



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



**ANALISIS KEAMANAN *CALLBACK PAYMENT*
GATEWAY TERHADAP *REPLAY ATTACK* DAN
WEBHOOK SPOOFING MENGGUNAKAN
PENDEKATAN *DEFENSE-IN-DEPTH* BERBASIS
REDIS**

SKRIPSI

Jibrán Khairý Akram 2207421011

**POLITEKNIK
NEGERI
JAKARTA**

**PROGRAM STUDI TEKNIK MULTIMEDIA DAN
JARINGAN**

**JURUSAN TEKNIK INFORMATIKA DAN
KOMPUTER**

POLITEKNIK NEGERI JAKARTA

2026



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



**ANALISIS KEAMANAN *CALLBACK PAYMENT*
GATEWAY TERHADAP *REPLAY ATTACK* DAN
WEBHOOK SPOOFING MENGGUNAKAN
PENDEKATAN *DEFENSE-IN-DEPTH* BERBASIS
REDIS**

SKRIPSI

**Dibuat untuk Melengkapi Syarat – Syarat yang Diperlukan untuk
Memperoleh
Diploma Empat Politeknik**

**POLITEKNIK
NEGERI
JAKARTA**

Jibran Khairy Akram

2207421011

**PROGRAM STUDI TEKNIK MULTIMEDIA DAN
JARINGAN**

**JURUSAN TEKNIK INFORMATIKA DAN
KOMPUTER**

POLITEKNIK NEGERI JAKARTA

2026



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertanda tangan di bawah ini:

Nama Mahasiswa : Jibrán Khairy Akram
NIM : 2207421011
Jurusan/Program Studi : T.Informatika dan
Komputer/T.Multimedia dan Jaringan
Judul Skripsi : Analisis Keamanan *Callback Payment*
Gateway Terhadap Replay Attack dan Webhook Spoofing Menggunakan
Pendekatan *Defense-in-Depth* Berbasis Redis

Menyatakan dengan sebenarnya bahwa skripsi ini benar-benar merupakan hasil karya saya sendiri, bebas dari peniruan terhadap karya dari orang lain. Kutipan pendapat dan tulisan orang lain ditunjuk sesuai dengan cara-cara penulisan karya ilmiah yang berlaku.

Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa dalam skripsi ini terkandung ciri-ciri plagiat dan bentuk-bentuk peniruan lain yang dianggap melanggar peraturan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Depok 9 Juni, 2026,

Yang membuat pernyataan



Jibrán Khairy Akram

NIM. 2207421011



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Skripsi diajukan oleh:

Nama : Jibrán Khairy Akram
NIM : 2207421011
Program Studi : Teknik Multimedia dan Jaringan
Judul Skripsi : Analisis Keamanan *Callback Payment Gateway*
Terhadap *Replay Attack* dan *Webhook Spoofing* Menggunakan Pendekatan
Defense-in-Depth

Telah diuji oleh tim penguji dalam Sidang Skripsi pada hari ... , Tanggal ... ,
Bulan ... , Tahun 2026 dan dinyatakan LULUS.

Disahkan oleh:

Pembimbing I	Ayu Rosyida Zain, S.ST., M.T.
Penguji I	Asep Kurniawan, S.Pd., M.Kom.
Penguji II	Fachroni Arbi Murad, S.Kom., M.Kom.
Penguji III	lik Muhamad Malik Matin, S.Kom., M.T.

(
(
(
(

POLITEKNIK
NEGERI
JAKARTA

Mengetahui:

Jurusan Teknik Informatika dan Komputer

Ketua



Dr. Anita Hidayati, S.Kom., M.Kom.

NIP. 197908032003122003



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT atas limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi berjudul "*Analisis Keamanan Callback Payment Gateway Terhadap Replay Attack dan Webhook Spoofing Menggunakan Pendekatan Defense-in-Depth Berbasis Redis*". Skripsi ini disusun untuk memenuhi salah satu syarat memperoleh gelar Diploma IV pada Program Studi Teknik Multimedia dan Jaringan, Jurusan Teknik Informatika dan Komputer, Politeknik Negeri Jakarta.

Penulis menyadari bahwa penyusunan skripsi ini tidak lepas dari bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Orang tua dan seluruh keluarga tercinta, atas kasih sayang, doa, dan dukungan moril agar proses penyusunan skripsi penulis berjalan lancar, serta memberikan dukungan dalam menyelesaikan skripsi.
2. Ibu Ayu Rosyida Zain, S.ST., M.T., sebagai dosen pembimbing yang telah meluangkan waktu, membantu, mendukung dan memberikan masukan serta saran kepada penulis selama pengerjaan skripsi hingga selesai.
3. Bapak dan Ibu Dosen Jurusan Teknik Informatika dan Komputer, khususnya program studi Teknik Multimedia dan Jaringan, atas ilmu, motivasi, dan inspirasi yang telah diberikan.
4. Teman-teman seperjuangan penulis: Sahrul, Vito, Syifa, Raden, Abdan, Stepani, Audi, Rizki, Wibi, Dhea, Yoga, Rafli, dan Darma, serta rekan-rekan seprodi lainnya yang selalu memberikan dukungan, semangat, dan kebersamaan dalam suka maupun duka selama masa perkuliahan hingga skripsi ini selesai.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan demi perbaikan di masa mendatang. Semoga laporan ini dapat memberikan manfaat bagi pembaca dan pihak-pihak yang berkepentingan.



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

SURAT PERNYATAAN PERSETUJUAN PUBLIKASI
SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Politeknik Negeri Jakarta, saya bertanda tangan di bawah ini :

Nama Mahasiswa : Jibran Khairy Akram

NIM : 2207421011

Jurusan/Program Studi : T.Informatika dan Komputer/T.Multimedia dan Jaringan

Demi pengembangan ilmu pengetahuan , menyetujui untuk memberikan kepada Politeknik Negeri Jakarta Hak Bebas Royalti Non-Eksklusif atas karya ilmiah saya yang berjudul :

Analisis Keamanan *Callback Payment Gateway* Terhadap *Replay Attack* dan *Webhook Spoofing* Menggunakan Pendekatan *Defense-in- Depth* Berbasis Redis

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-Eksklusif ini Politeknik Negeri Jakarta Berhak menyimpan, mengalihmediakan/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Depok 9 Juni, 2026,

Yang n



Jibran Khairy Akram

NIM. 2207421011



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Analisis Keamanan *Callback Payment Gateway* Terhadap *Replay Attack* dan *Webhook Spoofing* Menggunakan Pendekatan *Defense-in- Depth* Berbasis Redis

ABSTRAK

Pertumbuhan transaksi digital memunculkan kerentanan pada arsitektur komunikasi asinkron, khususnya pada integrasi *Payment Gateway*. Permasalahan utama pada *Website PsikoTest* adalah rentannya jalur transmisi *callback* terhadap manipulasi logika bisnis (*Webhook Spoofing*) dan penggandaan transaksi akibat serangan serentak (*Replay Attack*) yang memicu kerugian finansial. Penelitian ini bertujuan menganalisis kerentanan tersebut dan mengimplementasikan arsitektur keamanan berlapis (*Defense-in-Depth*). Sistem dirancang menggunakan kriptografi *HMAC-SHA256* pada lapisan jaringan, mekanisme *Idempotency Key* dan *Rate Limiter* berbasis *Redis* pada *middleware*, serta *Database Absolute Lock* pada persistensi. Pengujian dieksekusi menggunakan *Apache JMeter*. Hasil pengujian menunjukkan arsitektur berhasil mengisolasi 100% anomali muatan dan memblokir *Webhook Spoofing* dengan rata-rata latensi 65,2 ms. Pada simulasi konkurensi tinggi, sistem mencegah 99,00% muatan duplikat di tingkat memori dan memblokir 100% eksploitasi kedaluwarsa waktu di basis data. Pengujian beban puncak 10.000 threads mengonfirmasi sistem secure menekan latensi hingga 60,6% (5.369 ms) dibandingkan sistem rentan (13.632 ms), menjaga *throughput* pada 94,1 req/s, serta meredam *Load Average* di angka 1,67. Arsitektur ini terbukti beroperasi secara komplementer untuk mengamankan integritas transaksi dan mencegah kelumpuhan layanan.

Kata Kunci: *Defense-in-Depth, HMAC-SHA256, Idempotency, Payment Gateway, Redis, Replay Attack, Webhook Spoofing.*



DAFTAR ISI

PERATURAN PERNYATAAN BEBAS PLAGIARISME.....	II
LEMBAR PENGESAHAN	ERROR! BOOKMARK NOT DEFINED.
KATA PENGANTAR	IV
PERATURAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPEMILIKAN AKADEMIS	V
ABSTRAK	VI
DAFTAR ISI	VII
DAFTAR TABEL	X
DAFTAR GAMBAR.....	XI
BAB I PENDAHULUAN.....	1
1.1 LATAR BELAKANG.....	1
1.2 PERUMUSAN MASALAH	3
1.3 BATASAN MASALAH.....	4
1.4 TUJUAN DAN MANFAAT	5
1.4.1 Tujuan	5
1.4.2 Manfaat	6
1.5 SISTEMATIKA PENULISAN.....	6
BAB II TINJAUAN PUSTAKA.....	8
2.1 PENELITIAN TERKAIT.....	8
2.2 PAYMENT GATEWAY	12
2.3 WEBHOOK	13
2.4 REPLAY ATTACK (SERANGAN PENGULANGAN)	13
2.5 WEBHOOK SPOOFING (PEMALSUAN IDENTITAS)	13
2.6 RACE CONDITION (KONDISI BALAPAN DATA)	14
2.7 HMAC-SHA256.....	14
2.8 IDEMPOTENCY KEY.....	14
2.10 MySQL.....	15
2.11 BURP SUITE	15

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2.12	APACHE JMeter	15
2.13	METRIK PENGUJIAN SISTEM	16
BAB III METODE PENELITIAN.....		18
3.1	RANCANGAN PENELITIAN	18
3.2	TAHAPAN PENELITIAN	19
3.3	OBJEK PENELITIAN	21
BAB IV HASIL DAN PEMBAHASAN		23
4.1	ANALISIS KEBUTUHAN	23
4.2	PERANCANGAN SISTEM	25
4.2.1	Diagram Blok Sistem.....	25
4.2.2	Flowchart Sistem	26
4.2.3	Perancangan Basis Data.....	27
4.2.4	Topologi Jaringan Sistem	29
4.2.5	Konfigurasi Jaringan Publik dan Firewall	30
4.3	IMPLEMENTASI SISTEM.....	32
4.3.1	Inisialisasi Virtual Private Server (VPS) dan Manajemen Panel.....	32
4.3.2	Instalasi Environment Node.js.....	34
4.3.3	Instalasi dan Konfigurasi Redis Server.....	35
4.3.4	Konfigurasi Basis Data dan Kredensial Koneksi.....	38
4.3.5	Konfigurasi Endpoint pada Payment Gateway Tripay	41
4.3.6	Implementasi Verifikasi HMAC SHA-256	43
4.3.7	Implementasi Idempotency Key dan Redis Mutex Lock.....	46
4.3.8	Integrasi Keamanan pada Endpoint Webhook Callback	49
4.3.9	Deployment dan Inisialisasi Server Aplikasi.....	55
4.4	PENGUJIAN	59
4.4.1	Deskripsi Pengujian	59
4.4.2	Prosedur Pengujian	61
4.4.3	Data Hasil Pengujian	93
4.4.4	Analisis Data Pengujian.....	110
BAB V PENUTUP.....		123
5.1	KESIMPULAN	123



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

5.2	SARAN.....	124
	DAFTAR PUSTAKA	126
	DAFTAR RIWAYAT HIDUP.....	129
	LAMPIRAN	131





DAFTAR TABEL

Tabel 2. 1 Penelitian Terdahulu	8
Tabel 4. 1 Analisis Kebutuhan Perangkat	23
Tabel 4. 2 Konfigurasi DNS (Domain Name System).....	30
Tabel 4. 3 Aturan Firewall (Inbound Rules)	31
Tabel 4. 4 Perbandingan Arsitektur API Callback Payment Gateway.....	60
Tabel 4. 5 Spesifikasi Virtual Machine.....	61
Tabel 4. 6 Konfigurasi Variasi Manipulasi Panjang Signature.....	66
Tabel 4. 7 Skenario dan Parameter Beban Pengujian	84
Tabel 4. 8 Data Hasil Pengamatan Kerentanan Manipulasi Panjang Signature	94
Tabel 4. 9 Komparasi Waktu Respons Pemrosesan Anomali.....	98
Tabel 4. 10 Data Hasil Pengamatan Simulasi Manipulasi (Webhook Spoofing) ..	99
Tabel 4. 11 Komparasi Waktu Respons pada Skenario Webhook Spoofing	102
Tabel 4. 12 Data Hasil Komparasi Fase Pertama (Beban 100 Threads < 60 Detik)	104
Tabel 4. 13 Data Hasil Pengamatan Fase Kedua (Beban 100 Threads > 60 Detik)	106
Tabel 4. 14 Komparasi Hasil Pengujian Ketahanan Beban (Fase Ketiga).....	107

**POLITEKNIK
NEGERI
JAKARTA**

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumpulkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



DAFTAR GAMBAR

Gambar 4. 1 Diagram Blok Sistem	25
Gambar 4. 2 Flowchart Sistem.....	26
Gambar 4. 3 Entity Relationship Diagram (ERD) Modul	27
Gambar 4. 4 Entity Relationship Diagram (ERD) Modul	28
Gambar 4. 5 Topologi Jaringan Sistem.....	29
Gambar 4. 6 Akses Remote SSH ke VPS Ubuntu	32
Gambar 4. 7 Antarmuka Dashboard Manajemen aaPanel	33
Gambar 4. 8 Pengecekan Versi Node.js dan NPM pada Server	34
Gambar 4. 9 Instalasi dan Verifikasi Package Manager pnpm	35
Gambar 4. 10 Instalasi Redis Server melalui Manajemen Panel	36
Gambar 4. 11 Antarmuka Manajemen dan Pemantauan Redis.....	37
Gambar 4. 12 Verifikasi Responsivitas Redis Server	37
Gambar 4. 13 Antarmuka Manajemen Basis Data MySQL pada aaPanel.....	38
Gambar 4. 14 Konfigurasi Kredensial Basis Data pada Variabel Lingkungan (.env)	39
Gambar 4. 15 Implementasi Fisik Tabel Payment pada phpMyAdmin.....	40
Gambar 4. 16 Konfigurasi Kredensial API pada Dasbor Tripay	41
Gambar 4. 17 Konfigurasi URL Callback.....	42
Gambar 4. 18 Sinkronisasi Kredensial Tripay pada Variabel Lingkungan (.env).....	42
Gambar 4. 19 Deklarasi Struktur Data Payload	44
Gambar 4. 20 Implementasi Fungsi Keamanan Inti dan Mitigasi Timing Attack	44
Gambar 4. 21 Implementasi Validasi Jalur Utama (HTTP Header Signature).....	45
Gambar 4. 22 Implementasi Jalur Cadangan dan Kebijakan Default Deny	46
Gambar 4. 23 Inisialisasi Koneksi ke Server Redis	47
Gambar 4. 24 Implementasi Payload Fingerprint dan Atomic Operation	48
Gambar 4. 25 Mekanisme Forensik dan Mitigasi Hybrid Attack	49
Gambar 4. 26 Implementasi Gerbang Layer 0 (Rate Limiting) pada Webhook	50
Gambar 4. 27 Implementasi Gerbang Validasi Kriptografi HMAC pada Webhook	51
Gambar 4. 28 Implementasi Gerbang Validasi Idempotency Lock pada Webhook	52

