PANEL ADMIN MULTI-PERAN BERBASIS RBAC DENGAN *PAYMENT GATEWAY* MIDTRANS DAN *AUTOMATED DUNNING* PADA ISP LOKAL

Supangat, Ridho Aji Nugroho
Teknik Informatika, Universitas 17 Agustus 1945 Surabaya
supangat@untag-sby.ac.id

ABSTRAK

Industri penyedia layanan internet (ISP) skala lokal di Indonesia menghadapi tekanan transformasi digital yang membutuhkan sistem operasional terintegrasi. Wikarta, ISP berbasis WISP di Surabaya, menghadapi tiga permasalahan utama: tidak adanya pembatasan hak akses berbasis peran yang terstruktur, proses rekonsiliasi pembayaran manual yang rawan ketidaksesuaian data (*payment mismatch*), dan ketiadaan sistem penagihan terstruktur yang menyebabkan tingkat keterlambatan pembayaran yang tinggi. Penelitian ini merancang dan mengimplementasikan panel admin berbasis web menggunakan Laravel 12.x yang mengintegrasikan tiga komponen: (1) *Role-Based Access Control* (RBAC) dengan empat peran menggunakan Spatie Laravel *Permission* berdasarkan prinsip *least privilege* dan *separation of duties*; (2) integrasi Midtrans dengan *signature* verification HMAC-SHA512 dan *idempotency mechanism* berbasis SHA-256; (3) *automated dunning* system berbasis *Laravel Queue* dan Horizon dengan notifikasi multi-*channel* pada tahap T-7, T-3, T-1, dan T+3. Pengembangan menggunakan model Waterfall. Evaluasi meliputi *black-box testing*, *white-box testing* (coverage ?70%), *security testing* OWASP WSTG v4.2, *performance testing*, dan UAT (SUS ?80). Hasil: black-box *pass rate* 100%, *code coverage* 73,2%, seluruh skenario OWASP aman, *webhook* p95 3,8 detik, SUS s*core* 82,4, pelanggaran akses 0%, *payment mismatch* 0,3%, *dunning SLA* 99,7%. Penelitian ini menyediakan model implementasi panel admin terintegrasi bagi ISP skala kecil-menengah Indonesia.

Kata kunci : *Role-Based Access Control, Payment Gateway, Midtrans, Automated Dunning, Webhook Security, ISP*

1. PENDAHULUAN

Industri penyedia layanan internet (*Internet Service Provider*/ISP) di Indonesia tumbuh pesat, khususnya ISP skala kecil-menengah berbasis RT/RW Net dan *Wireless ISP* (WISP) yang melayani wilayah suburban dan rural. Karakteristik bisnisnya berbasis langganan dengan tagihan periodik dan pengelolaan pelanggan yang masif, sehingga tanpa sistem informasi yang memadai pertumbuhan pelanggan justru menurunkan efisiensi operasional.

Wikarta, ISP lokal berbasis WISP di Surabaya, menghadapi tiga permasalahan operasional yang saling berkaitan: (1) tidak adanya pembatasan hak akses berbasis peran, sehingga seluruh staf dapat mengakses semua data dan fungsi tanpa kendali — meningkatkan risiko insider threat; (2) rekonsiliasi pembayaran yang sepenuhnya manual sehingga rawan *payment mismatch* terutama pada periode peak awal bulan; dan (3) ketiadaan sistem penagihan terstruktur, di mana *dunning* bergantung pada upaya manual via WhatsApp tanpa jadwal dan dokumentasi terverifikasi.

Penelitian terdahulu telah membuktikan efektivitas masing-masing komponen secara terpisah: Nasich et al. [1] mengeliminasi pelanggaran akses dari 72,73% menjadi 0% melalui RBAC berbasis Spatie; Djuwitaningrum dan Jati [2] membahas integrasi Midtrans pada sistem berbasis Laravel; dan Wijaya [3] menerapkan *automated dunning* pada layanan berlangganan. Namun, ketiga penelitian tersebut belum mengintegrasikan kontrol akses multi-peran, keamanan *webhook* pembayaran, dan penagihan otomatis sebagai satu alur operasional ISP. Studi RBAC berfokus pada otorisasi pengguna, studi pembayaran berfokus pada transaksi, dan studi *dunning* berfokus pada penagihan. Karena itu, belum tersedia model yang secara terpadu menjamin akses staf terkendali, status pembayaran diproses tepat satu kali, serta penagihan dilakukan berdasarkan status invoice yang valid. Kesenjangan inilah yang menjadi dasar penelitian ini dalam konteks ISP lokal Indonesia.

