SESS-10 / API2:2023
T-05	GET /api/transactions /history	Token USR-A; akses riwayat sendiri	200 OK; data USR-A	ATHZ-04 / API1:2023
T-06	PATCH /api/transactions /:id/status	Token USR-A; ubah transaksi milik sendiri	200 OK	ATHZ-04 / API1:2023

E. Identifikasi, Klasifikasi, dan Penilaian Kerentanan

Kerentanan dinyatakan tervalidasi apabila skenario pengujian menunjukkan bahwa pengguna dapat melakukan tindakan terhadap objek yang seharusnya tidak berada dalam kewenangannya. Temuan kerentanan kemudian diklasifikasikan menggunakan taksonomi *Common Weakness Enumeration* (CWE) sebagai sistem acuan utama, antara lain CWE-639 (*Authorization Bypass Through User-Controlled Key*) [13], karena CWE menamai kelemahan secara spesifik pada level implementasi perangkat lunak. Setiap CWE selanjutnya dipetakan ke kategori risiko pada OWASP API Security Top 10 2023, misalnya API1:2023 – *Broken Object Level Authorization* [5], yang lebih relevan untuk objek pengujian berupa API, serta ke OWASP Top 10:2021, misalnya A01:2021 – *Broken Access Control* [4], sebagai pemetaan pendamping.

Pelaksanaan *penetration testing* mengacu pada OWASP *Web Security Testing Guide* (WSTG) sebagai acuan teknis pengujian keamanan aplikasi, dengan pemetaan temuan pada

kode pengujian WSTG yang relevan, yaitu WSTG-ATHZ-04 (*Testing for Insecure Direct Object References*), WSTG-ATHN-04 (*Testing for Bypassing Authentication Schema*), dan WSTG-SESS-10 (*Testing JSON Web Tokens*). Sementara itu, NIST SP 800-115 digunakan sebagai acuan umum dalam proses pengujian keamanan teknis.

Tingkat keparahan kerentanan dinilai menggunakan *Common Vulnerability Scoring System* (CVSS) versi 3.1 [14]. Penilaian mempertimbangkan metrik dasar eksploitabilitas (*Attack Vector, Attack Complexity, Privileges Required, User Interaction*) dan dampak (*Confidentiality, Integrity, Availability*) sehingga setiap temuan memiliki dasar evaluasi kuantitatif yang terukur dan objektif secara keilmuan. Penilaian CVSS digunakan sebagai informasi pendukung dalam menentukan tingkat keparahan temuan, sedangkan efektivitas *mitigation control* ditentukan berdasarkan hasil pengujian ulang terhadap skenario eksploitasi yang sama.

F. Mitigation Control, API Hardening, dan Retest

Setelah kerentanan tervalidasi melalui proses pengujian, dilakukan *mitigation control* berdasarkan akar permasalahan yang ditemukan melalui analisis *source code*. *Mitigation control* bertujuan menghilangkan atau membatasi kondisi yang memungkinkan terjadinya akses tidak sah terhadap objek.

Pada kerentanan autentikasi, perbaikan dilakukan dengan menghapus logika penerimaan token berprefiks demo dan menerapkan verifikasi tanda tangan HMAC-SHA256 pada *middleware* JWT. Pada kerentanan BOLA, perbaikan dilakukan dengan memastikan bahwa objek yang diminta tidak hanya diidentifikasi berdasarkan *object identifier*, tetapi juga dikaitkan dengan identitas pengguna atau konteks kepemilikan yang telah terautentikasi. Dengan demikian, pengguna hanya dapat mengakses atau memodifikasi objek yang berada dalam kewenangannya.

Setelah *mitigation control* diterapkan, dilakukan *API hardening* pada implementasi *endpoint* yang terdampak. *Hardening* dalam penelitian ini mengacu pada penguatan implementasi API, khususnya mekanisme autentikasi dan kontrol akses terhadap objek, sehingga pemeriksaan otorisasi diterapkan secara konsisten pada proses pengambilan maupun perubahan data.

Selanjutnya dilakukan *retest* menggunakan skenario pengujian yang sama dengan pengujian awal. *Retest* dilakukan untuk memastikan bahwa *mitigation control* yang diterapkan mampu mencegah eksploitasi yang sebelumnya berhasil dan tidak menghilangkan akses yang sah terhadap objek milik pengguna. Untuk itu, pengujian ulang disertai pengujian kontrol positif (*positive control*) oleh pemilik sah objek.

Efektivitas perbaikan dievaluasi melalui perbandingan kondisi sebelum dan sesudah penerapan *mitigation control*, dengan memperhatikan hasil eksploitasi, respons HTTP, isolasi data objek, serta hasil pengujian kontrol positif. Parameter evaluasi sebelum dan sesudah penerapan *mitigation control* tersebut dirangkum pada Tabel 3.

Tabel 3. Parameter evaluasi sebelum/sesudah mitigation

Parameter	Sebelum Mitigasi	Setelah Mitigasi
Eksploitasi	Berhasil/Gagal	Berhasil/Gagal
HTTP *response*	Hasil aktual	Hasil aktual
Akses objek	Diizinkan/Ditolak	Diizinkan/Ditolak
Modifikasi objek	Berhasil/Gagal	Berhasil/Gagal
Akses pemilik sah (kontrol positif)	Diizinkan	Diizinkan
Status kerentanan	Open/Confirmed	Closed/Open

Kerentanan dinyatakan tertutup (*closed*) apabila skenario eksploitasi yang sebelumnya berhasil tidak lagi dapat memperoleh akses atau melakukan modifikasi terhadap objek target setelah *mitigation control* diterapkan, sementara akses yang sah terhadap objek milik pengguna tetap berfungsi. Dengan demikian, efektivitas perbaikan tidak hanya ditentukan berdasarkan perubahan pada *source code*, tetapi dibuktikan melalui hasil *retest* terhadap skenario eksploitasi yang sama.