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Gambar 4. 29 Implementasi Gerbang Layer 3 (Database Absolute Lock) pada Webhook	52
Gambar 4. 30 Implementasi Eksekusi Pembaruan Basis Data Persisten	54
Gambar 4. 31 Implementasi Pencetakan Akses Token dan Penyelesaian Siklus ..	55
Gambar 4. 32 Kompilasi Aplikasi Next.js pada Lingkungan Server	56
Gambar 4. 33 Inisialisasi Layanan Node.js dan Manajemen Port	57
Gambar 4. 34 Konfigurasi Transport Layer Security (SSL/TLS) dan Force HTTPS	58
Gambar 4. 35 Tampilan Respons HTTP 200 OK pada Sistem Rentan	68
Gambar 4. 36 Perubahan Status Transaksi Menjadi Success secara Ilegal.....	68
Gambar 4. 37 Log Terminal Server Menunjukkan Pencetakan Token Akses Ilegal	69
Gambar 4. 38 Tampilan Respons HTTP 400 Bad Request pada Sistem Secure ...	69
Gambar 4. 39 Log Terminal Sistem Secure Mendeteksi dan Memblokir Serangan	70
Gambar 4. 40 Tampilan Respons HTTP 200 OK pada Sistem Rentan (Karakter Berlebih).....	71
Gambar 4. 41 Perubahan Status Transaksi Menjadi Success secara Ilegal.....	71
Gambar 4. 42 Log Terminal Server Menunjukkan Pencetakan Token Akses Ilegal	72
Gambar 4. 43 Penolakan Serangan Karakter Berlebih pada Sistem Secure	72
Gambar 4. 44 Log Terminal Sistem Secure Mendeteksi Anomali Memori	73
Gambar 4. 45 Tampilan Respons HTTP 200 OK pada Sistem Rentan (Injeksi Angka Mentah).....	74
Gambar 4. 46 Perubahan Status Transaksi Menjadi Success secara Ilegal.....	74
Gambar 4. 47 Log Terminal Server Menunjukkan Pencetakan Token Akses Ilegal	74
Gambar 4. 48 Penolakan Serangan Injeksi Angka Mentah pada Sistem Secure ...	75
Gambar 4. 49 Log Terminal Sistem Secure Mendeteksi Anomali Format Data ...	75
Gambar 4. 50 Tampilan Respons HTTP 200 OK pada Sistem Rentan (Injeksi Simbol).....	76
Gambar 4. 51 Perubahan Status Transaksi Menjadi Success secara Ilegal.....	77



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Gambar 4. 52 Log Terminal Server Menunjukkan Pencetakan Token Akses Ilegal	77
Gambar 4. 53 Penolakan Serangan Injeksi Simbol pada Sistem Secure.....	78
Gambar 4. 54 Log Terminal Sistem Secure Mendeteksi Anomali Karakter Ilegal	78
Gambar 4. 55 Tampilan Respons HTTP 200 OK pada Sistem Rentan (Webhook spoofing).....	80
Gambar 4. 56 Perubahan Status Transaksi Menjadi Success secara Ilegal.....	80
Gambar 4. 57 Log Terminal Server Menunjukkan Pencetakan Token Akses Ilegal	81
Gambar 4. 58 Status Transaksi pada Antarmuka Payment Gateway	82
Gambar 4. 59 Penolakan Serangan Webhook Spoofing pada Sistem Secure.....	82
Gambar 4. 60 Log Terminal Sistem Secure Mendeteksi Modifikasi Muatan Data	83
Gambar 4. 61 Tampilan Respons Sistem Rentan terhadap Serangan High Concurrency	85
Gambar 4. 62 Log Terminal Server Menunjukkan Pemrosesan Ganda	86
Gambar 4. 63 Anomali Pencetakan Hak Akses Ganda pada Basis Data	86
Gambar 4. 64 Penolakan Race Condition secara Presisi pada Lapisan Redis	87
Gambar 4. 65 Pencatatan Idempotency Key dan Mitigasi Lonjakan Beban pada Redis.....	88
Gambar 4. 66 Jejak Audit (Audit Trail) Penolakan Payload Duplikat	89
Gambar 4. 67 Penolakan Serangan Jeda Waktu (TTL Expiry) pada Lapisan Database	90
Gambar 4. 68 Ilusi Ketahanan dan Bencana Resource Exhaustion pada Sistem Rentan	91
Gambar 4. 69 Stabilitas Pemulihan Layanan dan Pencegahan Efek DoS.....	91
Gambar 4. 70 Deteksi dan Interupsi Serangan Masif pada Terminal Server	92
Gambar 4. 71 Komparasi Waktu Respons (Buffer Mismatch)	111
Gambar 4. 72 Komparasi Waktu Respons pada Pengujian Webhook Spoofing .	113
Gambar 4. 73 Komparasi Tren Latensi pada Beban Serentak 100 Threads	115
Gambar 4. 74 Komparasi Waktu Respons (Latensi) pada Uji Ketahanan Beban	118



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Gambar 4. 75 Komparasi Kapasitas Penyelesaian (Throughput) pada Uji Ketahanan Beban	119
Gambar 4. 76 Komparasi Utilisasi Sumber Daya Server (Resource Exhaustion)	120
Gambar 4. 77 Distribusi Pemblokiran Serangan.....	121





Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

BAB I PENDAHULUAN

1 Latar Belakang

Tren ancaman siber global pada sektor keuangan kini telah berevolusi dari serangan infrastruktur tradisional menuju serangan terhadap logika aplikasi (*Business Logic Abuse*), khususnya pada antarmuka pemrograman aplikasi (API). Laporan terbaru dari Oryx Security (2025) mencatat bahwa 37% dari total masalah keamanan yang ditemukan pada API produksi adalah kategori kerentanan (*vulnerabilities*), yang membuka celah kritis bagi eksploitasi teknis dan manipulasi alur logika bisnis (*Business Logic Abuse*) untuk keuntungan finansial. Urgensi ini dipertegas oleh laporan Akamai (2024) yang menempatkan sektor jasa keuangan (*Financial Services*) sebagai target utama serangan global. Eskalasi ancaman ini diperburuk oleh serangan *Spoofing* yang menjadi risiko kritis pada API Fintech karena mengeksploitasi kelemahan autentikasi dengan menyamar sebagai entitas sah (Ranjan et al., 2022). Selain itu, teridentifikasi pula persistensi serangan *Replay Attack* yang mampu mengeksploitasi transmisi data valid namun minim verifikasi waktu atau *timestamp* (Obaidat et al., 2024). Kerentanan manipulasi transaksi ini terbukti fatal pada *Merchant Payment Platform*, di mana Al-Omari (2025) menyoroti bahwa celah tersebut sering dieksploitasi untuk merugikan integritas keuangan. Kondisi ini menjadikan pengamanan pada jalur komunikasi pembayaran digital (*Payment Gateway*) sangat krusial, mengingat serangan pada logika bisnis seringkali lolos dari deteksi keamanan standar karena lalu lintas datanya terlihat valid secara teknis namun manipulatif secara konteks.

Relevansi ancaman tersebut teramati secara nyata pada studi kasus Website PsikoTest yang dikelola oleh PT Sudami Jaya Medika. Berdasarkan hasil observasi dan analisis kebutuhan sistem, proses pembayaran token tes psikologi sebelumnya masih dilakukan melalui mekanisme konvensional yang mewajibkan pengguna mengirimkan bukti transfer secara manual. Mekanisme ini teridentifikasi memiliki kelemahan signifikan berupa inefisiensi operasional akibat antrean verifikasi yang menghambat kecepatan layanan. Untuk mengatasi kendala



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

kalabilitas tersebut, sistem beralih mengadopsi pembayaran otomatis (*cashless*)

berbasis *Webhook*. Namun, transisi teknologi ini justru membuka permukaan serangan (*attack surface*) baru yang jauh lebih kompleks. Tanpa adanya validasi kriptografis yang memadai, sistem otomatis ini menjadi sangat rentan terhadap manipulasi alur data, khususnya ancaman *Webhook Spoofing* dan *Replay Attack* yang dapat memvalidasi transaksi ilegal secara *real-time* tanpa interaksi admin.

Untuk mengatasi ancaman *Webhook Spoofing* dan *Replay Attack*, berbagai penelitian telah mengusulkan mekanisme keamanan berbasis verifikasi integritas data dan validasi waktu transaksi. Salah satu pendekatan yang banyak digunakan adalah algoritma HMAC-SHA256 yang terbukti efektif dalam menjamin keaslian dan integritas data yang dikirimkan melalui API. Selain itu, validasi berbasis timestamp juga umum diterapkan untuk membatasi rentang waktu validitas suatu permintaan sehingga dapat mengurangi risiko serangan *Replay Attack*.

Meskipun demikian, implementasi mekanisme tersebut masih memiliki sejumlah keterbatasan. Penggunaan HMAC-SHA256 sering kali menghadapi kendala latensi ketika proses verifikasi masih bergantung pada basis data relasional (Hendra et al., 2025). Di sisi lain, strategi validasi berbasis timestamp dinilai belum sepenuhnya efektif dalam menangani duplikasi event secara presisi pada kondisi jaringan yang tidak stabil (Wang, 2024). Selain itu, pemanfaatan teknologi Redis pada penelitian terdahulu masih lebih banyak difokuskan pada optimasi performa sistem dan belum dieksplorasi secara maksimal sebagai lapisan keamanan (*security layer*) untuk mencegah serangan berbasis logika aplikasi (Joshi, 2024). Keterbatasan tersebut menunjukkan masih adanya kesenjangan penelitian dalam pengembangan mekanisme pertahanan yang tidak hanya aman, tetapi juga mampu merespons ancaman secara *real-time* dengan tetap mempertahankan performa sistem.

Untuk menjawab tantangan tersebut, penelitian ini mengusulkan strategi mitigasi *Defense-in-Depth* yang mengintegrasikan dua lapisan pertahanan vital. Lapisan pertama menggunakan verifikasi HMAC-SHA256 sebagai standar otentikasi data, sedangkan lapisan kedua diperkuat dengan mekanisme *Idempotency Key* berbasis Redis. Mekanisme ini berfungsi sebagai penanda unik untuk mendeteksi dan



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

memblokir eksekusi ulang pada transaksi yang sama secara otomatis. Fungsi validasi ini menjadi krusial karena arsitektur asinkron rentan terhadap anomali *Race Condition*, yang terbukti menyebabkan 54% pelanggaran semantik serius berupa pencatatan ganda (*Duplicate Entries*) yang merugikan integritas keuangan (Qiu et al., 2022). Risiko ini diperparah oleh *bottleneck* basis data relasional yang memiliki waktu respons lambat hingga 980 ms (Iurchenko, 2025). Sebaliknya, pemanfaatan operasi atomik pada Redis terbukti mampu memangkas waktu respons signifikan menjadi 78 ms dan meningkatkan *throughput* sebesar 40% (Joshi, 2024), sehingga efektif memitigasi serangan duplikasi secara *real-time* sekaligus menutupi kelemahan strategi terdahulu.

Berdasarkan akumulasi kerentanan protokol dan risiko inefisiensi basis data tersebut, penelitian ini bertujuan untuk menganalisis efektivitas strategi *Defense-in-Depth* dalam memitigasi serangan *Replay Attack* dan *Webhook Spoofing* pada studi kasus Website PsikoTest. Urgensi ini didorong oleh risiko finansial nyata berupa *Double Spending* akibat mekanisme *retry* otomatis dari *Payment Gateway* yang gagal ditangani oleh validasi konvensional. Melalui integrasi verifikasi HMAC-SHA256 dan mekanisme *Idempotency Key* berbasis Redis, penelitian ini diharapkan dapat menghadirkan solusi keamanan yang tidak hanya menjamin integritas transaksi secara presisi, tetapi juga mempertahankan performa sistem yang tinggi sesuai tuntutan ekosistem pembayaran digital modern.

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka perumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana menerapkan mekanisme pertahanan berlapis (*defense-in- depth*) yang mengintegrasikan validasi *Idempotency Key* berbasis Redis dan Verifikasi HMAC-SHA256 pada modul *callback* sistem eksisting *Payment Gateway* di Website PsikoTest?
2. Bagaimana hasil analisis efektivitas mitigasi serangan *Replay Attack* dan *Webhook Spoofing* setelah penerapan mekanisme pertahanan berlapis



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

(*defense-in-depth*) tersebut dibandingkan dengan kondisi sistem sebelumnya yang hanya mengandalkan validasi basis data konvensional?

Bagaimana dampak penggunaan penyimpanan data berbasis *memory* (*In-Memory Data Store*) Redis terhadap kinerja sistem (latensi dan throughput) saat menghadapi kondisi *high concurrency*?

3. Batasan Masalah

Penelitian ini difokuskan pada analisis keamanan untuk memitigasi serangan *Replay Attack* dan *Webhook Spoofing* pada *callback Payment Gateway Website* sikoTest, dengan menerapkan pendekatan *Defense-in-Depth* yang mengintegrasikan verifikasi HMAC-SHA256 dan mekanisme *Idempotency Key* berbasis Redis. Adapun batasan masalah dalam penelitian ini adalah:

1. Analisis terbatas pada dua jenis serangan spesifik, yaitu *Replay Attack* (pengiriman ulang paket transaksi valid) dan *Webhook Spoofing* (pemalsuan identitas server pengirim), dengan target operasi pada *endpoint callback* pembayaran.
2. Solusi keamanan diterapkan menggunakan pendekatan *Defense-in-Depth* yang hanya mencakup dua lapisan pertahanan: verifikasi validitas pengirim menggunakan HMAC-SHA256 dan pencegahan duplikasi data menggunakan *Idempotency Key* berbasis Redis.
3. Lingkungan pengujian diimplementasikan pada layanan awan *Virtual Private Server* (VPS) bersistem operasi Ubuntu Server yang telah dilengkapi dengan nama domain dan sertifikat SSL/HTTPS terenkripsi. Simulasi *callback* pembayaran dikirimkan langsung dari simulator Tripay menuju *endpoint* publik VPS.
4. Perangkat lunak (*tools*) yang digunakan meliputi Postman untuk validasi fungsional API, Burp Suite untuk simulasi serangan penetrasi (*Replay Attack* dan *Spoofing*), serta Apache JMeter untuk pengujian beban (*stress test*) dan konkurensi sistem.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Sistem yang diuji dan diimplementasikan berbasis Node.js (Next.js) dengan basis data MySQL dan Redis, tidak mencakup perbandingan dengan basis data NoSQL atau bahasa pemrograman lain.

Konfigurasi Redis Difokuskan pada penggunaan Operasi Atomik (*Atomic Operations*) dan manajemen *Time-To-Live* (TTL) agar kunci idempotensi kadaluarsa secara otomatis demi efisiensi *memory*.

Analisis efektivitas sistem dibatasi pada parameter keberhasilan pencegahan *double spending*, deteksi *payload* palsu, serta kinerja sistem (*latency* dan *throughput*), tanpa melakukan analisis kriptanalisis mendalam terhadap algoritma SHA-256.

1.4 Tujuan dan Manfaat

1.4.1 Tujuan

Adapun Tujuan penelitian ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan mekanisme pertahanan berlapis (*defense-in-depth*) yang mengintegrasikan *Idempotency Key* berbasis Redis dan verifikasi HMAC-SHA256 pada arsitektur *callback* Payment Gateway di Website PsikoTest.
2. Menganalisis efektivitas sistem dalam memitigasi serangan *Replay Attack* dan *Webhook Spoofing* dengan membandingkan tingkat keberhasilan serangan pada sistem yang diusulkan terhadap sistem validasi basis data konvensional.
3. Mengevaluasi dampak penggunaan penyimpanan data berbasis memori (*In-Memory Data Store*) Redis terhadap performa sistem, khususnya pada parameter latensi (*response time*) dan *throughput* saat menangani lalu lintas transaksi yang tinggi (*high concurrency*).