Untuk menjawab kesenjangan tersebut, penelitian ini bertujuan merancang, mengimplementasikan, dan mengevaluasi panel admin berbasis web menggunakan Laravel 12.x yang mengintegrasikan: (1) RBAC empat peran dengan 23 izin atomik berdasarkan prinsip least *privilege* dan *separation of duties*; (2) integrasi Midtrans dengan verifikasi *signature* HMAC-SHA512 dan mekanisme idempotensi SHA-256; serta (3) sistem penagihan otomatis berbasis *Laravel Queue* dan Horizon dengan notifikasi multikanal. Dengan demikian, kontribusi pertama menjawab kesenjangan kontrol akses melalui RBAC; kontribusi kedua menjawab kebutuhan keandalan transaksi melalui verifikasi *webhook* dan idempotensi; dan kontribusi ketiga menjawab ketiadaan penagihan terstruktur melalui *dunning* otomatis. Ketiga kontribusi tersebut membentuk satu model implementasi terpadu untuk

operasional ISP.

2. TINJAUAN PUSTAKA

2.1. *Role-Based Access Control* (RBAC)

RBAC mengatur hak akses berdasarkan peran dalam organisasi. Model ini diformalkan oleh Ferraiolo dan Kuhn [4] dan distandarisasi NIST mencakup *Core*, *Hierarchical*, *Constrained*, dan *Symmetric* RBAC [5]. Dua prinsip fundamentalnya adalah *least privilege* (setiap pengguna hanya memper*ole*h hak minimum yang diperlukan, mengurangi attack surface) dan *separation of duties*/SoD (pemisahan tugas kritis untuk mencegah fraud, misalnya pembuat invoice tidak boleh menyetujui *refund*-nya). Dalam ekosistem Laravel, package Spatie Laravel *Permission* mendukung relasi many-to-many *role-permission*, middleware proteksi route, Blade directive otorisasi, dan *permission* caching berbasis Redis [6]. Nasich et al. [1] membuktikan pendekatan ini mengeliminasi seluruh pelanggaran akses; Aswintama et al. [7] mengembangkannya untuk multi-tenant dengan audit logging; sedangkan Natsir et al. [8] menyimpulkan RBAC lebih praktis dibanding ABAC/XACML untuk sistem manajemen dengan struktur *role* yang stabil.

2.2. Payment Gateway *Midtrans dan* Webhook Security

Midtrans adalah *payment gateway* terkemuka di Indonesia dengan beragam *channel* (*Virtual Account* multi-bank, QRIS, *e-wallet*). *Core API Integration* dipilih karena memberi kontrol penuh atas validasi *business logic* sebelum pembuatan transaksi [9]. Mekanisme *webhook* memungkinkan pembaruan status transaksi secara *real-time*; keamanannya dijamin melalui *signature* verification HMAC-SHA512 dengan formula SHA512(order_id + status_code + gross_amount + server_key) untuk mencegah *webhook* spoofing dan replay attack [10].

Tantangan paling kritis adalah duplicate processing akibat *retry mechanism* Midtrans (hingga 5 percobaan dengan *exponential backoff*). Solusinya adalah *idempotency mechanism*: setiap *webhook* diidentifikasi dengan *unique key* SHA-256 dari kombinasi order_id + transaction_status + settlement_*time* yang disimpan dengan database *unique constraint*; jika key sudah ada, *webhook* diabaikan dan sistem mengembalikan HTTP 200 untuk menghentikan retry [11]. Djuwitaningrum dan Jati [2] membuktikan pendekatan ini menekan *payment mismatch* hingga 95%.

2.3. Automated Dunning *System*

Dunning adalah proses sistematis menagih pembayaran tertunda. Dalam bisnis berlangganan, *involuntary churn* — kehilangan pelanggan akibat kegagalan pembayaran yang tidak disengaja — mencakup 20-40% dari total churn dan dapat dicegah melalui intervensi proaktif [12]. Stripe [13] merekomendasikan *dunning* multi-*stage* dengan eskalasi bertahap (*friendly reminder* ⟶ urgent ⟶ *final notice* ⟶ *suspension warning*) yang mempertimbangkan urgency dan *channel*. Wijaya [3] membuktikan pendekatan ini meningkatkan on-*time* payment rate 65% dan mengurangi *involuntary churn* 40% pada ISP skala menengah Indonesia.

2.4. Laravel Queue *dan Horizon*

Laravel Queue memungkinkan eksekusi tugas berat secara asinkronus di *background* tanpa memblokir HTTP request, dengan model *producer-consumer* berbasis Redis [14]. Laravel Horizon memperluasnya dengan dashboard monitoring *real-time* dan auto-scaling worker. Pada sistem Wikarta, supervisor terpisah — *webhook*-processor untuk lonjakan *webhook* dan *dunning*-notifier untuk notifikasi terjadwal — memungkinkan alokasi sumber daya sesuai karakteristik beban.