III. HASIL DAN PEMBAHASAN

A. Hasil Identifikasi dan Pemilihan Endpoint

Berdasarkan analisis *source code* dan struktur *routing* pada aplikasi DWANPOS, dilakukan identifikasi terhadap *endpoint* yang berkaitan dengan proses transaksi. Analisis difokuskan pada fungsi *endpoint*, mekanisme autentikasi, penggunaan *object identifier*, serta mekanisme pemeriksaan otorisasi terhadap objek transaksi.

Dari hasil identifikasi tersebut dipilih dua *endpoint* transaksi dan satu *endpoint* pendukung sebagai objek pengujian. Pemilihan *endpoint* didasarkan pada keterkaitannya dengan proses transaksi, kebutuhan autentikasi, penggunaan *object identifier*, serta adanya potensi pengujian kontrol akses terhadap objek. Ketiga *endpoint* API yang terpilih dipaparkan pada Tabel 4.

Tabel 4. Endpoint API yang Menjadi Objek Pengujian

No	Method	Endpoint	Fungsi	Fokus Pengujian
1	GET	/api/transactions/history	Mengambil riwayat transaksi	*Authentication* dan *access control*
2	PATCH	/api/transactions/{id}/status	Mengubah status transaksi	*Object-level authorization*
3*	GET	/api/profile	Mengambil profil pengguna	Autentikasi (middleware JWT bersama)

Keterangan: *Endpoint pendukung

Hasil identifikasi kemudian digunakan sebagai dasar pelaksanaan *white-box analysis* dan pengujian dinamis menggunakan Postman.

B. Hasil Analisis White-Box

Analisis *white-box* dilakukan dengan menelusuri implementasi *endpoint* yang telah dipilih, terutama pada

e-ISSN: 2548-6861

bagian autentikasi, pengambilan identitas pengguna, penerimaan *object identifier*, serta proses pengambilan objek transaksi dari basis data.

Pada *endpoint* PATCH /api/transactions/:id/status, ditemukan bahwa proses pengambilan transaksi sebelum mitigasi dilakukan berdasarkan transaction_id tanpa disertai pembatasan berdasarkan identitas pengguna yang telah terautentikasi. Kondisi tersebut secara konseptual dapat digambarkan sebagai berikut:

```
transaction_id -> pencarian objek transaksi

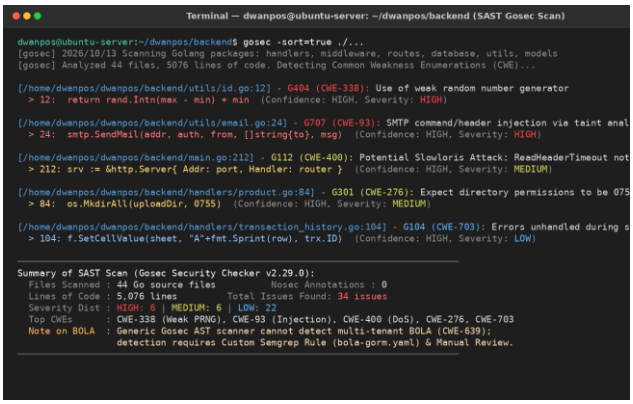
tanpa adanya pemeriksaan:
transaction_id + user_id -> kepemilikan objek
```



Gambar 3. Analisis Source Code pada Proses Pengambilan Objek Transaksi Sebelum Mitigasi

Potongan *source code* pada Gambar 3 memperlihatkan bahwa pencarian dan pengambilan objek pada fungsi UpdateTransactionStatus (berkas handlers/transaction.go) dilakukan semata-mata berdasarkan parameter URL transaction_id tanpa disertai pemeriksaan relasi kepemilikan terhadap identitas pengguna (user_id) yang terautentikasi.

Hasil analisis kode statis (SAST) menggunakan Gosec [15] disajikan pada Gambar 4. Pemindaian otomatis dilakukan terhadap seluruh modul *backend* Golang aplikasi DWANPOS yang mencakup 44 berkas kode sumber dan 5.076 baris kode. Dari pemindaian tersebut, Gosec berhasil mengidentifikasi 34 temuan kerentanan keamanan yang terbagi menjadi 6 temuan berstatus *High*, 6 temuan *Medium*, dan 22 temuan *Low*. Temuan tingkat keparahan *High* didominasi oleh penggunaan generator angka acak yang lemah secara kriptografis (G404/CWE-338) berbasis pustaka math/rand pada pembuatan token dan ID transaksi (berkas utils/id.go dan handlers/auth.go), serta potensi injeksi perintah SMTP (G707/CWE-93) pada utils/email.go. Pada tingkat keparahan *Medium*, terdeteksi potensi serangan *Denial of Service* tipe *Slowloris* (G112/CWE-400) akibat tidak ditetapkannya ReadHeaderTimeout pada konfigurasi http.Server (main.go), serta izin hak akses berkas dan direktori yang terlalu longgar (G301/G302 dengan izin 0755/0644 alih-alih <= 0750/0600 pada modul handlers/product.go). Sementara itu, temuan tingkat *Low* didominasi oleh galat kembalian fungsi yang tidak ditangani (G104/CWE-703) pada modul pembacaan basis data dan ekspor berkas *spreadsheet*.