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4.2 Manfaat

Adapun manfaat penelitian ini adalah sebagai berikut:

1. Memberikan kontribusi terhadap pengembangan studi di bidang keamanan web (*Web Security*), khususnya dalam implementasi strategi *Defense in Depth* untuk mengamankan arsitektur *Event- Driven* dan *Webhook* pada sistem pembayaran digital.
2. Menawarkan solusi keamanan komprehensif yang mampu menutup celah kerentanan logika bisnis (*Business Logic Abuse*) seperti manipulasi saldo ganda (*Double Spending*) tanpa mengorbankan kecepatan pemrosesan
3. Memungkinkan pengembang sistem untuk menerapkan proteksi integritas data tingkat tinggi yang efisien menggunakan perangkat lunak *open source* (Redis), tanpa bergantung pada peningkatan spesifikasi perangkat keras basis data yang memakan biaya tinggi.
4. Menjadi referensi dalam pengembangan penelitian lanjutan terkait optimasi keamanan infrastruktur *Microservices* dan integrasi pola *Idempotency* pada sistem transaksi *real-time*.
5. Memberikan panduan implementatif bagi praktisi IT dan *Backend Developer* dalam membangun mekanisme validasi *callback* yang efisien dan resisten terhadap ancaman pengulangan paket data.
6. Memberikan dasar bagi Website PsikoTest (Klinik Yuliarpan Medika) untuk meningkatkan standar keamanan operasional pada layanan transaksi digital, sehingga integritas data keuangan perusahaan dan kepercayaan pengguna dapat terjaga.

1.5 Sistematika Penulisan

Sistematika penulisan dalam penelitian ini terdiri dari 5 bab, yaitu:

1. BAB I PENDAHULUAN

Berisi latar belakang masalah, rumusan masalah, batasan masalah, tujuan,



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

manfaat, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Berisi teori-teori yang mendukung penelitian dan ulasan penelitian terdahulu berdasarkan studi literatur.

BAB III METODOLOGI PENELITIAN

Menjelaskan rancangan penelitian, objek penelitian, dan tahapan penelitian.

BAB IV HASIL DAN PEMBAHASAN

Berisi realisasi rancangan penelitian, yaitu identifikasi kebutuhan, pembuatan model, pengujian, implementasi model ke aplikasi web, dan analisa data yang didapat dari pengujian.

5. BAB V PENUTUP

Berisi kesimpulan penelitian dan saran untuk pengembangan lebih lanjut.

**POLITEKNIK
NEGERI
JAKARTA**

BAB II

TINJAUAN PUSTAKA

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

1 Penelitian Terkait

berikut merupakan hasil penelitian terkait tentang karya peneliti sebelumnya dengan riset yang saat ini dilakukan.

Tabel 2. 1 Penelitian Terdahulu

No	Penulis	Judul/Fokus Utama	Hasil Utama	Perbedaan
1.	Hendra et al. (2025)	Memperkuat Autentikasi dan Integritas Data REST-API Menggunakan Token HMAC SHA-256	Formulasi token HMAC terbukti efektif memvalidasi integritas data dan mendeteksi modifikasi <i>payload</i> . Sistem mampu menolak token yang digunakan ulang (<i>token reused</i>) melalui mekanisme pengecekan log di database.	Penelitian ini mengintegrasikan teknologi Redis (In-Memory) sebagai lapisan keamanan utama untuk validasi <i>Idempotency</i> secara <i>real-time</i> . Hal ini berbeda dengan Hendra yang mekanisme validasinya masih bergantung sepenuhnya pada pencatatan ke basis data relasional (MySQL) yang memiliki latensi lebih tinggi (lebih lambat).



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

No	Penulis	Judul/Fokus Utama	Hasil Utama	Perbedaan
2.	Wang (2024)	Optimizing Payment System with Microservices and Event- Driven Architecture: The Case of Mollie Platform	Transisi arsitektur berhasil menurunkan latensi API secara signifikan (dari >1 detik menjadi stabil di kisaran 200–450 ms). Namun, penelitian ini menemukan kendaladuplikasi event (<i>event duplication</i>) yang menyebabkan proses berulang (<i>redundant triggers</i>), di manastrategi anti-duplikasi berbasis <i>timestamp</i> yang diterapkan terbukti belum sepenuhnya efektif menangani masalah tersebut.	Penelitian ini mengisi celah (<i>gap</i>) tersebut dengan mengimplementasika solusi teknis spesifik berupa Idempotency Key berbasis Redis (In-duplikasi event (<i>event Memory</i>) yang dirancang khusus untuk memblokir duplikasi tersebut secara lebih presisi dibandingkan metode <i>timestamping</i> biasa.



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

No	Penulis	Judul/Fokus Utama	Hasil Utama	Perbedaan
3.	Qiu et al. (2022)	A Deep Study of the Effects and Fixes of Server-Side Request Races in Web Applications	Menemukan bahwa 54% efek dari <i>request races</i> menyebabkan pelanggaran semantik serius, seperti pencatatan data ganda (<i>Duplicate Entries</i>) dan penimpaan data (<i>Data Overwritten</i>), yang seringkali tidak terdeteksi oleh error standar (<i>Silent Failures</i>). Studi ini juga mengungkapkan bahwa mekanisme <i>locking</i> basis data konvensional seringkali tidak cukup cepat untuk menangani <i>race condition</i> pada trafik tinggi.	Penelitian ini mengambil langkah lebih lanjut dengan berfokus pada implementasi solusi mitigasi spesifik menggunakan <i>Idempotency Key</i> berbasis Redis untuk mencegah jenis <i>race condition</i> yang diidentifikasi oleh Qiu pada konteks sistem pembayaran.



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

No	Penulis	Judul/Fokus Utama	Hasil Utama	Perbedaan
4.	Joshi (2024)	Redis Cache Optimization for Payment Gateways in the Cloud	Penerapan Redis terbukti menurunkan rata-rata waktu respons (<i>response time</i>) signifikan sebesar 35% (dari 120 ms menjadi 78 ms) dan meningkatkan <i>throughput</i> transaksi sebesar 40%. Studi ini menyimpulkan bahwa Redis sangat efektif mengurangi beban basis data utama (<i>Database Offloading</i>) pada skenario trafik tinggi.	Penelitian ini mengalihfungsikan Redis untuk tujuan keamanan (<i>Security</i>), sebagai penyimpanan sementara kunci untuk <i>Idempotency</i> untuk memblokir serangan berulang (<i>Replay Attack</i>) secara <i>real-time</i> .



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

No	Penulis	Judul/Fokus Utama	Hasil Utama	Perbedaan
5.	Shrikant (2025)	Enhancing Retry Policies for Webhook Delivery in Fintech Platforms	Strategi <i>Hybrid Model</i> (kombinasi <i>retry</i> cepat di awal dan interval terbatas) terbukti paling efektif meningkatkan tingkat keberhasilan pengiriman (<i>Delivery Success Rate</i>) sekaligus mengurangi beban sistem (<i>System Load</i>) dan risiko	Penelitian ini berfokus pada keamanan (<i>Security</i>) (memastikan pesan yang sampai berkali-kali tidak diproses ganda) dengan menerapkan filter Idempotency Key di sisi penerima (<i>Receiver</i>) menggunakan Redis.
			"badai <i>retry</i> " (<i>Retry Storms</i>). Studi ini menekankan pentingnya manajemen <i>retry</i> untuk mencegah kegagalan sistem akibat lonjakan trafik.	

2.2 Payment Gateway

Payment Gateway didefinisikan sebagai medium transaksi yang berfungsi mengotorisasi pemrosesan pembayaran melalui berbagai metode seperti kartu kredit, transfer bank, hingga *e-wallet*. Layanan ini memfasilitasi proses penyelesaian transaksi (*settlement*) secara daring pada saat transaksi berlangsung dan umumnya menyediakan fasilitas *Application Programming Interface* (API) untuk integrasi sistem yang fleksibel (Baihaqie, 2021).



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2.3 Webhook

Webhook atau "*Reverse API*" merupakan mekanisme komunikasi *event-driven* di mana server mengirimkan data secara otomatis (*push*) melalui HTTP POST ke *endpoint* tujuan saat terjadi peristiwa tertentu. Metode ini menawarkan efisiensi tinggi dan komunikasi *real-time* dengan menghilangkan latensi yang umum terjadi pada metode tradisional *polling*. Kendati demikian, implementasinya mewajibkan standar keamanan ketat melalui verifikasi tanda tangan kriptografis (HMAC-SHA256) dan penerapan prinsip *idempotency* untuk menjamin integritas data serta mencegah duplikasi notifikasi (Mahadik et al., 2025).

2.4 Replay Attack (Serangan Pengulangan)

Replay Attack merupakan metode serangan siber di mana penyerang menyalin seluruh atribut data dari sebuah permintaan (*request*) yang valid termasuk *header* dan *payload* yang terekam dalam sesi komunikasi sebelumnya untuk dikirimkan kembali ke *server* target. Serangan ini mengeksploitasi celah pada sistem autentikasi dengan cara menggunakan ulang kredensial atau token akses (*token reuse*) agar *server* memproses permintaan duplikat tersebut sebagai transaksi baru yang sah, sebuah kerentanan yang terbukti dapat ditembus jika sistem tidak memiliki mekanisme audit *log* atau validasi keunikan *token* untuk mendeteksi pengulangan data (Hendra et al., 2025).

2.5 Webhook Spoofing (Pemalsuan Identitas)

Webhook Spoofing merupakan teknik serangan siber di mana pihak yang tidak berwenang memanipulasi identitas pengirim untuk menyamar sebagai entitas yang sah (*legitimate user*) saat berinteraksi dengan sistem API. Dalam konteks keamanan aplikasi *Fintech*, ancaman ini memungkinkan penyerang untuk mengelabui mekanisme autentikasi guna mendapatkan akses tak terotorisasi atau memvalidasi transaksi palsu seolah-olah data tersebut berasal dari sumber tepercaya, yang mengakibatkan pelanggaran integritas data dan potensi kerugian finansial (Ranjan et al., 2022).



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2.6 Race Condition (Kondisi Balapan Data)

Race Condition atau *Server-Side Request Races* adalah kerentanan yang timbul ketika beberapa permintaan dieksekusi secara konkuren dan bersaing mengakses sumber daya dalam urutan yang tidak terprediksi. Dalam konteks finansial, anomali ini menyebabkan pelanggaran atomisitas (*atomicity violation*), di mana jeda waktu pemrosesan memungkinkan terjadinya eksekusi ganda pada satu transaksi sebelum status data berhasil diperbarui, yang berujung pada kerugian finansial nyata (Qiu et al., 2022).

2.7 HMAC-SHA256

HMAC-SHA256 (*Hash-based Message Authentication Code with Secure Hash Algorithm 256-bit*) adalah konstruksi kriptografis yang menggabungkan fungsi *hashing* dengan kunci rahasia (*secret key*) untuk menghasilkan tanda tangan digital. Algoritma ini berfungsi vital dalam menjamin otentikasi dan integritas data selama transmisi, di mana setiap upaya manipulasi pada data (*payload*) atau penggunaan kunci yang tidak sah akan menghasilkan nilai *hash* yang berbeda secara drastis, sehingga pemalsuan data dapat terdeteksi secara instan (Taofik et al., 2023).

2.8 Idempotency Key

Idempotency adalah prinsip desain sistem yang memastikan bahwa sebuah operasi transaksi dapat dieksekusi berulang kali tanpa menghasilkan efek samping yang berbeda dari eksekusi pertama. Dalam arsitektur *Webhook*, mekanisme ini diterapkan dengan memvalidasi kunci unik (*unique event ID*) pada setiap *payload* yang masuk untuk menangani pengiriman ulang (*retry*) otomatis dari penyedia layanan, sehingga mencegah terjadinya duplikasi data atau pencatatan ganda meskipun pesan yang sama diterima berkali-kali akibat gangguan jaringan (Mahadik et al., 2025).

2.9 Redis (Remote Dictionary Server)

Redis diklasifikasikan sebagai penyimpanan data *Key-Value* berbasis memori (*In-memory Key-Value Store*) yang dirancang khusus untuk menangani kebutuhan aplikasi *real-time* dengan kecepatan ekstrim, seperti analitik data dan manajemen



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

sesi pengguna. Dalam arsitektur mikroservis finansial, teknologi ini memainkan peran vital dalam menjamin konsistensi data berkecepatan tinggi (*high-speed data consistency*) melalui mekanisme akses data memori yang jauh lebih responsif dibandingkan basis data berbasis disk. Selain kecepatan, Redis menyediakan fitur optimasi performa berupa mekanisme TTL (*Time to Live*), yang memungkinkan kunci data memiliki batas waktu kedaluwarsa otomatis, sehingga efisiensi penggunaan memori dan kesegaran data dalam sistem terdistribusi tetap terjaga secara optimal (Chaudhari & Verma, 2024).

2.10 MySQL

MySQL merupakan implementasi Sistem Manajemen Basis Data (DBMS) yang berfungsi mengelola data dalam format terstruktur (tabel, kolom, dan baris) untuk efisiensi pemrosesan. Peran DBMS dalam arsitektur sistem sangat krusial untuk mengatur akses pengguna, menjamin keamanan informasi sensitif, serta mengintegrasikan data guna mendukung pengambilan keputusan yang valid (Aulia et al., 2023).

2.11 Burp Suite

Burp Suite diklasifikasikan sebagai platform terintegrasi yang dirancang khusus untuk melakukan pengujian keamanan pada aplikasi web (*web application security testing*). Perangkat lunak ini menyediakan berbagai fitur yang memungkinkan penguji untuk melakukan analisis kerentanan secara menyeluruh, mulai dari pemetaan aplikasi hingga eksploitasi celah keamanan. Dalam konteks pengujian penetrasi, *Burp Suite* berfungsi efektif untuk mendeteksi berbagai jenis kerentanan web standar seperti *SQL Injection* dan *Cross-Site Scripting (XSS)* melalui mekanisme intersepsi dan manipulasi lalu lintas data antara pengguna dan server (Rainita et al., 2023).

2.12 Apache JMeter

Apache JMeter merupakan perangkat lunak berbasis Java yang digunakan untuk melakukan pengujian beban (*load testing*) dan pengukuran kinerja sistem dengan cara mensimulasikan pengiriman lalu lintas (*traffic*) jaringan yang tinggi ke server



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

target. Alat ini dimanfaatkan untuk memverifikasi stabilitas konfigurasi sistem dan mekanisme *autoscaling* dengan cara mengirimkan beban trafik pengguna secara bertahap (mulai dari 1.000 hingga 20.000 *user*), guna memastikan server mampu menangani lonjakan permintaan secara simultan tanpa mengalami kegagalan layanan (Gurohman et al., 2024).

4.13 Metrik Pengujian Sistem

Untuk mengukur keberhasilan sistem pertahanan yang diusulkan, penelitian ini meninggalkan parameter berbasis probabilitas dan berfokus pada evaluasi arsitektur jaringan secara deterministik. Parameter evaluasi utama ditekankan pada efisiensi performa, stabilitas beban, dan tingkat ketahanan sistem keamanan dalam memblokir muatan anomali (Joshi, 2024; Hendra et al., 2025). Adapun metrik pengujian yang digunakan adalah sebagai berikut:

1. Tingkat Penolakan Serangan (*Error Percentage / Block Rate*)

Metrik ini mengukur persentase muatan serangan yang berhasil dideteksi dan ditolak secara aktif oleh sistem dibandingkan dengan total muatan eksploitasi yang dikirimkan. Penolakan ini ditandai dengan interupsi kode status HTTP secara spesifik (seperti HTTP 400 *Bad Request*, 401 *Unauthorized*, 409 *Conflict*, atau 429 *Too Many Requests*) yang memotong siklus permintaan sebelum mencapai lapisan basis data utama (Hendra et al., 2025; Qiu et al., 2022).