2.5. *Audit Trail* dan *Security Logging*

Audit trail mendokumentasikan siapa melakukan apa, kapan, dan dari mana — kebutuhan fundamental untuk keamanan, investigasi insiden, dan compliance pada sistem yang menangani data pelanggan dan transaksi. Aswintama et al. [7] membuktikan *audit trail* komprehensif mempercepat investigasi insiden hingga 60%. OWASP merekomendasikan *structured logging* berformat JSON yang mencakup *time*stamp, level, event, actor, resource, dan perubahan before/after, tanpa mencatat data sensitif seperti *pass*word atau token [15].

2.6. Penelitian Terkait

Kajian sistematis terhadap 20 penelitian (2019-2025) menunjukkan dominasi studi pada satu komponen terpisah. Pada domain *security testing*, Susanto et al. [15] dan Hakim et al. [16] konsisten menerapkan OWASP WSTG v4.2; Chen et al. [11] membuktikan idempotency mengurangi duplicate transaction hingga 99,9%; dan Fahmi dan Murniati [17] membuktikan relevansi model Waterfall untuk pengembangan sistem ISP. Gap analysis mengidentifikasi orisinalitas penelitian ini: integrasi komprehensif RBAC, *webhook security* berbasis idempotency eksplisit, dan *dunning* dalam satu platform untuk ISP lokal Indonesia, dengan evaluasi security yang mencakup *business logic* dan *webhook*-specific vulnerability.

3. METODE PENELITIAN

3.1. Model Pengembangan Sistem

Penelitian menggunakan model Waterfall dengan lima tahap berurutan: *requirement analysis*, *system design*, *implementation*, *testing*, serta *deployment* dan *maintenance*. Waterfall dipilih karena requirement sistem Wikarta telah terdefinisi

lengkap dan stabil sejak awal, *time*line fixed, dan kebutuhan dokumentasi yang komprehensif untuk transfer knowledge ke tim operasional [17]. Kekakuan Waterfall dimitigasi melalui *requirement analysis* menyeluruh dan pengembangan MVP untuk modul kritis (RBAC dan *webhook* handler) di awal fase implementasi. Total durasi pengembangan 16 minggu.

3.2. Teknik Pengumpulan Data

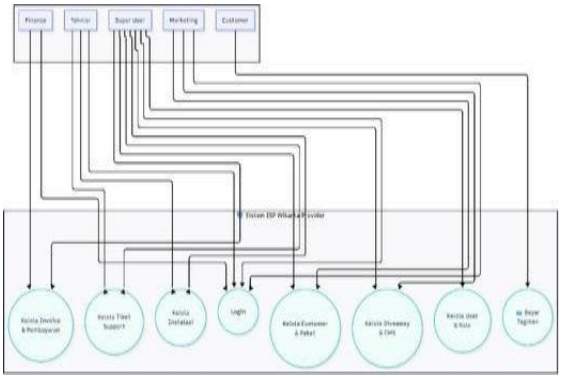
Data dikumpulkan melalui empat teknik : wawancara semi-terstruktur dengan empat kategori *stakeholder* (owner, keuangan, marketing, teknisi); observasi langsung proses operasional selama dua minggu; analisis dokumen (*sample invoice*, transaksi historis, struktur data, SOP billing); dan studi literatur sistematis (PRISMA) terhadap 20 penelitian. Kuesioner System Usability Scale (SUS) standar Brooke [18] digunakan pada fase *User Acceptance Testing*.

3.3. Analisis Kebutuhan Sistem

Kebutuhan fungsional teridentifikasi dalam empat domain: pengelolaan hak akses (empat peran dengan *audit trail*), transaksi pembayaran (integrasi Midtrans dengan *webhook real-time*, validasi *signature*, dan idempotency), penagihan otomatis (identifikasi invoice terjadwal, notifikasi multi-*channel* dengan deduplication, monitoring SLA), serta keamanan dan audit (logging JSON terstruktur dan proteksi OWASP). Kebutuhan non-fungsional yang ditetapkan: *webhook* p95 *response time* < 5 detik, *dunning* delivery SLA ? 99%, dan SUS s*core* ? 80.