Gambar 4. Hasil Pemindaian Static Application Security Testing (SAST) Menggunakan Gosec pada Backend DWANPOS

Meskipun Gosec efektif dalam mendeteksi kelemahan sintaksis dan implementasi pustaka bawaan Go, analisis menunjukkan adanya keterbatasan inheren (*false negative*) pada alat SAST generik terhadap kerentanan otorisasi logika bisnis multi-tenant seperti *Broken Object Level Authorization* (BOLA/CWE-639). Gosec tidak memiliki aturan semantik untuk memverifikasi apakah kueri *Object-Relational Mapping* (ORM) telah menyertakan filter klaim identitas penyewa. Oleh karena itu, penelitian ini memadukan Gosec dengan Semgrep yang dilengkapi aturan kustom *Abstract Syntax Tree* (AST) berbasis rule gorm-missing-tenant-filter serta tinjauan kode manual (*manual code review*). Pendekatan kombinasi ini berhasil mengidentifikasi ketiadaan klausa penyaringan kepemilikan (user_id/store_id) pada kueri basis data fungsi UpdateTransactionStatus (handlers/transaction.go) dan GetTransactionHistory (handlers/transaction_history.go). Cakupan temuan yang berhasil ditangkap dan yang terlewat oleh masing-masing pendekatan dirangkum pada Tabel 5.

Tabel 5. Cakupan Pendekatan Analisis Statis

Kelemahan	Gosec	Semgrep kustom + manual review
Weak PRNG (G404/CWE-338)	Terdeteksi	Tidak menjadi target
Potensi injeksi SMTP (G707/CWE-93)	Terdeteksi	Tidak menjadi target
Slowloris (G112/CWE-400)	Terdeteksi	Tidak menjadi target
Izin berkas longgar (G301/G302)	Terdeteksi	Tidak menjadi target
Galat tidak ditangani (G104/CWE-703)	Terdeteksi	Tidak menjadi target
Kueri objek tanpa filter user_id (BOLA) pada UpdateTransactionStatus dan GetTransactionHistory	Terlewat	Terdeteksi
Bypass autentikasi lewat prefiks demo_jwt_token_ (VULN-03)	Terlewat	Teridentifikasi melalui manual code review

Hasil *white-box analysis* tersebut menunjukkan adanya potensi kelemahan pada kontrol otorisasi objek. Namun, keberadaan kelemahan pada *source code* belum secara langsung membuktikan bahwa kerentanan dapat

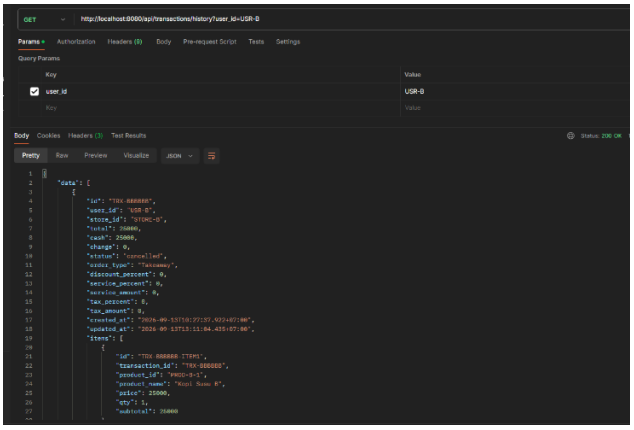
dieksploitasi. Oleh karena itu, dilakukan validasi melalui pengujian dinamis menggunakan Postman.

C. Hasil Pengujian BOLA pada Pengambilan Data

Pengujian pertama dilakukan terhadap *endpoint*:

```
GET /api/transactions/history
```

Skenario pengujian dilakukan dengan mengirimkan parameter user_id=USR-B tanpa menyertakan autentikasi (T-01). Berdasarkan kontrol keamanan yang diharapkan, *request* tanpa autentikasi seharusnya ditolak oleh sistem. Namun, hasil pengujian menunjukkan bahwa server memberikan respons 200 OK dan mengembalikan data transaksi yang terkait dengan USR-B.



Gambar 5. Hasil Pengujian BOLA pada Endpoint GET /api/transactions/history Sebelum Mitigasi

Hasil pengujian Postman pada Gambar 5 memperlihatkan bahwa permintaan HTTP GET terhadap *endpoint* /api/transactions/history dengan parameter *query* user_id=USR-B berhasil memperoleh respons HTTP 200 OK tanpa menyertakan token autentikasi, sehingga data transaksi finansial milik Tenant B bocor ke pihak yang tidak berwenang.

Hasil tersebut membuktikan bahwa pengguna yang tidak terautentikasi dapat memperoleh data transaksi berdasarkan user_id yang dikendalikan melalui *request*. Kondisi ini menunjukkan kegagalan kontrol akses terhadap objek dan menyebabkan terjadinya pengungkapan data transaksi. Temuan tersebut diklasifikasikan sebagai CWE-862 (*Missing Authorization*), dipetakan ke API1:2023 – *Broken Object Level Authorization* dan A01:2021 – *Broken Access Control*.