2. Waktu Respons (*Latency*)

Dalam konteks performa transaksi, latensi didefinisikan sebagai interval waktu rata-rata yang dibutuhkan sistem untuk memproses sebuah permintaan. Waktu ini dihitung sejak instrumen pengujian menginisiasi pengiriman muatan data (request initiation) hingga sistem mengembalikan status akhir. Penurunan tingkat latensi menjadi indikator krusial dalam menjaga stabilitas sistem ketika menghadapi kondisi lalu lintas jaringan dengan konkurensi tinggi (Joshi, 2024; Wang, 2024).

Kapasitas Penyelesaian (*Throughput*)

Throughput mendeskripsikan kapasitas maksimal server dalam menyelesaikan dan merespons jumlah permintaan secara serentak dalam kurun waktu satu detik (*requests/second*). Metrik ini digunakan untuk mengukur efisiensi komputasi dan ketahanan arsitektur sistem dari potensi kelumpuhan (*resource exhaustion*) saat dihadapkan pada skenario serangan bervolume masif (Joshi, 2024).

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



BAB III METODE PENELITIAN

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

1 Rancangan Penelitian

Rancangan penelitian ini menggunakan metode Kuantitatif dengan pendekatan Eksperimental (*Experimental Research*). Metode ini dipilih secara spesifik karena pengujian efektivitas keamanan siber menuntut pembuktian yang objektif dan terukur, di mana keberhasilan suatu mitigasi tidak dapat dinilai hanya berdasarkan analisis deskriptif. Melalui pendekatan eksperimental, peneliti dapat memanipulasi variabel bebas (kondisi sistem) dan mengukur dampaknya terhadap variabel terikat dalam lingkungan yang terkendali. Hal ini mendasari tujuan penelitian untuk mengukur secara sistematis efektivitas mekanisme pertahanan berlapis (*Defense-in-Depth*) dalam memitigasi serangan *Replay Attack* dan *Webhook Spoofing* pada sistem pembayaran digital, sehingga menghasilkan data empiris yang valid berupa persentase penolakan serangan (*Error Percentage*) serta metrik performa jaringan seperti waktu respons (*Latency*) dan kapasitas penyelesaian (*Throughput*) dari hasil pengujian.

Pelaksanaan penelitian diawali dengan tahap simulasi serangan dan pengujian beban menggunakan instrumen standar yang telah dibatasi pada ruang lingkup penelitian. Pada tahap ini, Burp Suite diterapkan untuk mensimulasikan serangan penetrasi dan manipulasi paket data, Postman digunakan untuk validasi fungsional logika bisnis API, serta Apache JMeter dimanfaatkan untuk mengukur parameter performa (*Latency* dan *Throughput*) pada kondisi lalu lintas tinggi (*high concurrency*). Pengujian dilakukan secara komparatif pada dua kondisi sistem, yaitu kondisi sebelum mitigasi (*Vulnerable State*) yang hanya mengandalkan validasi basis data konvensional, dan kondisi setelah mitigasi (*Secured State*) yang menerapkan solusi yang diusulkan.

Sistem pertahanan dikembangkan dengan mengintegrasikan dua lapisan keamanan utama pada *Middleware* server. Lapisan pertama menggunakan algoritma HMAC-SHA256 untuk memverifikasi tanda tangan digital (*signature*) dari setiap permintaan masuk guna memastikan integritas data dan otentikasi pengirim.



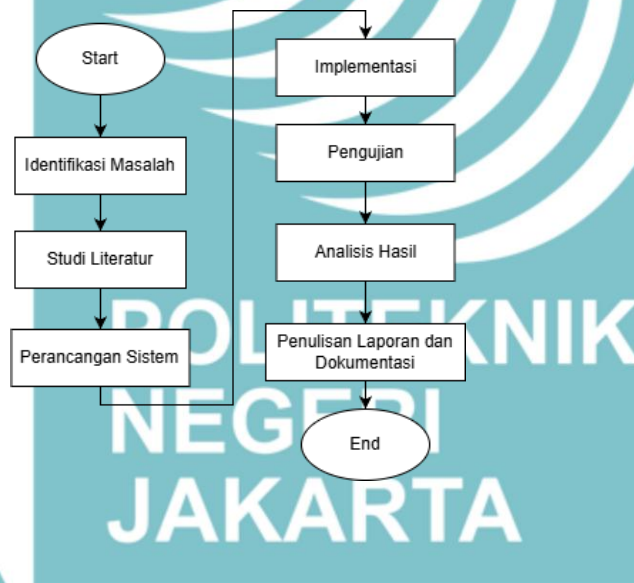
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

apian kedua menerapkan mekanisme Idempotency Key menggunakan penyimpanan berbasis memori Redis, yang berfungsi mendeteksi dan menolak permintaan dengan ID transaksi duplikat secara *real-time*.

Output yang diharapkan dari rancangan ini meliputi: kemampuan sistem dalam mendeteksi dan menolak akses tidak sah (ditandai dengan interupsi kode status HTTP seperti 400 *Bad Request*, 401 *Unauthorized*, 409 *Conflict*, atau 429 *Too Many Requests*), terjaminnya integritas data transaksi yang bebas dari duplikasi (*double spending*) maupun manipulasi, serta pemrosesan transaksi yang valid tanpa gangguan performa yang signifikan. Data hasil pengujian ini selanjutnya akan dianalisis untuk membuktikan hipotesis penelitian.

2. Tahapan Penelitian



Gambar 3. 1 Tahapan Penelitian

Tahapan penelitian ini menjelaskan langkah-langkah yang dilakukan dalam proses penelitian. Berikut adalah uraian dari *flowchart* pada Gambar 3.1 mengenai tahapan penelitian yang dilaksanakan:

1. Identifikasi Masalah

Tahap ini merupakan langkah fundamental untuk merumuskan urgensi penelitian berdasarkan data faktual di lapangan. Proses identifikasi dilakukan melalui pengumpulan data primer menggunakan metode observasi langsung dan



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

wawancara. Penulis meninjau alur kerja sistem yang berjalan (*existing system*) serta membedah kode program untuk memetakan celah keamanan teknis secara

mendalam. Berdasarkan hasil penelusuran tersebut, ditemukan kelemahan signifikan pada modul *Callback Handler* di Website PsikoTest, di mana sistem saat ini menerima data *webhook* secara mentah tanpa adanya mekanisme validasi autentikasi pengirim maupun filter duplikasi. Temuan teknis ini kemudian divalidasi melalui sesi diskusi dengan pengguna guna mengonfirmasi risiko operasional yang ditimbulkan, yang menyimpulkan bahwa absensi mekanisme pertahanan tersebut menjadikan sistem sangat rentan terhadap eksploitasi serangan manipulasi data dan *Replay Attack*.

Studi literatur

Studi literatur merupakan proses kajian ilmiah untuk mengumpulkan referensi dari berbagai sumber terpercaya, seperti jurnal, buku, prosiding, dan dokumentasi teknis (API Documentation, Redis, & Burp Suite) yang relevan dengan topik penelitian. Referensi tersebut dijadikan sebagai landasan teoritis dan acuan utama dalam membangun gagasan solusi serta merancang mekanisme mitigasi yang standar.

3. Perancangan Sistem

Tahap ini berfokus pada perancangan arsitektur keamanan *Defense-in-Depth* yang mencakup logika *Middleware* dan skema penyimpanan Redis. Desain ini dirancang untuk menjamin integritas data dengan mencegah *Double Spending*, serta mengaktifkan mekanisme mitigasi otomatis terhadap serangan *Replay Attack* dan *Webhook Spoofing* sebagai dasar acuan implementasi teknis.

4. Implementasi

Pada tahap ini dilakukan pengimplementasian mekanisme keamanan pada lingkungan server sesuai dengan rancangan sistem. Proses ini meliputi konfigurasi infrastruktur dan penerapan kode program untuk mengaktifkan fungsi mitigasi pada sisi *backend* agar siap dilakukan pengujian.



Pengujian

Tahap ini dilakukan uji coba terhadap sistem keamanan untuk memastikan bahwa mekanisme mitigasi yang dibangun berfungsi sesuai dengan rancangan. Uji coba ini meliputi simulasi serangan menggunakan Burp Suite dan Postman untuk mengevaluasi efektivitas pemblokiran serangan secara mutlak, serta pengujian beban (*load testing*) menggunakan Apache JMeter untuk memastikan sistem mampu menangani lalu lintas tinggi (*high concurrency*) tanpa degradasi performa yang fatal.

Analisis Hasil

Setelah pengujian selesai, dilakukan analisis terhadap data yang dikumpulkan dari log server dan hasil pengukuran instrumen. Analisis ini melibatkan interpretasi data berdasarkan metrik persentase penolakan serangan (*Error Percentage*) dan evaluasi dampak penerapan keamanan terhadap performa sistem, khususnya pada parameter waktu respons (*Latency*) dan kapasitas penyelesaian (*Throughput*), guna memastikan hipotesis penelitian terbukti secara empiris.

7. Penulisan Laporan dan Dokumentasi

Pada tahapan akhir ini, seluruh proses pengujian sistem yang telah dilakukan akan dicatat dalam dokumentasi teknis dan disusun ke dalam laporan tugas akhir. Proses ini bertujuan untuk mengevaluasi data kinerja sistem yang diperoleh, serta menjadi dasar perumusan saran bagi pengembangan penelitian selanjutnya.

3.3 Objek Penelitian

Objek penelitian ini adalah modul *Callback Handler* pada arsitektur *Backend* Website PsikoTest yang dibangun menggunakan teknologi berbasis Node.js dan diimplementasikan pada server simulasi Ubuntu sebagai media pengujian komparatif. Pemilihan modul ini sebagai objek utama didasarkan pada peran vitalnya sebagai titik entri (*entry point*) sinkronisasi pembayaran yang merupakan vektor serangan paling kritis terhadap manipulasi logika bisnis, sehingga kegagalan validasi pada area ini berimplikasi langsung pada kerugian finansial perusahaan. Untuk membuktikan efektivitas mitigasi, sistem dirancang agar dapat diuji dalam

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

dua kondisi berbeda, yaitu kondisi rentan (*vulnerable*) tanpa validasi dan kondisi aman (*secured*) yang telah mengintegrasikan verifikasi integritas HMAC-SHA256 serta mekanisme *Idempotency Key* berbasis Redis, di mana respons sistem terhadap stimulasi serangan *Replay Attack* dan *Webhook Spoofing* termasuk status blokir dan waktu respons (*latency*) akan dicatat secara otomatis dalam log aplikasi sebagai data embanding yang empiris.





Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :

a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.

b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun

tanpa izin Politeknik Negeri Jakarta

BAB IV HASIL DAN PEMBAHASAN

1 Analisis Kebutuhan

Analisis kebutuhan sistem merupakan tahapan krusial untuk mengidentifikasi dan menentukan spesifikasi perangkat keras (*hardware*), infrastruktur *cloud*, maupun perangkat lunak (*software*) yang diperlukan dalam membangun, mengintegrasikan, dan menguji lingkungan penelitian. Berdasarkan batasan masalah dan objek penelitian yang berfokus pada analisis mitigasi serangan *Replay Attack* dan *Webhook Spoofing* menggunakan arsitektur *Defense-in-Depth* (HMAC-SHA256 dan *Idempotency Key* berbasis Redis), maka kebutuhan lingkungan simulasi (*environment*) diimplementasikan langsung pada layanan *Virtual Private Server* (VPS) untuk mendapatkan kondisi *server production* secara empiris. Rincian perangkat yang digunakan dijabarkan pada Tabel 4.1.

Tabel 4. 1 Analisis Kebutuhan Perangkat

No	Nama	Spesifikasi
Perangkat Keras (Hardware)		
1.	Laptop	CPU: AMD Ryzen 5500 RAM: 12GB Storage: 512 GB
Perangkat Lunak (Software)		
2.	Virtual Private Server (VPS)	OS: Ubuntu Server 24.04 LTS IP Address: 31.97.51.230 Domain: psiko.deltaindonesialab.com CPU: 1 Core RAM: 4 GB Storage: 50 GB
3.	Node.js	Version 18.19.1
4.	MariaDB	Version: 10.11.14-MariaDB Host: 109.106.253.171
5.	Redis	Version 7.2.7



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

No	Nama	Spesifikasi
6.	Postman API	Version 12.0.1
7.	Burp Suite	Community Edition
8.	Apache JMeter	Version 5.6.3
9.	Tripay Sandbox	Website Payment Gateway

Berdasarkan Tabel 4.1 di atas, seluruh spesifikasi perangkat keras dan lunak yang digunakan tidak sekadar beroperasi secara independen, melainkan saling terintegrasi untuk menciptakan lingkungan simulasi pengujian yang komprehensif. Perangkat keras berupa laptop digunakan oleh peneliti sebagai *host* lokal untuk melakukan pengembangan arsitektur kode sekaligus mengeksekusi berbagai skenario pengujian penetrasi keamanan. Sementara itu, infrastruktur sistem utama diimplementasikan pada layanan VPS bersistem operasi Ubuntu Server 24.04 LTS guna mereplikasi kondisi *server production* publik yang sebenarnya.

Pada tahapan integrasi sistem *back-end*, Node.js difungsikan sebagai *runtime environment* utama untuk membangun antarmuka pemrograman aplikasi (API) yang bertugas menerima dan menganalisis *webhook* dari Tripay Sandbox. Penyimpanan data relasional, seperti riwayat pembayaran dan status transaksi, dikelola menggunakan MariaDB. Sebagai komponen krusial dalam arsitektur keamanan, Redis diintegrasikan secara khusus sebagai sistem penyimpanan *in-memory* berkecepatan tinggi untuk menangani mekanisme *Idempotency Key*, yang bertugas melakukan penguncian tingkat memori (*distributed lock*) guna mencegah eksekusi ganda pada sistem.

Untuk pemenuhan tujuan analisis keamanan, digunakan beberapa perangkat lunak spesifik yang disesuaikan dengan skenario serangan. Postman API digunakan untuk validasi fungsionalitas dan integrasi awal *endpoint webhook*. Burp Suite digunakan untuk melakukan intersepsi, modifikasi *header* otentikasi (HMAC-SHA256), dan manipulasi *payload* guna menganalisis efektivitas mitigasi sistem terhadap serangan *Webhook Spoofing*. Selanjutnya, Apache JMeter difungsikan sebagai generator beban untuk mengirimkan ratusan *request* duplikat secara serentak (*concurrency*), yang bertujuan untuk menganalisis secara mendalam ketahanan



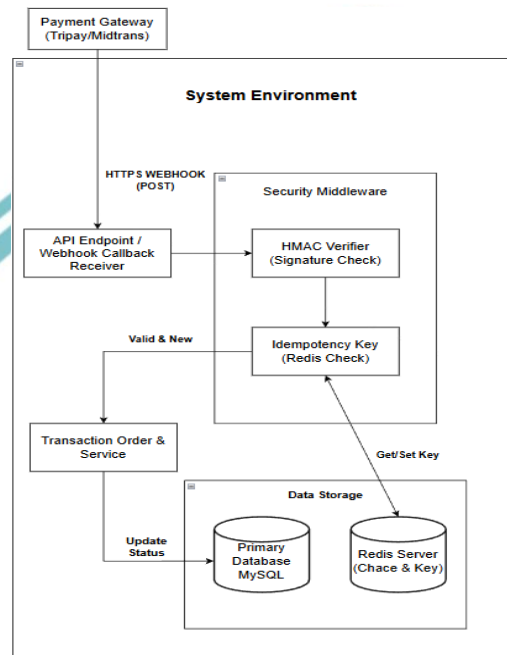
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

integrasi sistem terhadap *Replay Attack* dan eksploitasi celah *Race Condition*.