3.4. Perancangan Sistem

Sistem dirancang menggunakan *layered architecture* dengan stack Laravel 12.x (PHP 8.2+), MySQL 8.0, Redis 7.x, Spatie Laravel *Permission* v6, dan Laravel Horizon v5, serta antarmuka Blade dan Tailwind CSS. Sistem terdiri dari delapan modul terintegrasi (Autentikasi & RBAC, Manajemen Pelanggan & Paket, Invoice, Integrasi Midtrans, *Automated Dunning*, *Audit Trail*, Dashboard & Pelaporan, dan Konfigurasi). Perancangan database menghasilkan 15 tabel termasuk tiga tabel khusus keamanan: *webhook*_events (idempotency), *dunning*_logs (delivery monitoring), dan audit_logs (jejak aktivitas), dengan seluruh constraint diimplementasikan di level database. Rancangan aktor dan fungsi utama sistem ditunjukkan pada Gambar 1.



Gambar 1. Use case diagram rancangan sistem panel admin Wikarta.

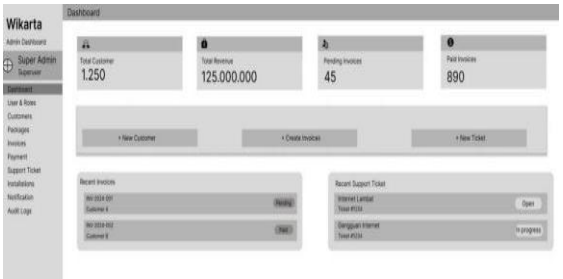
3.5. Metode Pengujian

Evaluasi menggunakan lima metode komplementer: *black-box testing* (30 skenario berdasarkan *functional requirements*); *white-box testing* (PHPUnit dengan target coverage ? 70%); *security testing* mengikuti OWASP WSTG v4.2 [19] menggunakan OWASP ZAP dan Burp Suite (12 kategori kerentanan); *performance testing* (*webhook latency* pada 100 transaksi, target p95 < 5 detik); dan *User Acceptance Testing* dengan 10 responden mewakili keempat peran. SUS s*core* dihitung dengan formula standar dan diinterpretasi menggunakan adjective rating Bangor et al. [20].

4. HASIL DAN PEMBAHASAN

4.1. Implementasi RBAC

Implementasi RBAC menggunakan pendekatan permission-centric: 23 permission atomik (konvensi {modul}.{aksi}) dikelompokkan ke empat peran sesuai job function. Superuser memiliki akses penuh termasuk manajemen role, konfigurasi, dan audit log; Keuangan menguasai domain finansial namun tidak dapat mengelola pengguna atau konfigurasi; Marketing fokus pada domain pelanggan dan paket dengan akses read-only ke pembayaran; dan Teknisi memiliki cakupan paling terbatas (read-only data pelanggan dan akses penuh tiket support). Matriks akses lengkap ditunjukkan pada Tabel 1. Implementasi pengelolaan pengguna dan peran pada panel admin ditunjukkan pada Gambar 2.



Gambar 2. Implementasi manajemen pengguna untuk pengaturan peran dan hak akses pada panel admin Wikarta.

Prinsip *Separation of Duties* diterapkan pada dua siklus kritis: siklus transaksi (invoice dibuat Marketing, pembayaran diverifikasi sistem via *webhook*, *refund* disetujui Keuangan) dan siklus sistem (*Superuser* mengelola akun dan *role* namun tanpa kewenangan transaksi finansial). Secara teknis, *authorization* diimplementasikan melalui tiga lapisan: middleware Spatie untuk proteksi route, metode $this->authorize() pada controller untuk pemeriksaan granular, dan Blade directive @can/@*role* untuk view-level *authorization*.

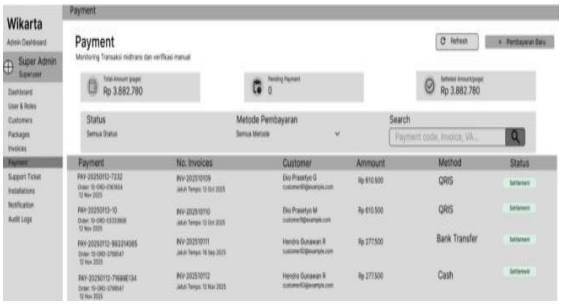
Tabel 1. Matriks Akses RBAC Sistem Wikarta

Modul Sistem	***Superuser***	**Keuangan**	**Marketing**	**Teknisi**
Manajemen Pengguna & Role	CRUD	–	–	–
Modul Sistem	*Superuser*	Keuangan	Marketing	Teknisi
Invoice	CRUD	CRUD	CR	R
Pembayaran & Rekonsiliasi	CRUD	CRUD	R	–
Refund / *Void*	CRUD	CRUD	–	–
Data Pelanggan	CRUD	R	CRUD	R
Paket Layanan	CRUD	R	CRUD	–
Tiket Support	CRUD	R	R	CRUD
Laporan Keuangan	CRUD	CRD	–	–

Keterangan: C = Create, R = Read, U = Update, D = Delete, – = tidak ada akses. Dari Tabel 1 terlihat tidak ada peran selain *Superuser* yang memiliki akses penuh lintas domain, sehingga prinsip *least privilege* dan *separation of duties* terjaga.