D. Hasil Pengujian BOLA pada Perubahan Data

Pengujian berikutnya dilakukan terhadap *endpoint*:

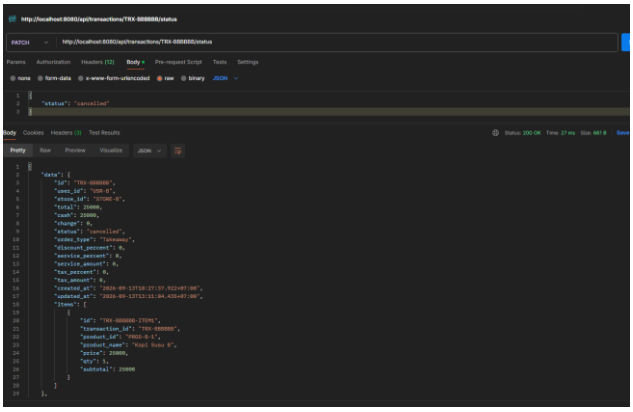
```
PATCH /api/transactions/:id/status
```

Pengujian menggunakan identitas USR-A untuk mengakses transaksi TRX-BBBBBB (T-03) yang merupakan objek milik USR-B. *Request* yang digunakan adalah:

```
PATCH /api/transactions/TRX-BBBBBB/status
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
(Token JWT Asli USR-A)

{
  "status": "cancelled"
}
```

Berdasarkan mekanisme otorisasi yang seharusnya diterapkan, *request* tersebut harus ditolak karena pengguna USR-A tidak memiliki kewenangan terhadap transaksi milik USR-B. Namun, sistem memberikan respons 200 OK dan perubahan status transaksi berhasil dilakukan.



Gambar 6. Hasil Pengujian BOLA pada Endpoint PATCH /api/transactions/:id/status Sebelum Mitigasi

Bukti pengujian pada Gambar 6 memperlihatkan keberhasilan manipulasi objek transaksi, di mana akun terautentikasi USR-A (Tenant A) menggunakan token JWT HMAC-SHA256 sah hasil *login* sungguhan pada *endpoint* /auth/login. Token sah milik USR-A tersebut dikirimkan dalam permintaan HTTP PATCH untuk memperbarui status transaksi TRX-BBBBBB milik USR-B (Tenant B). Sistem merespons dengan HTTP 200 OK dan mengubah status transaksi menjadi cancelled, membuktikan ketiadaan validasi kepemilikan objek pada level *handler* (BOLA murni tanpa ketergantungan *token bypass*).

Hasil tersebut menunjukkan adanya kelemahan *object-level authorization* yang memungkinkan pengguna terautentikasi melakukan modifikasi terhadap objek yang bukan menjadi kewenangannya. Berbeda dengan pengujian sebelumnya yang terutama berdampak pada *confidentiality*, temuan ini memiliki dampak langsung terhadap *integrity* data transaksi. Temuan ini diklasifikasikan sebagai CWE-639 (*Authorization Bypass Through User-Controlled Key*), dipetakan ke API1:2023 dan A01:2021.

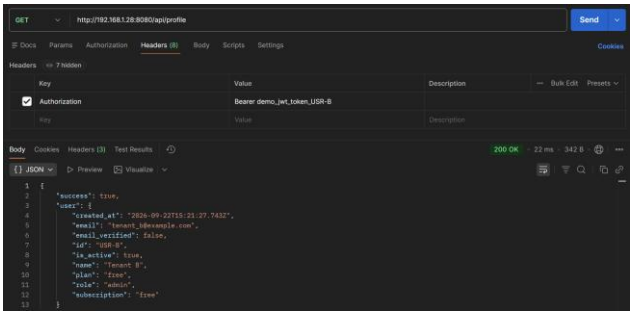
E. Hasil Pengujian Authentication

Selain pengujian BOLA, dilakukan pengujian terhadap mekanisme autentikasi (T-04). Pada kondisi sebelum mitigasi, sistem menerima token dengan pola tertentu tanpa melakukan validasi JWT secara kriptografis. Pengujian dilakukan menggunakan token:

```
Authorization: Bearer demo_jwt_token_USR-B
```

e-ISSN: 2548-6861

Token tersebut seharusnya ditolak apabila tidak memiliki tanda tangan JWT yang valid. Namun, sistem memberikan akses berdasarkan pola token tersebut.



Gambar 7. Hasil Pengujian Authentication Bypass Sebelum Mitigasi

Hasil pengujian pada Gambar 7 membuktikan terjadinya *authentication bypass*, di mana pengiriman token dengan prefiks *hardcoded* demo_jwt_token_USR-B pada *header* Authorization berhasil memperoleh respons HTTP 200 OK beserta seluruh data profil milik Tenant B tanpa melewati proses verifikasi tanda tangan kriptografis. Temuan ini menunjukkan bahwa mekanisme autentikasi dapat dilewati menggunakan token yang tidak tervalidasi secara kriptografis. Kondisi tersebut menjadi risiko tambahan karena identitas pengguna yang digunakan dalam proses otorisasi dapat dipercaya secara tidak semestinya.

Pengujian bypass dilakukan pada endpoint *GET /api/profile* sebagai representasi route terproteksi, karena kelemahan berada pada middleware JWT yang digunakan bersama oleh route terproteksi, termasuk endpoint transaksi. Respons memuat profil USR-B dengan atribut role admin, sehingga identitas yang dipercaya oleh endpoint terproteksi dapat dipalsukan tanpa kredensial.