4.2 Perancangan Sistem

4.2.1 Diagram Blok Sistem

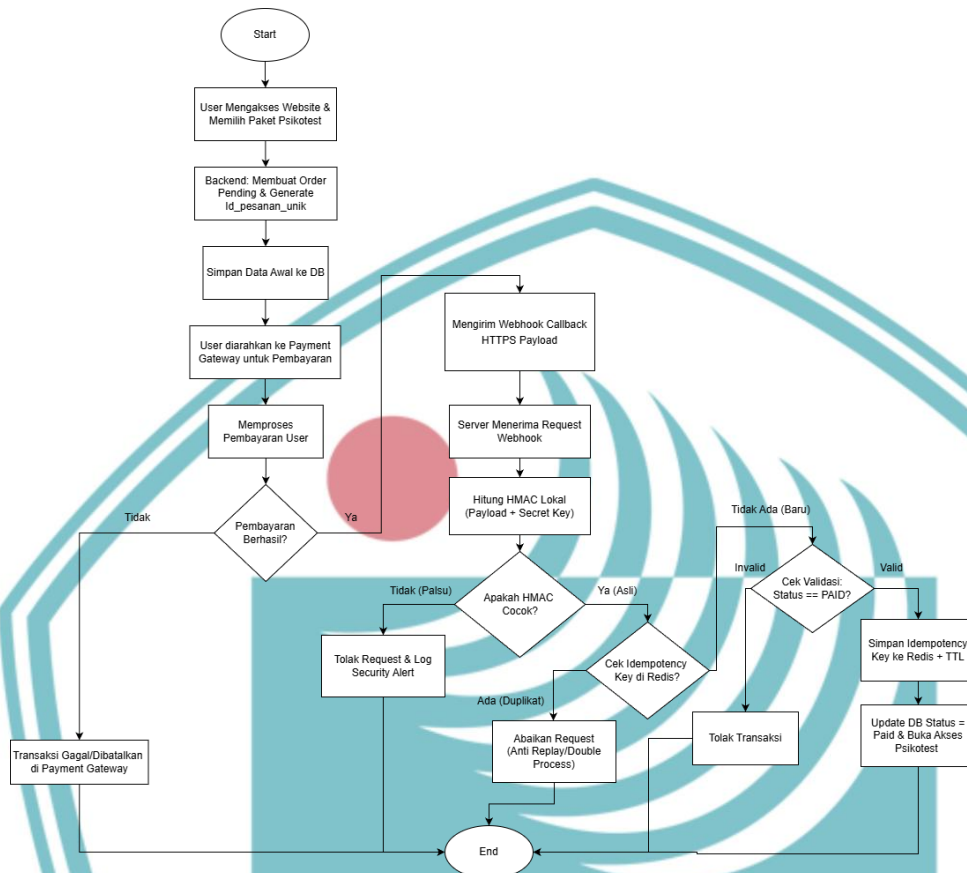


Gambar 4. 1 Diagram Blok Sistem

Pada Gambar 4.1 memvisualisasikan rancangan arsitektur sistem yang berfokus pada mekanisme penanganan aman (*secure handling*) untuk *callback* pembayaran, dimulai dari *Payment Gateway* (Tripay/Midtrans) yang mengirimkan notifikasi melalui protokol *HTTPS Webhook (POST)* ke *API Endpoint Receiver*. Data yang diterima tidak langsung ditulis ke basis data utama, melainkan diarahkan masuk ke blok *Security Middleware* untuk menjalani validasi bertingkat, diawali dengan *HMAL Verifier (Signature Check)* guna memverifikasi otentikasi pengirim dan dilanjutkan dengan pengecekan *Idempotency Key* ke *Redis Server (Get/Set Key)* untuk mendeteksi paket duplikat. Mekanisme ini memastikan bahwa hanya permintaan yang terkonfirmasi valid dan baru (*Valid & New*) yang diizinkan lolos untuk melakukan pembaruan status transaksi (*Update Status*) secara pembaruan status transaksi (*Update Status*) secara permanen ke dalam basis data utama.



4.2.2 Flowchart Sistem



Gambar 4. 2 Flowchart Sistem

Pada gambar 4.2 menunjukkan alur kerja sistem validasi *callback payment gateway* yang menerapkan strategi *Defense-in-Depth* untuk memitigasi serangan manipulasi transaksi. Saat *server* menerima permintaan *webhook*, sistem pertama-tama melakukan verifikasi integritas dengan menghitung ulang HMAC-SHA256 untuk memastikan data berasal dari sumber resmi (*Anti-Spoofing*). Jika validasi tanda tangan berhasil, sistem melanjutkan pengecekan keunikan *Idempotency Key* pada penyimpanan memori Redis untuk mendeteksi indikasi pengulangan paket data (*Replay Attack*). Sistem memastikan bahwa hanya transaksi yang terbukti otentik dan belum pernah diproses sebelumnya yang akan disimpan ke dalam basis data utama, sementara permintaan yang terindikasi manipulatif atau duplikat akan ditolak secara otomatis tanpa membebani kinerja server.

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkannya dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

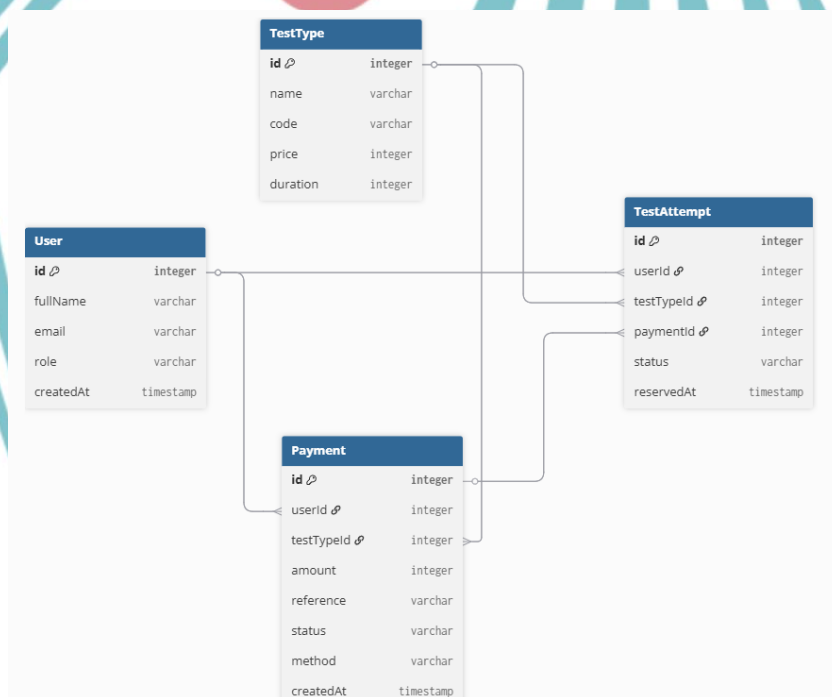


4.2.3 Perancangan Basis Data

Perancangan basis data pada sistem ini dimodelkan menggunakan *Entity Relationship Diagram* (ERD) untuk memetakan entitas dan relasi antar tabel secara logis. Untuk memudahkan pemahaman arsitektur data, perancangan ERD dibagi menjadi dua modul utama, yaitu Modul Transaksi dan Pembayaran, serta Modul Pelaksanaan Tes.

1. Modul Transaksi dan Pembayaran

Modul ini merupakan inti dari fungsionalitas *payment gateway* yang menjadi fokus utama dalam implementasi mitigasi keamanan.



Gambar 4. 3 Entity Relationship Diagram (ERD) Modul

Transaksi dan Pembayaran

Berdasarkan Gambar 4.3, modul ini berpusat pada entitas *Payment* yang mencatat seluruh riwayat transaksi pengguna. Entitas *Payment* memiliki relasi dengan entitas *User* (pengguna yang melakukan transaksi), *TestType* (jenis tes yang dibeli), serta *TestAttempt* (sesi tes yang akan diakses setelah pembayaran berhasil). Dalam konteks keamanan sistem, entitas *Payment* memiliki peran yang sangat kritical,



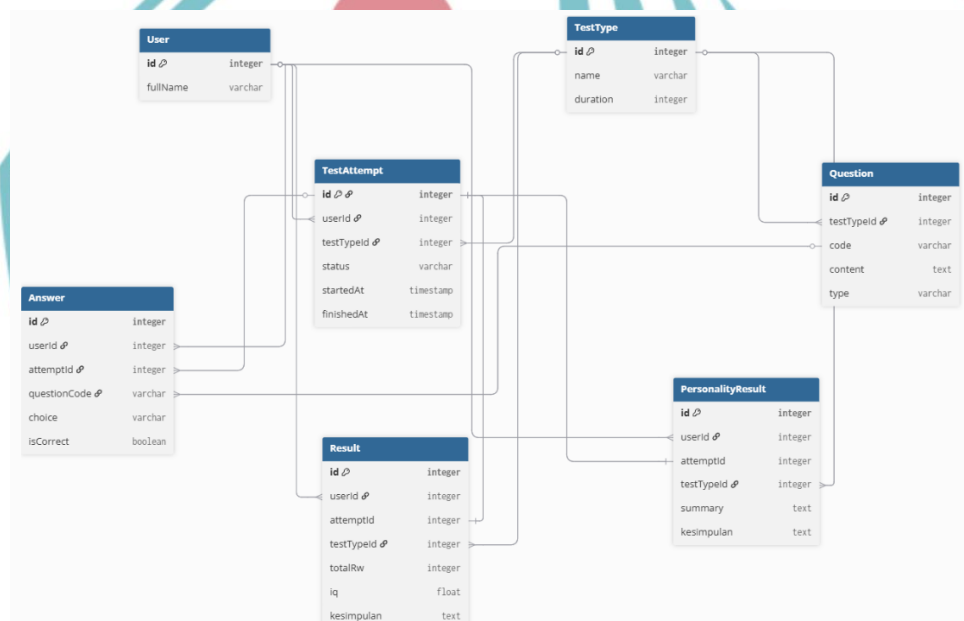
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

khususnya pada atribut status dan reference. Atribut-atribut inilah yang menjadi target utama pelindungan dari ancaman *Replay Attack* dan *Webhook Spoofing*. Perubahan status dari *PENDING* menjadi *SUCCESS* harus melalui validasi *signature* HMAC dan pengecekan *Idempotency Key* di *Redis* sebelum rekaman pada tabel ini diperbarui.

2. Modul Pelaksanaan dan Hasil Tes

Setelah alur transaksi tervalidasi dengan aman, sistem akan mengelola alur pelaksanaan tes psikologi melalui entitas-entitas pada modul kedua.



Gambar 4. 4 Entity Relationship Diagram (ERD) Modul
Pelaksanaan dan Hasil Tes

Gambar 4.4 mengilustrasikan relasi data saat pengguna mengeksekusi tes. Pusat dari aktivitas ini berada pada entitas *TestAttempt*, yang mencatat waktu mulai (*startedAt*) dan waktu selesai (*finishedAt*) dari sebuah sesi tes. Entitas ini memiliki relasi *One-to-Many* dengan entitas *Answer* untuk menyimpan setiap jawaban yang diinputkan pengguna terhadap entitas *Question*. Setelah tes diselesaikan, sistem akan mengalkulasi skor dan menyimpannya pada entitas *Result* (untuk hasil kuantitatif seperti IQ atau total benar/salah) serta *PersonalityResult* (untuk hasil analisis kualitatif atau kesimpulan kepribadian). Pemisahan entitas hasil ini

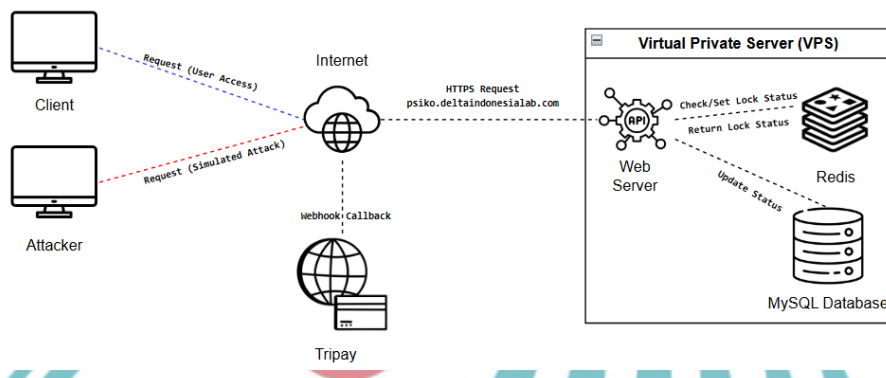


Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

dirancang agar sistem dapat mengakomodasi berbagai format laporan psikotes secara fleksibel.

4.2.4 Topologi Jaringan Sistem



Gambar 4. 5 Topologi Jaringan Sistem

Pada Gambar 4.5 ditunjukkan topologi jaringan sistem yang digunakan dalam proses implementasi dan pengujian keamanan *callback payment gateway*. Arsitektur ini dirancang untuk mereplikasi kondisi server produksi (*production environment*) secara empiris pada lingkungan *staging*. Topologi ini terdiri dari *client node*, *attacker node*, entitas Tripay, jaringan publik (Internet), serta *Virtual Private Server* (VPS) berbasis Ubuntu yang mengintegrasikan layanan *web server* (Next.js), basis data MariaDB, dan *in-memory storage* Redis.

Secara operasional, *client node* merepresentasikan pengguna sah yang mengakses aplikasi melalui peramban (*browser*). Sebaliknya, *attacker node* digunakan untuk menyimulasikan ancaman siber, seperti *Webhook Spoofing* dan *Replay Attack*, menggunakan instrumen pengujian keamanan. Selain itu, terdapat entitas Tripay yang bertugas mengirimkan *payload* status pembayaran secara asinkron melalui mekanisme *webhook callback*. Seluruh permintaan dari *client*, *attacker*, maupun Tripay melewati jaringan publik dan diarahkan langsung menuju *endpoint* publik VPS melalui protokol HTTPS (psiko.deltaindonesialab.com). Penggunaan HTTPS ini memastikan seluruh data yang ditransmisikan, termasuk *signature* HMAC, berada dalam jalur komunikasi yang terenkripsi (SSL/TLS).



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Pada sisi *server*, aplikasi *backend* Next.js memproses setiap permintaan yang masuk. Untuk memitigasi *Replay Attack*, sistem secara sinkron memanfaatkan layanan Redis untuk menerapkan mekanisme *distributed lock* (*Idempotency Key*). Mekanisme ini krusial untuk mencegah pemrosesan transaksi ganda (*double spending*) pada kondisi *high concurrency* sebelum pembaruan status pembayaran dicatat ke dalam basis data MariaDB. Dari perspektif *server*, seluruh permintaan yang masuk diperlakukan sebagai *HTTPS request* yang serupa. Oleh karena itu, sistem memerlukan lapisan validasi logika yang kuat pada *middleware*, khususnya verifikasi *signature* HMAC, untuk membedakan antara *callback* otentik dari server Tripay dan upaya manipulasi data (*Webhook Spoofing*) dari *attacker node*.

4.2.5 Konfigurasi Jaringan Publik dan Firewall

Berbeda dengan sistem virtualisasi lokal yang memerlukan konfigurasi NAT (Network Address Translation) dan Port Forwarding, implementasi pada Virtual Private Server (VPS) secara bawaan telah terhubung langsung ke jaringan internet global menggunakan Alamat IP Publik. Oleh karena itu, konfigurasi jaringan pada pengujian ini difokuskan pada pemetaan nama domain (DNS) serta penerapan aturan Firewall untuk mengatur lalu lintas data yang masuk (*inbound traffic*).