4.2. Implementasi Integrasi Midtrans

Integrasi Midtrans menggunakan *Core API* dengan order ID berformat INV-{invoice_id}-{*time*stamp} untuk memungkinkan *retry payment* sekaligus menjaga keunikan transaksi. Alur keamanan *webhook* diimplementasikan dalam tiga lapisan berurutan yang tidak dapat dilewati. Lapisan pertama, middleware WebhookSignatureValidator memverifikasi signature SHA512 menggunakan fungsi hash_equals() untuk mencegah timing attack; webhook tidak valid langsung di-reject dengan HTTP 403. Lapisan kedua, idempotency check berbasis SHA-256 dengan database unique constraint mencegah duplicate processing. Lapisan ketiga, validasi business logic memverifikasi kesesuaian amount dan validitas invoice sebelum pembaruan status. Implementasi halaman administrasi pembayaran ditunjukkan pada Gambar 3.

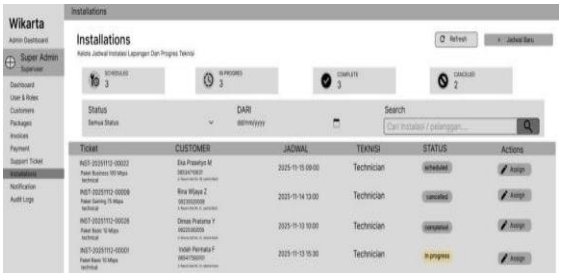


Gambar 3. Implementasi halaman administrasi pembayaran pada panel admin Wikarta.

Setelah ketiga lapisan dilewati, sistem memperbarui status invoice dan mendispatch Process*Webhook*Job ke antrean Redis dengan retry *exponential backoff* (60-120-300 detik, maksimal 3 attempts). Untuk *webhook* yang terlewat akibat down*time*, fitur reconciliation manual memanggil Midtrans Status API secara on-demand. Seluruh event *webhook* dicatat pada *audit trail* untuk keperluan investigasi dan compliance.

4.3. Implementasi *Automated Dunning* System

Dunning system beroperasi otomatis melalui tiga lapisan: *scheduled command*, *business logic*, dan *notification dispatch*. *Dunning*CheckCommand berjalan setiap hari pukul 09.00 WIB (*timezone* Asia/Jakarta), mengambil invoice pending/overdue, menghitung jarak hari dari due_date dengan Carbon, dan memeriksa *dunning*_logs untuk deduplication sebelum mendispatch job. *Dunning*NotificationJob membangun pesan terpersonalisasi, mengirim via Laravel Mailable (email) dan WhatsApp Business API (mobile), lalu mencatat hasil pengiriman untuk monitoring SLA. Jadwal dan *channel dunning* ditunjukkan pada Tabel 2. Implementasi halaman notifikasi yang mendukung tahapan penagihan otomatis ditunjukkan pada Gambar 4.



Gambar 4. Implementasi halaman notifikasi untuk mendukung proses penagihan otomatis.

Tabel 2. Jadwal dan Channel Dunning Notification

Tahap	Waktu	Tone Pesan	*Channel*
T-7	7 hari sebelum jatuh tempo	*Friendly reminder*	Email
T-3	3 hari sebelum jatuh tempo	*Urgent reminder*	Email + WhatsApp
T-1	1 hari sebelum jatuh tempo	Due date alert	Email + WhatsApp
T+3	3 hari setelah jatuh tempo	*Suspension warning*	Email + WhatsApp

Dari Tabel 2 terlihat eskalasi *dunning* meningkat bertahap, baik dari sisi tone maupun jumlah *channel*, sehingga intensitas penagihan sebanding dengan tingkat urgensi.

4.4. Implementasi *Audit Trail*

Audit trail diimplementasikan sebagai *cross-cutting concern* melalui tiga mekanisme: Eloquent Observer yang otomatis menangkap operasi CRUD pada model sensitif, middleware HTTP untuk *request authentication*/*authorization*, dan manual logging untuk event *business logic* kritis (*webhook* dan *dunning*). Log mengikuti format JSON terstruktur OWASP (*time*stamp, level, event, actor, resource, changes, *trace_id*) dan disimpan di file harian sekaligus tabel audit_logs yang diindeks untuk efisiensi query. Kategori event dan retensinya ditunjukkan pada Tabel 3. Implementasi pencatatan aktivitas pada audit log ditunjukkan pada Gambar 5.