F. Klasifikasi dan Penilaian Kerentanan

Berdasarkan hasil *white-box analysis* dan pengujian dinamis, kerentanan yang berhasil divalidasi kemudian diklasifikasikan menggunakan CWE sebagai sistem acuan utama, dipetakan ke OWASP API Security Top 10 2023, serta ke OWASP Top 10:2021 sebagai pemetaan pendamping. Ringkasan hasil identifikasi dan klasifikasi kerentanan tersebut disajikan pada Tabel 6.

Tabel 6. Ringkasan Hasil Identifikasi Kerentanan

ID	Temuan dan Endpoint	CWE	OWASP API 2023 / Web 2021	Dampak
BOLA-GET-01 (T-01, T-02, T-05)	BOLA / *cross-tenant data leakage*; GET /api/transactions/history	CWE-862 (*Missing Authorization*)	API1:2023 *Broken Object Level Authorization* / A01:2021 *Broken Access Control*	*Confidentiality*
BOLA-PATCH-01 (T-03, T-06)	BOLA / *unauthorized status modification*; PATCH /api/transactions/:id/status	CWE-639 (*Authorization Bypass Through User-Controlled Key*)	API1:2023 *Broken Object Level Authorization* / A01:2021 *Broken Access Control*	*Integrity* dan *confidentiality*
VULN-03 (T-04)	*Authentication bypass* (*hardcoded demo prefix*); GET /api/profile (*global JWT*)	CWE-287 / CWE-347 (*Improper Authentication / Improper Verification of Cryptographic Signature*)	API2:2023 *Broken Authentication* / A07:2021 *Identification and Authentication Failures*	*Authentication* (*Critical*)

Penilaian tingkat keparahan risiko terhadap kerentanan yang teridentifikasi dilakukan menggunakan CVSS versi 3.1, sebagaimana dirangkum pada Tabel 7.

Tabel 7. Penilaian CVSS 3.1

Temuan	CVSS Vector	Skor	Severity
Authentication Bypass (VULN-03)	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	9.8	Critical
BOLA – *Transaction Modification* (PATCH)	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:H/A:L	7.6	High
BOLA – *Data Leakage* (GET *History*)	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	7.5	High

Hasil penilaian menunjukkan bahwa temuan BOLA berada pada tingkat *High* dengan skor CVSS 7.6 (pada *endpoint* PATCH status) dan 7.5 (pada *endpoint* GET riwayat), sedangkan kelemahan autentikasi (VULN-03) berada pada tingkat *Critical* (skor 9.8). Pada *endpoint* PATCH, metrik *Privileges Required* (PR) dinilai *Low* (PR:L) karena penyerang memerlukan autentikasi sah pada tingkat pengguna biasa (Tenant A) untuk mengeksploitasi data penyewa lain. Metrik *Confidentiality* (C) dinilai *Low* (C:L) karena respons HTTP PATCH mengembalikan *payload* rincian transaksi yang dimodifikasi kepada penyerang yang tidak berwenang, sehingga menaikkan skor keparahan dari 7.1 menjadi 7.6. Sementara itu, temuan VULN-03 berbobot *Critical* (9.8) karena ketiadaan verifikasi tanda tangan kriptografis memungkinkan pengambilalihan akun secara penuh tanpa kredensial.

G. Implementasi Mitigation Control dan API Hardening

Perbaikan autentikasi (T-04). Akar masalah berada pada fungsi extractUserIDFromToken di middleware/auth.go, yang memiliki cabang kompatibilitas lama: token berprefiks demo_jwt_token_ dilewatkan tanpa verifikasi tanda tangan dan sisa string dipercaya sebagai user_id. Perbaikan dilakukan dengan menghapus cabang strings.HasPrefix(tokenString, “demo_jwt_token_”) sepenuhnya, sehingga seluruh token wajib diverifikasi menggunakan jwt.ParseWithClaims() terhadap kunci rahasia JWT_SECRET dari environment variable (HMAC-SHA256). Token yang tidak bertanda tangan valid, kedaluwarsa, atau tidak berformat benar ditolak dengan HTTP 401 Unauthorized.

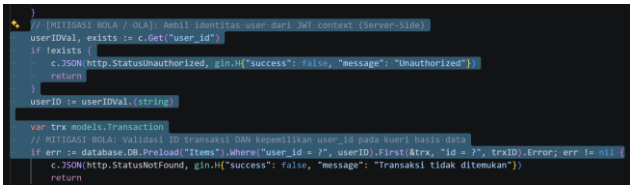
Perbaikan otorisasi pada pengambilan data (T-01, T-02). Endpoint riwayat transaksi (termasuk /export dan /export/pdf) sebelumnya tidak dilindungi AuthMiddleware() dan menerima user_id langsung dari query parameter.

AuthMiddleware() diterapkan pada route GET transaksi di routes/transaction.go, sehingga permintaan tanpa token ditolak dengan HTTP 401. Handler pada handlers/transaction_history.go tidak lagi membaca c.Query("user_id"), tetapi mengambil identitas dari konteks JWT melalui c.MustGet("user_id") dan mengikatnya ke kueri basis data.