1. Pemetaan Domain (DNS Record)

Tabel 4. 2 Konfigurasi DNS (Domain Name System)

Nama Domain	Tipe Record	Target / Value (IP Publik)	Segmentasi
psiko.deltaindonesialab.com	A Record	31.97.51.230	Proxied / Aktif

Berdasarkan Tabel 4.2, IP Publik VPS yaitu 31.97.51.230 dipetakan secara langsung ke sebuah nama domain resmi (psiko.deltaindonesialab.com) menggunakan metode *A Record* pada sistem manajemen DNS. Konfigurasi pemetaan ini merupakan prasyarat mutlak dalam membangun lingkungan *production*. Hal ini dikarenakan *Payment Gateway* (Tripay) memiliki kebijakan keamanan ketat yang mewajibkan URL *endpoint callback* beroperasi menggunakan



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

protokol HTTPS yang terenkripsi (SSL). Tanpa adanya nama domain yang valid, sertifikat keamanan HTTPS tidak dapat diterbitkan pada VPS, sehingga seluruh *request webhook* simulasi pembayaran akan ditolak secara otomatis oleh sistem Tripay.

2. Konfigurasi Keamanan Firewall

Tabel 4. 3 Aturan Firewall (Inbound Rules)

Protokol	Port	Layanan	Fungsi
TCP	80 / 443	HTTP / HTTPS	Akses web <i>client</i> dan penerimaan <i>webhook</i> dari Tripay
TCP	22	SSH	Akses <i>remote terminal</i> secara aman untuk administrator
TCP	10702	Custom Panel	Akses menuju antarmuka <i>dashboard</i> manajemen VPS (aaPanel)
TCP	3000	Node.js / Next.js	<i>Port</i> internal layanan <i>backend</i> API

Merujuk pada Tabel 4.3, sistem menerapkan pembatasan akses berlapis (*firewall*) pada tingkat *server* untuk memitigasi risiko keamanan akses dari luar. *Port* 80 dan 443 dibuka secara publik sebagai gerbang utama lalu lintas data untuk melayani akses *web browser* dari *client* sekaligus menerima *payload webhook* dari Tripay. Untuk kebutuhan konfigurasi dan pemeliharaan *server* jarak jauh, *port* 22 (SSH) dan 10702 (*Custom Panel*) diizinkan namun peruntukannya sangat dibatasi hanya untuk administrator. Selain itu, meskipun aplikasi terhubung ke internet, *port* 3000 yang digunakan oleh layanan *backend* (Next.js) tidak diekspos secara langsung melainkan berjalan secara internal melalui metode *reverse proxy* oleh *web server* utama. Strategi ini memastikan logika inti aplikasi tetap terisolasi dengan aman, sehingga mempersempit celah serangan (*attack surface*).



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4.3 Implementasi Sistem

4.3.1 Inisialisasi Virtual Private Server (VPS) dan Manajemen Panel

Sebelum sistem aplikasi dan mekanisme keamanan diimplementasikan, seluruh infrastruktur *server* dalam penelitian ini dibangun menggunakan layanan *Virtual Private Server* (VPS). Berbeda dengan simulasi mesin virtual lokal yang ruang lingkungannya terbatas, penggunaan VPS menyediakan lingkungan *production* yang sesungguhnya dengan sumber daya komputasi terisolasi dan Alamat IP Publik statis. Pemilihan infrastruktur *cloud* ini sangat krusial mengingat sistem yang diuji melibatkan integrasi *Payment Gateway* eksternal (Tripay) yang membutuhkan akses langsung dari jaringan internet global. Sistem operasi yang digunakan pada *server* ini adalah Ubuntu Server 24.04.1 LTS, yang dipilih karena tingkat keandalannya (*reliability*), standar keamanan yang tinggi, serta efisiensi pengelolaan memori karena berbasis *Command Line Interface* (CLI).

Untuk kebutuhan konfigurasi tingkat dasar (*root privilege*), akses *remote* pertama kali dilakukan secara aman dari perangkat *host* melalui protokol *Secure Shell* (SSH).

```

Local server
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-90-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Tue May  5 07:10:50 UTC 2026

System load:  0.33               Processes:    130
Usage of /:   81.9% of 47.39GB    Users logged in:  0
Memory usage: 34%               IPv4 address for eth0: 31.97.51.230
Swap usage:   0%                IPv6 address for eth0: 2a02:4780:59:d785::1

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

36 updates can be applied immediately.
2 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

5 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log

Last login: Tue May  5 02:23:40 2026 from 127.0.0.1
root@srv1481027:~#

```

Gambar 4. 6 Akses Remote SSH ke VPS Ubuntu



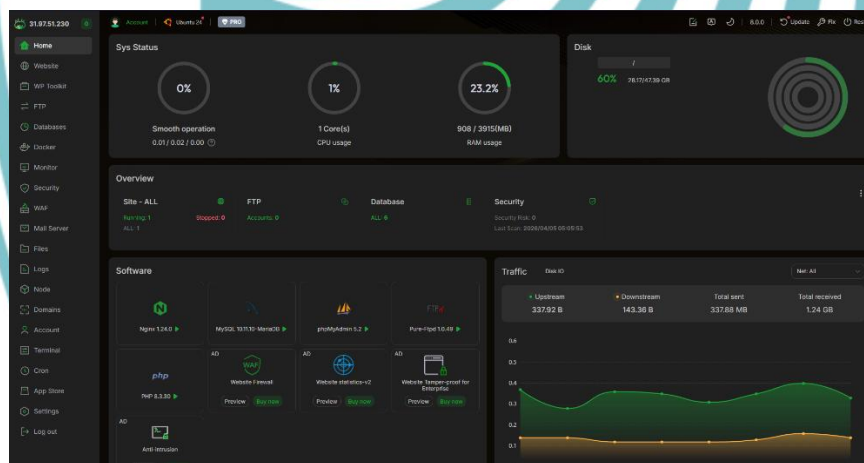
© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Gambar 4.6 menunjukkan antarmuka terminal saat akses *remote* ke *server* Ubuntu berhasil diinisiasi. Setelah otentikasi *login* tervalidasi, sistem menampilkan *Message of the Day* (MOTD) yang memuat ringkasan status *server*, termasuk versi sistem operasi, utilisasi perangkat keras awal, dan alamat IP yang aktif. Melalui akses terminal ini pula, langkah pertama pengamanan *server* dilakukan, yaitu memperbarui indeks repositori dan melakukan *patching* sistem menggunakan perintah *apt update* dan *apt upgrade*. Langkah preventif ini wajib dilakukan guna memastikan seluruh fondasi sistem berada pada versi mutakhir sebelum perangkat lunak lain diinstal.

Selain menggunakan akses CLI murni untuk urusan inti *server*, pengelolaan infrastruktur tingkat lanjut divisualisasikan menggunakan antarmuka panel manajemen aaPanel.



Gambar 4. 7 Antarmuka Dashboard Manajemen aaPanel

Gambar 4.7 memperlihatkan antarmuka *dashboard* utama dari aaPanel yang telah dikonfigurasi. Penggunaan panel kontrol ini tidak hanya berfungsi untuk menyederhanakan konfigurasi lingkungan (*environment*), tetapi juga bertindak sebagai instrumen *monitoring* yang komprehensif. Melalui *dashboard* ini, administrator dapat melacak metrik performa *server* secara *real-time*, seperti lonjakan beban pemrosesan (CPU Load), tingkat konsumsi RAM, serta aktivitas lalu lintas jaringan (*Network Traffic*). Fungsionalitas pemantauan visual ini sangat vital dalam penelitian ini, terutama ketika *server* harus menghadapi simulasi pengujian serangan dengan tingkat konkurensi tinggi (*high concurrency*) pada



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

tahapan selanjutnya.

4.3.2 Instalasi Environment Node.js

Aplikasi target yang dikembangkan pada penelitian ini menggunakan *framework* Next.js. Berbeda dengan aplikasi *web* tradisional yang umumnya berjalan di atas arsitektur *server* Apache dengan bahasa pemrograman PHP, Next.js merupakan *framework full-stack* yang membutuhkan *runtime environment* JavaScript untuk mengeksekusi logika di sisi *server* (*Server-Side Rendering* dan layanan API *backend*). Oleh karena itu, tahapan fundamental pertama setelah inisialisasi *server* adalah melakukan instalasi Node.js beserta Node Package Manager (NPM) sebagai fondasi berjalannya aplikasi.

```
root@srv1481027:~# node -v
v18.19.1
root@srv1481027:~# npm -v
9.2.0
root@srv1481027:~#
```

Gambar 4. 8 Pengecekan Versi Node.js dan NPM pada Server

Gambar 4.8 mengilustrasikan proses verifikasi pasca-instalasi untuk memastikan bahwa *environment* dasar telah terpasang dengan benar pada lingkungan VPS. Pengujian dilakukan dengan mengeksekusi perintah `node -v` dan `npm -v` melalui antarmuka terminal *server*. Hasil keluaran (*output*) terminal menunjukkan bahwa Node.js (versi 18) dan NPM (versi 9) telah terintegrasi dengan variabel lingkungan (*environment variables*) sistem operasi Ubuntu. Ketersediaan Node.js ini bersifat sangat krusial, karena seluruh logika validasi keamanan pada aplikasi—termasuk proses enkripsi HMAC SHA-256 dan inisiasi Redis *Mutex Lock*—akan dieksekusi di atas *runtime* ini.

Meskipun manajer paket NPM telah tersedia secara bawaan, pemeliharaan dan *deployment* aplikasi Next.js pada penelitian ini secara spesifik menggunakan *package manager* alternatif yang lebih modern, yaitu pnpm (*Performant NPM*). Implementasi pnpm dipilih karena memiliki sistem resolusi dependensi yang lebih ketat (*strict symlinking*) serta efisiensi penggunaan ruang penyimpanan (*disk space*) yang jauh lebih optimal dibandingkan NPM konvensional. Mengingat *server* VPS



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

memiliki kapasitas penyimpanan yang terbatas, efisiensi pengelolaan *library* ini menjadi faktor pertimbangan arsitektural yang penting.

```
root@srv1481027:~# source ~/.bashrc
root@srv1481027:~# pnpm -v
10.33.0
root@srv1481027:~#
```

Gambar 4. 9 Instalasi dan Verifikasi Package Manager pnpm

Proses instalasi pnpm dikonfigurasi secara global agar *command line* pnpm dapat dikenali dan diakses dari seluruh direktori sistem. Gambar 4.9 menunjukkan hasil verifikasi setelah konfigurasi jalur sistem (*path*) diperbarui menggunakan perintah `source ~/.bashrc`. Eksekusi perintah `pnpm -v` yang menghasilkan keluaran nomor versi (versi 10) mengonfirmasi bahwa *package manager* tersebut telah aktif dan fungsional. Dengan selesainya tahapan ini, *server* Ubuntu telah memiliki kapabilitas penuh untuk mengunduh, menyusun (*build*), dan menjalankan seluruh dependensi perpustakaan pihak ketiga yang dibutuhkan oleh aplikasi target pada fase implementasi selanjutnya.

4.3.3 Instalasi dan Konfigurasi Redis Server

Pada arsitektur sistem keamanan ini, mitigasi terhadap celah kerentanan *Race Condition* dan *Replay Attack* pada jalur *callback webhook* diimplementasikan menggunakan mekanisme *Idempotency Key* berbasis *Atomic Mutex Lock*. Untuk mengeksekusi penguncian (*locking*) yang sangat cepat dan konsisten di tingkat memori, sistem memanfaatkan layanan Redis (*Remote Dictionary Server*). Redis merupakan sistem penyimpanan struktur data *in-memory* (bekerja pada level RAM) yang beroperasi secara *single-threaded*. Karakteristik ini menjadikannya solusi arsitektural yang ideal untuk menangani operasi atomik pada kondisi lalu lintas jaringan dengan tingkat konkurensi tinggi (*high concurrency*), sekaligus menghindari beban *disk I/O* (Input/Output) yang berat pada basis data relasional utama.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

Install Log[Install[redis-7.2]]

--2026-04-05 10:15:09-- https://hk1-node.bt.cn/conf/redis.conf
Resolving hk1-node.bt.cn (hk1-node.bt.cn)... 36.150.27.13
Connecting to hk1-node.bt.cn (hk1-node.bt.cn)[36.150.27.13]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 61847 (60K) [application/octet-stream]
Saving to: '/www/server/redis/redis.conf'

0K ..... 82% 505K 0s
50K ..... 100% 19.4T=0.1s

2026-04-05 10:15:10 (609 KB/s) - '/www/server/redis/redis.conf' saved [61847/61847]