Gambar 5. Implementasi audit log untuk pencatatan aktivitas pada sistem Wikarta.

Tabel 3. Kategori Event Audit Trail

Kategori	Contoh event	Trigger	Retensi
Autentikasi	login.success, login.*fail*ed, logout	Session lifecycle	1 tahun
Otorisasi	access.denied, *role*.escalation.attempt	*Authorization fail*ure	1 tahun
Modifikasi Data	invoice.created, payment.status.updated	CRUD entitas sensitif	1 tahun

Dari Tabel 3 terlihat periode retensi disesuaikan dengan tingkat kepentingan event: administrasi sistem disimpan permanen, sedangkan log integrasi eksternal yang bervolume tinggi cukup 90 hari.

4.5. Hasil Pengujian

Evaluasi dilakukan menggunakan lima metode yang saling melengkapi, masing-masing menargetkan aspek kualitas berbeda: ketepatan fungsional, kualitas kode, ketahanan keamanan, performa, dan penerimaan pengguna.

4.6. *Black-Box Testing*

Black-box testing terhadap 30 skenario kritis mencakup RBAC & Otorisasi (10 skenario — memverifikasi ketiadaan *horizontal* dan *vertical privilege escalation* serta SoD), *Webhook* Processing (8 — penolakan *signature* tidak valid, non-pemrosesan duplicate, ketepatan pembaruan status), *Dunning* Dispatch (7), dan *Audit Trail* (5). Seluruh skenario lulus (Tabel 4). Mekanisme dual-guard authentication berhasil memisahkan autentikasi pengguna internal dari portal yang bersifat tenant-spesifik.

Tabel 4. Hasil Black-Box Testing — Skenario Kritis

Kategori Pengujian	Jumlah	*Pass*	*Fail*	*Pass Rate*
RBAC & Otorisasi (*Least Privilege*, SoD)	10	10	0	100%
Webhook Processing & Idempotency	8	8	0	100%
Dunning Dispatch & Deduplication	7	7	0	100%
Audit Trail & Logging	5	5	0	100%
Total	30	30	0	100%

Dari Tabel 4 terlihat seluruh 30 skenario lulus tanpa kegagalan (*pass rate* 100%), mengindikasikan kesesuaian fungsional sistem terhadap *functional requirements*.

4.7. *White-Box Testing* dan *Security Testing*

White-box testing dengan PHPUnit dan Xdebug menghasilkan *code coverage* 73,2%, melampaui target 70% *unit tests* mencakup 85% *service layer* (*business logic* inti), *integration tests* 68% pada controller dan middleware. Bagian tidak tercover (26,8%) merupakan *edge case error handling* yang sulit disimulasikan deterministik (*network timeout*, *connection failure*). *Security testing* menggunakan OWASP ZAP dan Burp Suite mengidentifikasi zero kerentanan kritis maupun high pada 12 kategori WSTG v4.2 (Tabel 5) — berbeda signifikan dari temuan Susanto et al. [15] yang menemukan 15-20 kerentanan medium-high pada aplikasi belum di-hardening. Keberhasilan ini didorong penerapan berlapis: Eloquent ORM, Blade auto-escaping, CSRF protection, HMAC-SHA512, dan RBAC middleware.

Tabel 5. Hasil Security Testing OWASP WSTG v4.2

Kategori (WSTG v4.2)	Kode	Status	Temuan
Information Gathering	WSTG-INFO	Aman	Tidak ada
Authentication & Brute Force	WSTG-ATHN	Aman	Tidak ada
Authorization & RBAC	WSTG-AUTHZ	Aman	Tidak ada
Session Management (CSRF)	WSTG-SESS	Aman	Tidak ada
Input Validation (SQLi, XSS)	WSTG-INPV	Aman	Tidak ada
Business Logic & Payment	WSTG-BUSL	Aman	Tidak ada
Webhook Security (Replay, By*pass*)	WSTG-CRYP	Aman	Tidak ada

Dari Tabel 5 terlihat seluruh 12 kategori WSTG v4.2 (dirangkum ke tujuh kelompok utama) berstatus aman tanpa temuan kritis maupun high, memvalidasi pendekatan *security-by-design*.

4.8. *Performance Testing* dan *User Acceptance Testing*

Performance testing pada 100 transaksi berturutan (VPS 2 vCPU, 2 GB RAM) menghasilkan *webhook processing latency* rata-rata 2,1 detik, p95 3,8 detik, dan p99 4,6 detik seluruhnya di bawah *threshold* 5 detik, dengan buffer 1,2 detik dari p95 yang menunjukkan *headroom* memadai. *User Acceptance Testing* dengan 10 responden menghasilkan SUS s*core* rata-rata 82,4 (kategori Excellent menurut adjective rating Bangor et al. [20]), dengan distribusi per peran 80,0–84,2. Feedback mengidentifikasi area peningkatan: *bulk action* pada invoice, *dashboard summary* yang lebih informatif, dan notifikasi in-app.