Perbaikan otorisasi pada perubahan data (T-03). Pada endpoint perubahan status transaksi, mekanisme pengambilan objek diperkuat dengan menambahkan identitas pengguna sebagai bagian dari kondisi pencarian objek. Apabila objek tidak ditemukan atau bukan milik pemohon, sistem mengembalikan HTTP 404 Not Found, bukan 403, sehingga penyerang tidak dapat membedakan ID transaksi yang tidak ada dari ID milik pengguna lain.

```
Sebelum mitigasi:
  transaction_id
    -> ambil transaksi

Setelah mitigasi:
  transaction_id + user_id
    -> validasi ownership
    -> ambil transaksi
```



Gambar 8. Implementasi Mitigation Control pada Pemeriksaan Kepemilikan Objek

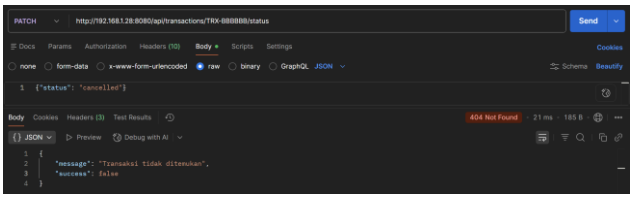
Implementasi kontrol mitigasi pada Gambar 8 memperlihatkan penguatan otorisasi objek (*API hardening*) pada fungsi UpdateTransactionStatus. *Handler* kini mengekstrak klaim identitas user_id dari konteks JWT dan mengikatnya ke dalam klausa kueri ORM (.Where("user_id = ?", userID)), sehingga sistem memastikan transaksi hanya dapat dimodifikasi oleh pemilik sah objek tersebut. Perubahan tersebut merupakan bentuk *API hardening* karena kontrol akses diperkuat langsung pada implementasi *endpoint*, bukan hanya mengandalkan validasi pada sisi klien.

H. Hasil Retest dan Evaluasi Efektivitas Mitigasi

Setelah *mitigation control* diterapkan, dilakukan pengujian ulang menggunakan skenario yang sama dengan pengujian awal. Penggunaan skenario yang sama bertujuan agar kondisi sebelum dan sesudah perbaikan dapat dibandingkan secara langsung. Selain pengujian negatif, dilakukan pengujian kontrol positif (*positive control*) oleh pemilik sah objek. Perbandingan hasil pengujian keamanan sebelum dan sesudah penerapan mitigasi disajikan pada Tabel 8.

Tabel 8. Perbandingan Hasil Pengujian Sebelum dan Sesudah Mitigasi

ID dan Tipe	Sebelum Mitigasi	Setelah Mitigasi (Hasil Retest)
T-01 (*Negative Test*)	200 OK – parameter user_id=USR-B membocorkan riwayat Tenant B	401 Unauthorized (tanpa token)
T-02 (*Negative Test*)	Data Tenant B dikembalikan (berdasarkan analisis kode; user_id dari query parameter dipercaya tanpa verifikasi dinamis)	200 OK – data hanya menampilkan transaksi USR-A (identitas diambil dari klaim JWT, bukan query parameter)
T-03 (*Negative Test*)	200 OK – token sah USR-A berhasil mengubah status TRX-BBBBBB	404 Not Found (transaksi tidak ditemukan pada tenant USR-A)
T-04 (*Negative Test*)	200 OK – token demo_jwt_token_USR-B melewati verifikasi tanda tangan	401 Invalid Token (wajib tanda tangan HMAC-SHA256 valid)
T-05 (*Positive Control*)	200 OK – pemilik sah USR-A meminta riwayat transaksi miliknya	200 OK – berhasil, riwayat transaksi USR-A ditampilkan normal
T-06 (*Positive Control*)	200 OK – pemilik sah USR-A memperbarui status transaksi miliknya (TRX-AAAAAA)	200 OK – berhasil, status terupdate tanpa gangguan fungsional



Gambar 9. Hasil Retest BOLA Setelah Penerapan Mitigation Control

Hasil pengujian ulang (*retest*) pada Gambar 9 menunjukkan bahwa upaya eksploitasi manipulasi status transaksi TRX-BBBBBB oleh USR-A menghasilkan respons HTTP 404 Not Found dengan pesan "Transaksi tidak ditemukan". Hal ini membuktikan bahwa kontrol mitigasi kepemilikan objek berhasil memutus akses lintas tenant dan menutup celah kerentanan BOLA pada skenario yang diuji. Di sisi lain, pengujian kontrol positif (*positive control*) membuktikan bahwa pemilik data yang sah (Tenant A) tetap dapat mengakses riwayat transaksinya serta memperbarui status transaksinya sendiri (TRX-AAAAAA) dengan respons HTTP 200 OK. Dengan demikian, pengerasan keamanan tidak merusak fungsionalitas normal operasional bisnis pengguna yang berhak.

Retest pada endpoint GET /api/transactions/history menunjukkan bahwa permintaan tanpa token dengan parameter user_id=USR-B (T-01) kini menghasilkan HTTP 401 *Unauthorized*. Permintaan menggunakan token JWT valid milik USR-A dengan parameter user_id=USR-B (T-02) tidak lagi mengembalikan data Tenant B; respons hanya memuat transaksi milik USR-A karena identitas diambil dari klaim JWT yang terverifikasi, bukan dari parameter query. Pada T-02, kondisi sebelum mitigasi tidak diuji secara dinamis; berdasarkan analisis kode, parameter user_id dari query dipercaya tanpa verifikasi. *Retest* pada endpoint GET

e-ISSN: 2548-6861

/api/profile menggunakan token demo (T-04) menghasilkan HTTP 401 Invalid Token, sehingga hanya token bertanda tangan HMAC-SHA256 yang valid yang diterima.