Starting redis server...
Starting redis success!
Reading package lists...
Building dependency tree...
Reading state information...
sudo is already the newest version (1.9.15p5-3ubuntu5.24.04.2).
The following packages were automatically installed and are no longer required:
  libintl-perl libintl-xs-perl libmodule-find-perl libmodule-scandeps-perl
  libproc-processtable-perl libsort-naturally-perl
Use 'apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 22 not upgraded.
|-Successify --- Command executed! ---
  
```

Gambar 4. 10 Instalasi Redis Server melalui Manajemen Panel

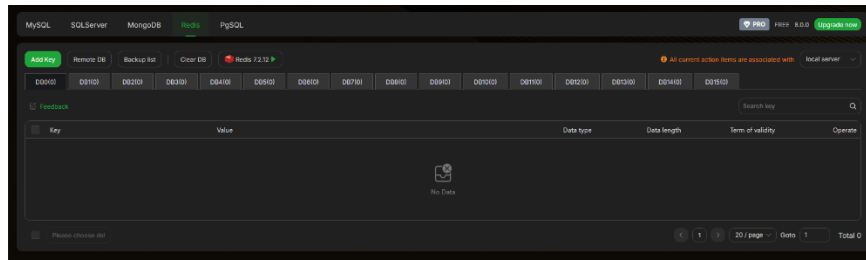
Untuk menjaga integritas ekosistem *server* dan memfasilitasi pemantauan berbasis visual, proses instalasi Redis Server versi 7.2 tidak dilakukan secara manual melalui terminal (CLI), melainkan diintegrasikan sepenuhnya melalui layanan manajemen dependensi bawaan aaPanel. Gambar 4.10 mengilustrasikan proses kompilasi dan instalasi paket Redis dari repositori resmi panel manajemen. Pendekatan ini secara otomatis mengonfigurasi layanan untuk berjalan sebagai *background service* (daemon) pada sistem operasi Ubuntu sejak *server* pertama kali melakukan *booting*, serta menetapkan *port* komunikasi standar Redis, yaitu pada *port* 6379.

Secara spesifik, area yang ditandai dengan kotak merah pada Gambar 4.10 menyoroti baris log eksekusi sistem yang menunjukkan pesan *|-Successify --- Command executed!*. Penyorotan ini bertujuan untuk memberikan bukti empiris bahwa *service* Redis telah berhasil diinstal dan berjalan secara stabil pada *environment production*. Konfirmasi operasional ini merupakan prasyarat mutlak bagi arsitektur sistem yang dibangun. Hal tersebut dikarenakan Redis memegang peran esensial sebagai basis data *in-memory* yang menangani mekanisme *distributed lock* untuk *Idempotency Key*, yang merupakan komponen mitigasi utama dalam mencegah terjadinya eksekusi data ganda saat skenario serangan *Replay Attack* dilakukan.



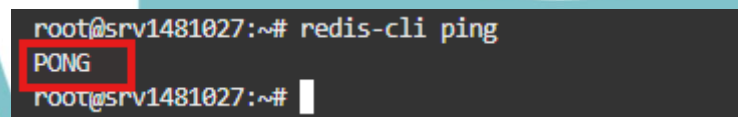
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4. 11 Antarmuka Manajemen dan Pemantauan Redis

Setelah proses instalasi berhasil diselesaikan, aaPanel menyediakan antarmuka *dashboard* khusus untuk manajemen *database in-memory* seperti yang ditunjukkan pada Gambar 4.11. Antarmuka ini menampilkan alokasi ruang penyimpanan logis mulai dari DB0 hingga DB15, lengkap dengan status *service* yang telah beroperasi aktif (*running*). Visibilitas antarmuka ini sangat esensial pada tahapan pengujian serangan (*penetration testing*) nantinya. Melalui panel ini, administrator dapat memantau secara *real-time* aktivitas pembuatan dan pelepasan *Key-Value* saat sistem secara masif membentuk *Idempotency Key* sementara dengan batas waktu tertentu (*Time-to-Live / TTL*) guna menahan duplikasi *request webhook* dari pihak yang tidak bertanggung jawab.



Gambar 4. 12 Verifikasi Responsivitas Redis Server

Meskipun pemantauan visual telah tersedia pada panel manajemen, verifikasi fungsionalitas dan tingkat responsivitas operasi internal Redis tetap perlu dilakukan secara langsung pada level sistem operasi. Pengujian interaktif ini dieksekusi melalui antarmuka baris perintah (*Command Line Interface*) dengan menggunakan instruksi `redis-cli ping`. Penanda kotak merah pada keluaran terminal di Gambar 4.12 berfungsi untuk menggarisbawahi respons `PONG` yang dikembalikan oleh *server* secara instan. Indikator visual ini sengaja ditonjolkan untuk menegaskan bahwa *request* pengujian dasar telah diterima tanpa adanya hambatan latensi internal (*bottleneck*). Bukti empiris mengenai kecepatan respons jaringan ini sangat esensial, mengingat Redis akan diintegrasikan sebagai lapisan pertahanan utama



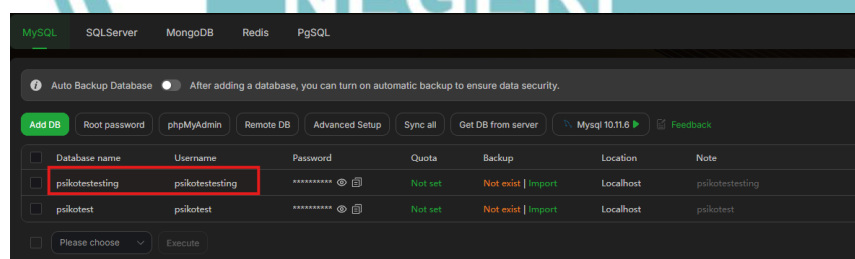
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

(Layer 2) pada *endpoint callback payment gateway* untuk mengeksekusi mekanisme *Idempotency Key* dalam hitungan milidetik guna memblokir ancaman *Replay Attack*.

4.3.4 Konfigurasi Basis Data dan Kredensial Koneksi

Basis data (*database*) merupakan komponen fundamental dalam arsitektur sistem yang berfungsi sebagai media penyimpanan persisten (*persistent storage*). Dalam sistem ini, basis data digunakan untuk mencatat riwayat transaksi, mengelola profil pengguna, serta menyimpan *payload* yang diterima dari *callback webhook* Tripay. Berbeda dengan rancangan arsitektur awal yang menempatkan basis data pada *server cPanel terpisah (remote database)*, penelitian ini mengimplementasikan arsitektur *monolithic deployment*. Pada arsitektur ini, layanan basis data diintegrasikan secara lokal (*localhost*) di dalam satu lingkungan *Virtual Private Server (VPS)* yang sama dengan aplikasi *backend Next.js*. Pemusatan infrastruktur ini merupakan strategi krusial untuk mengeliminasi latensi jaringan eksternal (*network latency*) dan meminimalkan hambatan proses (*bottleneck*). Dengan demikian, proses validasi, pencatatan data, dan pembaruan status pembayaran dapat dieksekusi secara instan, terutama saat sistem harus menghadapi pengujian serangan dengan tingkat konkurensi lalu lintas jaringan yang tinggi.



Gambar 4. 13 Antarmuka Manajemen Basis Data MySQL pada aaPanel

Merujuk pada Gambar 4.13, manajemen sistem basis data dikelola secara efisien menggunakan antarmuka visual terpusat dari aaPanel. Melalui fitur ini, proses administrasi *server* menjadi lebih sederhana tanpa mengharuskan eksekusi rentetan kueri SQL secara manual melalui terminal. Gambar tersebut mengonfirmasi bahwa layanan basis data telah beroperasi pada *localhost*



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

(arsitektur *monolithic*) dengan mengalokasikan dua skema yang terpisah, yaitu psikotest sebagai basis data operasional utama (*production*) dan psikotesttesting sebagai lingkungan pengujian (*staging environment*).

Secara spesifik, penanda kotak merah pada Gambar 4.13 menyoroti basis data psikotesttesting yang dipilih secara eksklusif untuk seluruh skenario pengujian dalam penelitian ini. Penggunaan skema terisolasi ini diimplementasikan sebagai standar keamanan (*best practice*) dalam *penetration testing* untuk mencegah intervensi terhadap data operasional. Pemisahan ini merupakan langkah yang mutlak diperlukan, mengingat pengujian efektivitas mitigasi akan memanfaatkan generator beban Apache JMeter untuk mengirimkan ratusan *request payload* duplikat (*Replay Attack*) secara serentak. Dengan mengalihkan target pengujian ke basis data *testing*, integritas basis data utama (*psikotest*) dapat sepenuhnya terjamin dari risiko penumpukan data anomali atau data sampah (*junk data*), sementara evaluasi performa sistem tetap berjalan dalam kondisi *server* yang empiris.



```
1 DATABASE_URL='mysql://psikotesttesting:user123@31.97.51.230:3306/psikotesttesting'
```

Gambar 4. 14 Konfigurasi Kredensial Basis Data pada Variabel Lingkungan (.env)

Untuk menghubungkan logika bisnis aplikasi *backend* Next.js dengan layanan basis data yang telah disiapkan pada VPS, sistem menggunakan pendekatan *Object-Relational Mapping* (ORM) dari Prisma. Kredensial koneksi ke infrastruktur *server* tersebut tidak ditulis secara langsung di dalam kode, melainkan dideklarasikan secara aman melalui fail variabel lingkungan (.env). Seperti yang ditunjukkan pada Gambar 4.14, konfigurasi ini secara spesifik memuat *connection string* DATABASE_URL yang diarahkan secara eksklusif pada skema psikotesttesting.

Secara struktural, parameter koneksi tersebut diimplementasikan menggunakan format *Uniform Resource Identifier* (URI). Format ini mencakup definisi protokol



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

basis data (mysql://), autentikasi pengguna (*username* dan kata sandi yang disensor), alamat IP server VPS target, *port* operasional (3306), serta nama basis data spesifik yang dituju. Pemisahan kredensial sensitif ke dalam fail .env ini merupakan implementasi standar keamanan perangkat lunak (*security best practice*) yang mutlak dilakukan guna mencegah risiko kebocoran data infrastruktur server ke dalam repositori *source code* publik.

Selain itu, konfigurasi melalui fail variabel lingkungan ini memberikan tingkat fleksibilitas dan skalabilitas operasional yang sangat tinggi bagi arsitektur sistem. Ketika aplikasi diinisialisasi atau saat proses sinkronisasi skema (*database migration*) dieksekusi, mesin Prisma Client akan secara otomatis membaca string koneksi ini untuk membangun *connection pool* yang stabil ke MariaDB. Melalui pendekatan dinamis ini, apabila seluruh tahapan pengujian mitigasi serangan telah selesai divalidasi dan sistem siap dirilis secara komersial (*production*), administrator hanya perlu memperbarui parameter nama basis data dari *psikotesttesting* menjadi *psikotest* di dalam fail ini, tanpa perlu melakukan modifikasi atau pembongkaran ulang pada *source code* aplikasi utama.

The screenshot shows the phpMyAdmin interface with the 'Payment' table selected. The table structure is as follows:

Field	Type	Length	Charset	Collation	Index	Comment
id	INT	11	utf8mb4	utf8mb4_0900_ai_ci	PRIMARY	
email	VARCHAR	255	utf8mb4	utf8mb4_0900_ai_ci	INDEX	
password	VARCHAR	255	utf8mb4	utf8mb4_0900_ai_ci	INDEX	
quantity	INT	11	utf8mb4	utf8mb4_0900_ai_ci	INDEX	
status	ENUM	10	utf8mb4	utf8mb4_0900_ai_ci	INDEX	
created_at	TIMESTAMP	19	utf8mb4	utf8mb4_0900_ai_ci	INDEX	
updated_at	TIMESTAMP	19	utf8mb4	utf8mb4_0900_ai_ci	INDEX	
expired_at	TIMESTAMP	19	utf8mb4	utf8mb4_0900_ai_ci	INDEX	
method	VARCHAR	255	utf8mb4	utf8mb4_0900_ai_ci	INDEX	
payload	VARCHAR	255	utf8mb4	utf8mb4_0900_ai_ci	INDEX	
payload	VARCHAR	255	utf8mb4	utf8mb4_0900_ai_ci	INDEX	

The data table shows several rows with various status values like 'FAILED', 'SUCCESS', and 'PENDING'. The interface also includes a sidebar with a tree view of the database structure.

Gambar 4. 15 Implementasi Fisik Tabel Payment pada phpMyAdmin

Setelah konfigurasi koneksi internal berhasil divalidasi dan proses migrasi skema basis data dijalankan, tahapan verifikasi struktur data tingkat lanjut dilakukan secara langsung pada level basis data menggunakan utilitas phpMyAdmin. Gambar 4.15 memperlihatkan struktur rekaman data dari tabel Payment di dalam basis data pengujian (*psikotesttesting*) yang telah terinisiasi dengan sempurna, lengkap dengan relasi dan penetapan tipe data yang presisi.



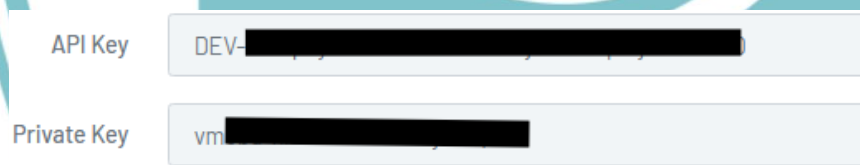
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Tabel Payment ini memiliki peran paling sentral dalam lingkup penelitian karena dirancang secara spesifik untuk menampung *payload* dinamis JSON dari respons *callback payment gateway* Tripay. Penggunaan antarmuka phpMyAdmin ini memberikan visibilitas mutlak bagi administrator untuk mengobservasi transisi status transaksi di level *backend* seperti perubahan status dari PENDING menjadi SUCCESS atau FAILED secara *real-time*. Selain itu, pemantauan pada tabel ini menjadi indikator utama keberhasilan mitigasi selama simulasi serangan *Replay Attack* dan *Webhook Spoofing* berlangsung, guna memastikan tidak ada transaksi ganda yang berhasil menembus lapisan pertahanan Redis dan HMAC.

4.3.5 Konfigurasi Endpoint pada Payment Gateway Tripay

Sebelum mengimplementasikan logika keamanan pada sisi aplikasi, sistem harus dikonfigurasi pada sisi penyedia layanan *payment gateway* (Tripay). Tahapan ini merupakan jembatan krusial untuk mensinkronisasi parameter keamanan antara *server* Tripay dengan lingkungan VPS agar proses pertukaran data notifikasi pembayaran dapat berjalan secara otomatis, valid, dan aman.



Gambar 4. 16 Konfigurasi Kredensial API pada Dasbor Tripay

Langkah pertama dalam konfigurasi ini adalah pengambilan parameter kriptografi sebagai identitas unik sistem. Seperti yang diilustrasikan pada Gambar 4.16, administrator mengekstraksi *Merchant Code*, *API Key*, dan *Private Key* melalui menu *Merchant Detail* pada mode *Sandbox* Tripay. Kredensial ini bersifat rahasia dan berfungsi sebagai "jabat tangan" digital (*digital handshake*) saat aplikasi melakukan permintaan (*request*) transaksi ke server simulator. Khusus untuk *Private Key*, parameter ini memegang peranan mutlak dalam penelitian ini karena akan digunakan sebagai kunci rahasia (*secret key*) oleh algoritma HMAC-SHA256 lokal untuk memverifikasi keaslian setiap *payload* yang diterima.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

URL Callback

<https://psiko.deltaindonesialab.com/api/payment/callback>

(*) URL untuk menerima notifikasi data transaksi

Gambar 4. 17 Konfigurasi URL Callback

Langkah selanjutnya adalah menentukan jalur pengiriman data notifikasi (*webhook callback*) dari *payment gateway* menuju sistem *backend*. Karena infrastruktur pengujian kini telah dialokasikan pada lingkungan VPS dengan domain resmi, mekanisme *tunneling* pihak ketiga (seperti Ngrok) yang lazim digunakan pada tahap pengembangan lokal kini ditiadakan untuk mengurangi latensi jaringan.

Merujuk pada Gambar 4.17, URL *Callback* pada mode *Sandbox* didaftarkan secara absolut menuju *endpoint* API aplikasi Next.js pada alamat

<https://psiko.deltaindonesialab.com/api/payment/callback>.

Konfigurasi rute ini sangat esensial, karena URL tersebut akan menjadi pintu masuk tunggal (*single point of entry*) bagi server Tripay untuk mengirimkan *payload* JSON yang berisi pembaruan status transaksi secara otomatis dan *real-time*. Mengingat pemrosesan pada jalur ini berjalan secara otomatis tanpa intervensi manual, maka kerentanan terhadap eksploitasi eksternal menjadi sangat kritis. Oleh karena itu, pada titik *endpoint* inilah seluruh skenario pengujian serangan, baik *Replay Attack* maupun *Webhook Spoofing*, akan disimulasikan. Serangan-serangan tersebut selanjutnya akan dihadapang dan dievaluasi efektivitasnya menggunakan lapisan keamanan HMAC-SHA256 serta mekanisme *distributed lock* berbasis Redis guna memastikan integritas data transaksi tetap terjaga.