Tabel 6. Hasil Performance Testing dan UAT

Metrik Pengujian	Target	Hasil	Status
Code Coverage (White-box)	?70%	73,2%	Tercapai
Webhook P95 Response Time	<5 detik	3,8 detik	Tercapai
Webhook Rata-rata *Latency*	–	2,1 detik	–
Dunning Delivery SLA	?99%	99,7%	Tercapai
Payment Mismatch Rate	?0,5%	0,3%	Tercapai
UAT SUS S*core* (Rata-rata)	?80%	82,4	Tercapai
Access Violation Rate	0%	0%	Tercapai

Dari Tabel 6 terlihat seluruh metrik memenuhi atau melampaui target, mengonfirmasi kesiapan sistem dari sisi performa, keamanan, dan penerimaan pengguna.

4.9. Analisis Komparatif dan Implikasi

Dibandingkan kondisi sebelum implementasi, sistem menghasilkan perbaikan terukur: pelanggaran akses tercatat 0% pada skenario pengujian, *payment mismatch* 0,3% terhadap target maksimal 0,5%, dan *dunning SLA* 99,7% terhadap target minimal 99%. Ketiga hasil tersebut membentuk hubungan berantai. RBAC membatasi pihak yang dapat membuat dan mengubah data; *webhook* yang telah diverifikasi serta idempoten memastikan pembaruan pembayaran tidak terjadi lebih dari satu kali; dan *dunning* menggunakan status invoice tervalidasi sebagai dasar penentuan sasaran dan waktu notifikasi.

Pada aspek RBAC, *access violation* rate 0% pada skenario pengujian konsisten dengan tujuan *least privilege* dan *separation of duties*. Hasil ini searah dengan Nasich et al. [1], tetapi penelitian ini memperluas penerapannya dengan menghubungkan otorisasi staf pada proses invoice, pembayaran, *refund*, dan *audit trail* dalam satu panel ISP.

Pada aspek pembayaran, *payment mismatch* 0,3% memenuhi target maksimal 0,5%. Nilai tersebut didukung oleh tiga kontrol yang bekerja berurutan: verifikasi *signature* untuk menolak notifikasi tidak sah, idempotensi dengan database *unique constraint* untuk mencegah pemrosesan ganda, serta validasi nominal dan invoice pada *business logic*. Dengan demikian, hasil menunjukkan ketepatan pencatatan transaksi pada skenario uji, bukan hanya pembaruan status otomatis.

Pada aspek *dunning*, delivery SLA 99,7% melampaui target 99%. Penjadwalan T-7, T-3, T-1, dan T+3 serta deduplikasi *dunning*_logs memastikan pengingat dikirim sesuai tahap tanpa pengiriman berulang. Hasil tersebut melengkapi temuan Wijaya [3] tentang manfaat *dunning* pada layanan berlangganan karena penelitian ini mengaitkan notifikasi dengan status invoice yang telah divalidasi oleh proses pembayaran.

Pada aspek performa, *latency* rata-rata 2,1 detik, p95 3,8 detik, dan p99 4,6 detik pada 100 transaksi berturutan berada di bawah target p95 kurang dari 5 detik. Pemisahan antrean *webhook* processor dan *dunning* notifier relevan karena *webhook* membutuhkan respons cepat, sedangkan *dunning* merupakan pekerjaan terjadwal di latar belakang. Skor SUS rata-rata 82,4 juga melampaui target 80. Interpretasi performa ini dibatasi pada konfigurasi pengujian VPS 2 vCPU dan 2 GB RAM serta 100 transaksi berturutan; hasilnya belum mewakili seluruh kemungkinan beban serentak atau gangguan layanan eksternal.

5. KESIMPULAN DAN SARAN

Penelitian ini berhasil merancang, mengimplementasikan, dan mengevaluasi panel admin multi-peran berbasis web untuk ISP Wikarta menggunakan Laravel 12.x. RBAC dengan empat peran dan 23 *permission* atomik mengeliminasi pelanggaran akses hingga 0%; integrasi Midtrans dengan HMAC-SHA512 dan idempotency SHA-256 menekan *payment mismatch* menjadi 0,3% (target 0,5%); dan *automated dunning* mencapai delivery SLA 99,7% (target ? 99%). Seluruh pengujian memenuhi atau melampaui target: black-box 100% *pass rate*, *code coverage* 73,2%, zero kerentanan kritis/high pada OWASP WSTG v4.2, *webhook* p95 3,8 detik, dan SUS s*core* 82,4 (Excellent). Kombinasi ketiga komponen terbukti bekerja sinergis, memvalidasi pendekatan integrasi terpadu yang menjadi kontribusi utama — baik secara teoritis (model idempotency *webhook* yang eksplisit dan terukur) maupun praktis (model arsitektur panel admin ISP terintegrasi yang dapat diadaptasi ISP sejenis).