Berdasarkan hasil *retest* aktual, efektivitas *mitigation control* ditentukan dengan melihat apakah eksploitasi yang sebelumnya berhasil masih dapat dilakukan serta apakah akses yang sah terhadap objek pengguna tetap berfungsi.

I. Pembahasan

Hasil penelitian menunjukkan bahwa permasalahan utama pada BOLA bukan terletak pada keberadaan transaction_id semata, tetapi pada tidak adanya hubungan antara *object identifier* dengan identitas pengguna yang telah terautentikasi. Pada skenario pengambilan data, sistem mempercayai user_id yang diberikan melalui *request* sehingga pengguna dapat memperoleh objek milik pengguna lain. Pada skenario perubahan data, pengguna telah berhasil melewati autentikasi dengan token sah, tetapi sistem tidak melakukan pemeriksaan apakah transaksi yang diminta merupakan objek yang berada dalam kewenangan pengguna tersebut.

Temuan tersebut menunjukkan bahwa *authentication* dan *authorization* merupakan dua kontrol keamanan yang berbeda. Pengguna yang berhasil diautentikasi belum tentu memiliki hak untuk mengakses seluruh objek dalam sistem. Pada lingkungan SaaS multi-tenant, kondisi tersebut menjadi lebih kritis karena kegagalan *object-level authorization* dapat menyebabkan akses lintas tenant. Oleh karena itu, pemeriksaan kepemilikan objek perlu diterapkan pada sisi server untuk setiap operasi yang mengambil atau memodifikasi objek berdasarkan *identifier* dari klien.

Selain itu, hasil analisis statis menunjukkan bahwa pemindaian otomatis generik tidak cukup untuk mendeteksi cacat logika otorisasi. Gosec mendeteksi kelemahan sintaksis dan penggunaan pustaka, tetapi tidak mendeteksi ketiadaan filter kepemilikan pada kueri objek, yang baru teridentifikasi melalui aturan kustom Semgrep dan tinjauan kode manual. Pengujian kontrol positif juga diperlukan agar perbaikan keamanan tidak menghilangkan akses sah pemilik data.

IV. KESIMPULAN

Penelitian ini telah melakukan pengujian keamanan terhadap endpoint API transaksi DWANPOS serta middleware autentikasi bersama yang melindunginya menggunakan pendekatan *white-box security testing*. Pengujian mengacu pada NIST SP 800-115 dan *OWASP Web Security Testing Guide* (WSTG), dengan CWE sebagai sistem klasifikasi utama yang dipetakan ke *OWASP API Security Top 10:2023* dan *OWASP Top 10:2021*, serta CVSS versi 3.1 untuk penilaian tingkat keparahan.

Hasil pengujian menunjukkan adanya kelemahan autentikasi berupa penerimaan token demo tanpa verifikasi tanda tangan (CWE-287/CWE-347, API2:2023, A07:2021) dengan skor CVSS 9,8 (*Critical*), serta *Broken Object Level Authorization* (BOLA) (CWE-639/CWE-862, API1:2023, A01:2021) dengan skor CVSS 7,6 dan 7,5 (*High*) yang memungkinkan pengguna mengakses dan memodifikasi objek transaksi di luar kewenangannya. Pemindaian *Static Application Security Testing* (SAST) menggunakan Gosec tidak mendeteksi BOLA maupun *authentication bypass*, sedangkan aturan kustom Semgrep dan tinjauan kode manual berhasil mengidentifikasi ketiadaan filter kepemilikan pada kueri objek dan logika penerimaan token demo.

Berdasarkan akar permasalahan yang ditemukan melalui analisis *white-box*, diterapkan *mitigation control* pada mekanisme autentikasi dan pemeriksaan kepemilikan objek, kemudian dilakukan *API hardening* pada endpoint yang terdampak. Efektivitas perbaikan dievaluasi melalui *retest* menggunakan skenario eksploitasi yang sama, disertai pengujian kontrol positif (*positive control*). Hasil *retest* menunjukkan bahwa eksploitasi lintas *tenant* ditolak pada skenario yang diuji, sementara pemilik sah objek tetap dapat mengakses dan memperbarui datanya.

Penelitian ini memiliki beberapa batasan. Pertama, pengujian dibatasi pada dua endpoint transaksi (GET /api/transactions/history dan PATCH /api/transactions/:id/status) serta satu endpoint pendukung (GET /api/profile) yang digunakan untuk membuktikan kelemahan pada middleware JWT bersama. Oleh karena itu, hasil penelitian tidak dimaksudkan untuk menyatakan kondisi keamanan seluruh endpoint DWANPOS. Kedua, pengujian dinamis dilakukan secara manual menggunakan Postman terhadap skenario yang diturunkan dari hasil analisis *white-box*, dengan teknik berupa pengujian tanpa token, manipulasi parameter pada *query* dan *path*, serta pemalsuan token pada *header Authorization*. Penelitian ini belum menggunakan alat bantu pengujian otorisasi seperti Burp Suite dengan ekstensi Autorize. Selain itu, penelitian belum menguji variasi teknik lain, seperti manipulasi metode HTTP, manipulasi parameter pada *body*, *mass assignment*, serta pembungkusan ID dalam format *array* atau JSON. Ketiga, pengujian lintas *tenant* dilakukan dengan USR-A sebagai penguji dan USR-B sebagai pemilik objek target, sehingga pengujian dengan arah sebaliknya belum dilakukan.

Penelitian selanjutnya dapat memperluas pengujian pada endpoint lainnya, termasuk endpoint POST, serta kategori kerentanan *OWASP Top 10:2021* yang belum diuji. Matriks pengujian dapat dikembangkan dengan variasi teknik yang telah disebutkan, didukung pengujian otorisasi otomatis menggunakan Burp Suite dan Autorize untuk memperluas cakupan pengujian, serta skenario multi-*tenant* dengan variasi pengguna dan hak akses yang lebih kompleks.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Politeknik Negeri Batam, dosen pembimbing, serta pihak-pihak yang telah memberikan bimbingan, dukungan, dan bantuan dalam pelaksanaan penelitian hingga penyusunan artikel ini.

DAFTAR PUSTAKA

[1] S. T. N. Aziza and Y. Nugraha, “Analisis Trade-Off Arsitektur Multi-Tenant Sebagai Dasar Justifikasi Strategi Basis Data SaaS”, Progresif J. Ilmi. Kom, vol. 22, no. 2, pp. 425–435, May 2026.

[2] A. R. Supriyatna, I. Asrowardi, S. D. Putra, and E. Subyantoro, "Analisis Kerentanan Aplikasi Web E-commerce Berdasarkan Standar OWASP Top 10: Studi Kasus pada Situs Kopi Lampung Nusantara," *EXPERT: Jurnal Manajemen Sistem Informasi dan Teknologi*, vol. 14, no. 2, p. 95, 2024, doi: 10.36448/expert.v14i2.4034.

[3] A. Zaini and R. Wijanarko, "Analisis Keamanan Website Menggunakan Standar Keamanan Open Web Application Security Project (OWASP) Studi Kasus Website Penerimaan Mahasiswa Baru Universitas Wahid Hasyim Semarang," *Jurnal Informatika dan Rekayasa Perangkat Lunak*, vol. 5, no. 2, pp. 262-272, 2023, doi: 10.36499/jinrpl.v5i2.9351.

[4] OWASP Foundation, "OWASP Top 10:2021." [Online]. Tersedia: https://owasp.org/Top10/2021/

[5] OWASP Foundation, "OWASP API Security Top 10 2023." [Online]. Tersedia: https://owasp.org/API-Security/editions/2023/en/0x00-header/

[6] R. Gunawan and A. Rahmatulloh, "JSON Web Token (JWT) untuk Authentication pada Interoperabilitas Arsitektur berbasis RESTful Web Service," *Jurnal Edukasi dan Penelitian Informatika (JEPIN)*, vol. 5, no. 1, p. 74, 2019, doi: 10.26418/jp.v5i1.27232.

[7] A. Rahmatulloh, H. Sulastri, and R. Nugroho, "Keamanan RESTful Web Service Menggunakan JSON Web Token (JWT) HMAC SHA512," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*, vol. 7, no. 2, 2018.

[8] Kusnana, P. Gymnastiar, and B. Riswanto, "Analisis Kerentanan Insecure Direct Object Reference (IDOR) dan Broken Access Control pada Aplikasi Invoice Berbasis Web," *Digital Transformation Technology*, vol. 5, no. 2, pp. 106-114, 2025, doi: 10.47709/digitech.v5i2.6948.

[9] S. E. Prasetyo, Haeruddin, and Tiara, "Evaluasi Kerentanan Insecure Direct Object Reference pada Aplikasi Pendaftaran Sidang Universitas XYZ," *Journal of Applied Computer Science and Technology*, vol. 5, no. 2, pp. 165-171, 2024, doi: 10.52158/jacost.v5i2.873.

[10] National Institute of Standards and Technology, "Technical Guide to Information Security Testing and Assessment," NIST Special Publication 800-115, 2008. [Online]. Tersedia: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-115.pdf

[11] S. S. Anelia, Jayanta, and B. Hananto, "Uji Penetrasi Server Universitas PQR Menggunakan Metode National Institute of Standards and Technology (NIST SP 800-115)," *Jurnal Ilmu Teknik dan Komputer*, vol. 7, no. 1, pp. 35-43, 2023, doi: 10.22441/jitkom.2023.v7i1.005.

[12] OWASP Foundation, "OWASP Web Security Testing Guide v4.2." [Online]. Tersedia: https://wstg.owasp.org/

[13] The MITRE Corporation, "CWE-639: Authorization Bypass Through User-Controlled Key," *Common Weakness Enumeration (CWE)*, 2023. [Online]. Tersedia: https://cwe.mitre.org/data/definitions/639.html. [Diakses: 10 Okt. 2026].

[14] Forum of Incident Response and Security Teams (FIRST), "Common Vulnerability Scoring System v3.1: Specification Document," 2019. [Online]. Tersedia: https://www.first.org/cvss/v3.1/specification-document. [Diakses: 10 Okt. 2026].

[15] Securego, "Gosec - Golang Security Checker," 2024. [Online]. Tersedia: https://github.com/securego/gosec. [Diakses: 10 Okt. 2026].

[16] Semgrep, Inc., "Semgrep: Static Analysis Tool." [Online]. Tersedia: https://semgrep.dev/. [Diakses: 14 Okt. 2026].