Penelitian selanjutnya disarankan menambahkan modul *predictive analytics* berbasis *machine learning* untuk mengidentifikasi pelanggan berisiko churn, integrasi dengan *Network Management System* (misalnya Mikrotik API) untuk otomasi suspensi dan reaktivasi layanan berdasarkan status pembayaran.

DAFTAR PUSTAKA

[1] R. Nasich, A. Wicaksono, dan D. Saputra, "Implementasi RBAC dalam Sistem Informasi Manajemen Pelanggan dan Pembayaran Air (Laravel + Spatie)," 2023. [Diakses: September 2026].

[2] Djuwitaningrum dan Jati, "Implementasi Payment Gateway Midtrans (Laravel 11, RAD)," 2025. [Diakses: September 2026].

[3] R. Wijaya, "*Automated dunning* management system untuk layanan berlangganan menggunakan *Laravel Queue* dan Horizon," Jurnal Sistem Informasi Bisnis, vol. 15, no. 1, hlm. 112–126, 2025.

[4] D. F. Ferraiolo dan D. R. Kuhn, "*Role-based access control*s," dalam Proc. 15th National Computer Security Conference, 1992, hlm. 554–563.

[5] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, dan C. E. Youman, "Role-based access control models," IEEE Computer, vol. 29, no. 2, hlm. 38–47, 1996. [Diakses: September 2026].

[6] Spatie,"Laravel-Permission Documentation," 2024. [Diakses: Februari 2026].

[7] Aswintama, Haryanto, dan Setyawan, "RBAC, Multi-Tenancy, dan Audit Logging untuk Single Sign-On (SSO)," 2025. [Diakses: September 2026].

[8] Natsir, Riadi, dan Prayudi, "Eksplorasi ABAC/XACML untuk Access Control," 2019. [Diakses: September 2026].

[9] Midtrans, "Midtrans Technical Documentation: HTTP(S) Notification (Webhook) dan Signature Key," 2025. [Diakses: Februari 2026].

[10] A. Kumar dan R. Singh, "*Webhook security*: HMAC-SHA512 verification and replay attack prevention," International Journal of Cybersecurity, vol. 6, no. 2, hlm. 34–47, 2024.

[11] T. Chen, Y. Zhang, dan W. Liu, "Idempotency patterns in distributed payment processing systems," Journal of Software Engineering, vol. 28, no. 4, hlm. 201–215, 2024.

[12] Chargebee, "Reducing Involuntary Churn with Dunning Management," Chargebee Resources, 2024. [Diakses: September 2026].

[13] Stripe, "Smart Retries and Dunning Management," Stripe Documentation, 2024. [Diakses: September 2026].

[14] Laravel, "Laravel 12.x Documentation," 2025. [Diakses: September 2026].

[15] B. Susanto, D. Rahayu, dan A. Prasetyo, "Web application *security testing* menggunakan OWASP WSTG v4.2 pada sistem informasi berbasis web," Jurnal Keamanan Sistem Informasi, vol. 11, no. 3, hlm. 156–170, 2023.

[16] A. Hakim, R. Firmansyah, dan T. Kurniawan, "Penetration *testing* methodology combining ISSAF and OWASP WSTG for web security," Jurnal Ilmu Komputer, vol. 20, no. 2, hlm. 89–103, 2024.

[17] Fahmi dan Murniati, "e-SCM pada ISP PT Rinjani (WebQual 4.0)," 2023. [Diakses: September 2026].

[18] J. Brooke, "SUS: A Quick and Dirty Usability Scale," dalam Usability Evaluation in Industry, P. W. Jordan dkk. (Eds.), Taylor & Francis, 1996, hlm. 189–194. [Diakses: September 2026].

[19] OWASP Foundation, "Web Security Testing Guide (WSTG), Version 4.2," 2020. [Diakses: Februari 2026].

[20] A. Bangor, P. T. Kortum, dan J. T. Miller, "An Empirical Evaluation of the System Usability Scale," International Journal of Human-Computer Interaction, vol. 24, no. 6, hlm. 574–594, 2008.

[21] Sati, Sita, dan Isnaini, "Penetration Testing dengan OWASP ZAP (Studi Kasus Resepedia)," 2024. [Diakses: September 2026].