```

1  TRIPAY_API_KEY=DEV [REDACTED]
2  TRIPAY_MERCHANT_CODE=T45859
3  TRIPAY_CALLBACK_URL=https://psiko.deltaindonesialab.com/api/
  payment/callback
4  TRIPAY_PRIVATE_KEY=vm [REDACTED]
```

Gambar 4. 18 Sinkronisasi Kredensial Tripay pada Variabel Lingkungan (.env)

Sebagai tahap akhir koordinasi infrastruktur jaringan, seluruh parameter yang telah



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

diekstraksi dari dasbor Tripay disinkronisasikan ke dalam *server* melalui berkas variabel lingkungan (.env). Gambar 4.18 menunjukkan implementasi kredensial tersebut pada level sistem, termasuk pembaruan alamat TRIPAY_CALLBACK_URL yang kini telah merujuk secara absolut pada domain publik server VPS.

Penyimpanan kredensial ini dilakukan dalam ruang lingkup (*environment variables*) yang terisolasi demi mencegah kebocoran data sensitif (*credential exposure*). Dengan tersimpannya data ini secara lokal di dalam *server*, modul keamanan HMAC dan mekanisme *distributed lock (Idempotency Key)* pada Redis kini memiliki referensi kunci yang valid. Kunci rahasia ini akan secara aktif digunakan oleh sistem untuk menjalankan proses komputasi kriptografi dan verifikasi integritas data secara otomatis pada setiap *request* transaksi yang masuk.

4.3.6 Implementasi Verifikasi HMAC SHA-256

Setelah seluruh infrastruktur pendukung seperti *Virtual Private Server (VPS)* dan sinkronisasi *gateway* telah disiapkan, tahapan krusial selanjutnya adalah membangun lapisan pertahanan aplikasi tingkat pertama (*First Line of Defense*). Fokus utama pada tahap ini adalah memitigasi risiko kerentanan terhadap serangan *Webhook Spoofing*. Secara teknis, serangan ini merupakan bentuk dari *Identity Theft* atau *Impersonation Attack*, di mana aktor ancaman mencoba memalsukan identitasnya sebagai entitas *payment gateway* (Tripay) untuk mengirimkan paket data (*payload*) palsu ke *endpoint* aplikasi. Tanpa adanya verifikasi identitas, sistem akan secara naif memproses data tersebut dan mengubah status transaksi menjadi berhasil (SUCCESS), yang berakibat pada kerugian finansial akibat pemberian akses layanan tanpa adanya pembayaran riil.

Untuk menanggulangi ancaman tersebut, penelitian ini mengimplementasikan metode autentikasi berbasis kriptografi menggunakan algoritma *Hash-based Message Authentication Code (HMAC)* dengan fungsi *hash* SHA-256.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

1  import crypto from "crypto";
2
3  interface TripayPayload {
4    reference?: string;
5    merchant_ref?: string;
6    total_amount?: number;
7    sign?: string;
8    [key: string]: any;
9  }

```

Gambar 4. 19 Deklarasi Struktur Data Payload

Sebagai langkah preventif awal, sistem mendeklarasikan arsitektur data melalui *Interface TripayPayload* pada bahasa pemrograman TypeScript seperti yang ditunjukkan pada Gambar 4.19. Deklarasi ini bertujuan untuk menetapkan standar struktur (*type safety*) terhadap *payload* yang diterima, mencakup parameter esensial seperti *reference*, *merchant_ref*, *total_amount*, dan *sign*. Pendekatan ini memastikan bahwa data yang masuk memiliki format yang dapat diprediksi sebelum diserahkan kepada modul pemrosesan kriptografi.

```

1  function secureCompare(
2    signatureInput: string,
3    calculatedHmac: string,
4  ): boolean {
5    try {
6      const inputBuffer = Buffer.from(signatureInput, "hex");
7      const calcBuffer = Buffer.from(calculatedHmac, "hex");
8
9      if (inputBuffer.length !== calcBuffer.length) return false;
10
11     return crypto.timingSafeEqual(inputBuffer, calcBuffer);
12   } catch (error) {
13     return false;
14   }
15 }

```

Gambar 4. 20 Implementasi Fungsi Keamanan Inti dan Mitigasi Timing Attack

Setelah arsitektur data didefinisikan, sistem menginisialisasi sebuah modul komparasi khusus bernama *secureCompare* (Gambar 4.20). Berbeda dengan komparasi *string* biasa, fungsi ini mengonversi nilai *hash* ke dalam bentuk *Buffer*



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

memori. Pendekatan ini dirancang untuk menambal dua celah keamanan kritis. Pertama, pencegahan serangan *Denial of Service* (DoS) dengan menolak langsung *input* yang panjang karakternya tidak valid. Kedua, memitigasi eksploitasi *Timing Attack* melalui pemanggilan metode `crypto.timingSafeEqual()`. Metode ini menjamin bahwa waktu komputasi yang dibutuhkan sistem untuk membandingkan *signature* akan selalu konstan (tetap), sehingga pihak penyerang tidak dapat menebak karakter *hash* yang benar berdasarkan jeda waktu respons (*response time latency*).

```

1  export function verifyTripaySignature(
2    rawBody: string,
3    bodyJson: TripayPayload,
4    headerSignature: string | null,
5  ): boolean {
6    const privateKey = process.env.TRIPAY_PRIVATE_KEY ?? "";
7    if (!privateKey) return false;
8
9    if (headerSignature) {
10     const calc = crypto
11       .createHmac("sha256", privateKey)
12       .update(rawBody)
13       .digest("hex");
14     return secureCompare(headerSignature.toLowerCase(), calc.toLowerCase());
15   }
16 }

```

Gambar 4. 21 Implementasi Validasi Jalur Utama (HTTP Header Signature)

Gerbang keamanan utama dijelaskan pada Gambar 4.21, di mana fungsi `verifyTripaySignature` melakukan validasi secara langsung terhadap *signature* yang dikirimkan oleh Tripay melalui HTTP Header. Sistem memuat kunci rahasia (`TRIPAY_PRIVATE_KEY`) yang bersifat esensial sebagai *salt*. Keunggulan pada blok ini adalah pemanfaatan variabel `rawBody`, yaitu data mentah HTTP sebelum dilakukan *parsing* menjadi objek JSON. Penggunaan data mentah ini krusial karena perubahan sekecil apa pun—seperti satu spasi pada proses transmisi—akan mengakibatkan perbedaan nilai *hash* secara total (*Avalanche Effect*). Hasil kalkulasi lokal ini kemudian divalidasi keamanannya menggunakan fungsi `secureCompare`.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

1  if (bodyJson.sign) {
2    const str = `${bodyJson.reference}${bodyJson.merchant_ref}${bodyJson.n.total_amount}`;
3    const calc = crypto
4      .createHmac("sha256", privateKey)
5      .update(str)
6      .digest("hex");
7    return secureCompare(bodyJson.sign.toLowerCase(), calc.toLowerCase
8  }
  
```

Gambar 4. 22 Implementasi Jalur Cadangan dan Kebijakan Default Deny

Sebagai bentuk penerapan prinsip *Defense in Depth* (pertahanan berlapis), sistem juga menyediakan jalur validasi cadangan (*redundancy mechanism*) seperti yang diilustrasikan pada Gambar 4.22. Dalam skenario anomali di mana *signature* pada *header* gagal terbaca di level proksi jaringan, sistem akan mengekstraksi parameter statis di dalam *body payload*, lalu melakukan *hashing* ulang secara mandiri. Terakhir, sistem menerapkan prinsip keamanan *Default Deny* pada baris paling akhir dengan mengeksekusi perintah `return false`. Hal ini secara otomatis menggagalkan segala upaya serangan kelalaian (*Omission Attack*), di mana penyerang mencoba menembus sistem dengan sengaja tidak menyertakan parameter *signature* sama sekali.

4.3.7 Implementasi Idempotency Key dan Redis Mutex Lock

Lapisan pertahanan kedua ini dirancang secara spesifik untuk menangani aspek integritas data pada kondisi konkurensi tinggi melalui mekanisme *Idempotency Key*. Ancaman utama yang dimitigasi adalah *Replay Attack*, sebuah teknik di mana penyerang menangkap paket *webhook* yang valid dan mengirimkannya kembali secara berulang dalam hitungan milidetik. Dalam sistem finansial, kegagalan menangani *request* duplikat ini berakibat fatal pada fenomena *Double Spending* atau penggandaan saldo ilegal.

Tantangan teknis utama dalam arsitektur ini adalah keterbatasan basis data relasional (MySQL) dalam melakukan penguncian data secara instan. MySQL beroperasi pada level *Disk I/O* yang memiliki latensi lebih tinggi dibandingkan memori akses acak (RAM). Pada skenario serangan dengan ratusan *request* per



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

detik, terdapat celah waktu (*race condition window*) di mana dua *request* atau lebih dapat melewati pengecekan status *PENDING* secara bersamaan sebelum proses *write* pertama selesai dilakukan. Oleh karena itu, penelitian ini mengimplementasikan *Distributed Lock* menggunakan Redis yang beroperasi sepenuhnya di dalam memori utama (RAM) untuk menjamin atomisitas transaksi.

```

1  import { Redis } from "ioredis";
2
3  const redisUrl = process.env.REDIS_URL || "redis://127.0.0.1:6379";
4  export const redis = new Redis(redisUrl);
5
6  redis.on("connect", () => console.log("[Redis] Berhasil terhubung ke server."));
7  redis.on("error", (err) =>
8    console.error("[Redis] Gagal terhubung ke server:", err),
9  );

```

Gambar 4. 23 Inisialisasi Koneksi ke Server Redis

Tahap awal implementasi dimulai dengan mengonfigurasi koneksi antara aplikasi Next.js dan server Redis melalui pustaka *ioredis*, seperti yang ditunjukkan pada Gambar 4.23. Berdasarkan konfigurasi pada *environment variables* (.env), parameter *REDIS_URL* diarahkan pada *localhost* (127.0.0.1) di dalam VPS. Strategi optimasi infrastruktur ini bertujuan untuk meniadakan lompatan jaringan (*network hop*) ke server eksternal, sehingga komunikasi antara aplikasi dan sistem pengunci dapat berjalan dengan *round-trip time* (RTT) yang mendekati nol. Objek *redis* yang diekspor bersifat *singleton* dan dilengkapi pemantauan status koneksi (*event listener*), memastikan bahwa seluruh modul keamanan menggunakan *instance* yang sama untuk menjaga konsistensi status gembok (*lock state*).



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

1  import { redis } from "@lib/redis";
2  import crypto from "crypto";
3
4  export async function checkIdempotency(
5    merchantRef: string,
6    rawBody: string,
7  ): Promise<boolean> {
8    const idempotencyKey = `idempotency:tripay:${merchantRef}`;
9
10   const payloadFingerprint = crypto
11     .createHash("sha256")
12     .update(rawBody)
13     .digest("hex");
14
15   const isLocked = await redis.set(
16     idempotencyKey,
17     payloadFingerprint,
18     "EX",
19     86400,
20     "NX",
21   );

```

Gambar 4. 24 Implementasi Payload Fingerprint dan Atomic Operation

Inti dari logika mitigasi *Race Condition* diimplementasikan pada fungsi `checkIdempotency` sebagaimana diilustrasikan pada Gambar 4.24. Sebelum melakukan penguncian, sistem terlebih dahulu menciptakan "sidik jari" (*fingerprint*) dari *payload* mentah menggunakan algoritma *hashing* SHA-256. Sidik jari ini kemudian disimpan sebagai nilai di dalam Redis bersamaan dengan nomor referensi transaksi.

Sistem memanfaatkan perintah `redis.set` yang bersifat *Atomic*, artinya operasi pengecekan dan penguncian dilakukan dalam satu kesatuan waktu yang tidak dapat diinterupsi oleh proses lain. Parameter `NX` (*Not Exists*) bertindak sebagai *Mutual Exclusion* (*Mutex*); perintah ini hanya akan berhasil jika kunci tersebut belum ada di memori. Dalam simulasi serangan serentak, Redis menjamin secara absolut bahwa hanya satu *request* tercepat yang mendapatkan izin proses. Selain itu, parameter `EX 86400` menetapkan *Time-to-Live* (TTL) selama 24 jam untuk mencegah terjadinya *Deadlock* permanen apabila terjadi *crash* pada server.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

1  if (!isLocked) {
2      const existingFingerprint = await redis.get(idempotencyKey);
3
4      if (existingFingerprint === payloadFingerprint) {
5          console.warn(
6              `⚠️ [REPLAY BLOCKED] Payload duplikat ditolak: ${merchantRef}`,
7          );
8      } else {
9          console.error(
10             `🔥 [HYBRID ATTACK] Payload dimodifikasi pada ref: ${merchantRef}!`,
11         );
12     }
13     return false;
14 }

```

Gambar 4. 25 Mekanisme Forensik dan Mitigasi Hybrid Attack

Pada Gambar 4.25, sistem mengelola *request* yang terdeteksi gagal mendapatkan gembok Redis (*isLocked*). Alih-alih sekadar menolak permintaan, sistem melakukan langkah audit forensik dengan membandingkan *fingerprint* yang baru masuk dengan *fingerprint* yang sudah terkunci di memori. Jika kedua nilai tersebut identik, sistem mengklasifikasikannya sebagai *Replay Attack* murni. Namun, jika *fingerprint* berbeda pada nomor referensi yang sama, sistem mencatatnya sebagai peringatan *Hybrid Attack* (manipulasi *payload*). Mekanisme deteksi dini ini menghasilkan *audit trail* yang komprehensif pada sistem *log* (konsol), sembari langsung menggugurkan proses eksekusi dengan mengembalikan nilai *false* tanpa membebani performa basis data MySQL.

4.3.8 Integrasi Keamanan pada Endpoint Webhook Callback

Setelah modul verifikasi kriptografi HMAC dan mekanisme penguncian *Redis Mutex* selesai dikembangkan secara modular, tahapan krusial selanjutnya adalah mengintegrasikan seluruh lapisan pertahanan tersebut ke dalam *endpoint* API utama. Strategi ini menerapkan prinsip *Defense in Depth*, di mana sistem keamanan disusun secara berlapis untuk memastikan bahwa kegagalan atau lolosnya satu lapisan tidak akan langsung meruntuhkan seluruh integritas sistem. *Endpoint* ini berada pada rute `/api/payment/callback` yang berfungsi sebagai *listener* pasif untuk menerima setiap *payload* HTTP POST dari *payment gateway*. Integrasi ini dirancang dengan alur kerja linier yang sangat ketat. Setiap permintaan yang masuk wajib melewati tiga gerbang validasi utama sebelum diizinkan menyentuh logika



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

bisnis atau melakukan manipulasi pada basis data MySQL.

```

1 import { NextRequest, NextResponse } from "next/server";
2 import { prisma } from "@lib/prisma";
3 import { redis } from "@lib/redis";
4 import { verifyTripaySignature } from "@lib/security/hmacVerifier";
5 import { checkIdempotency } from "@lib/security/redisIdempotency";
6
7 export async function POST(req: NextRequest) {
8   const ip = req.headers.get("x-forwarded-for") || "unknown_ip";
9
10  try {
11    const rateLimitKey = `rate_limit:callback:${ip}`;
12    const requestCount = await redis.incr(rateLimitKey);
13
14    if (requestCount === 1) {
15      await redis.expire(rateLimitKey, 60);
16    }
17
18    if (requestCount > 50) {
19      console.warn(`🚨 [DDoS BLOCKED] Terlalu banyak request dari IP: ${ip}`);
20    };
21    return NextResponse.json(
22      { success: false, message: "Too many requests" },
23      { status: 429 },
24    );
25  }

```

Gambar 4. 26 Implementasi Gerbang Layer 0 (Rate Limiting) pada Webhook

Berdasarkan Gambar 4.26, gerbang keamanan paling luar (*Layer 0*) difokuskan pada mitigasi eksploitasi ketersediaan layanan (*Denial of Service/DDoS*) dan *Spamming*. Sistem mengekstraksi alamat IP pengirim dan memanfaatkan Redis untuk menghitung jumlah permintaan secara *real-time*. Ambang batas (*threshold*) ditetapkan pada 50 permintaan per menit untuk setiap alamat IP. Jika permintaan melampaui batas tersebut, sistem akan langsung menolak koneksi dengan kode status HTTP 429 *Too Many Requests*. Lapisan ini sangat krusial untuk menjaga stabilitas *server VPS* saat menghadapi skenario pengujian beban tinggi (*stress testing*) menggunakan instrumen pengujian.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

1  const rawBody = await req.text();
2  const body = JSON.parse(rawBody);
3  const { reference, status, total_amount, merchant_ref } = body;
4
5  if (!reference || !merchant_ref || !status) {
6    return NextResponse.json(
7      { success: false, message: "Payload incomplete" },
8      { status: 400 },
9    );
10 }
11
12 const headerSignature = req.headers.get("x-callback-signature");
13
14 console.log(
15   `\n[Webhook HTTP POST] Menerima payload untuk: ${merchant_ref}`,
16 );
17 console.log(`[Signature Header] ${headerSignature}`);
18
19 const isAuthenticated = verifyTripaySignature(rawBody, body, headerSignature);
20 if (!isAuthenticated) {
21   console.error(
22     "❌ [SECURITY BREACH] Invalid HMAC Signature terdeteksi! Akses ditolak.",
23   );
24   return NextResponse.json(
25     { success: false, message: "Invalid signature" },
26     { status: 400 },
27   );
28 }

```

Gambar 4. 27 Implementasi Gerbang Validasi Kriptografi HMAC pada Webhook

Jika permintaan dinyatakan dalam batas wajar, permintaan tersebut akan masuk ke gerbang pertama yang berfokus menahan serangan *Webhook Spoofing* (Gambar 4.27). Sistem terlebih dahulu mengekstraksi data mentah menggunakan `req.text()` untuk mendapatkan `rawBody`. Langkah ini esensial karena fungsi *hashing* HMAC sangat sensitif terhadap perubahan karakter; konversi ke objek JSON yang terlalu dini dapat merusak struktur *string* asli dan menyebabkan kegagalan validasi. Sistem kemudian memanggil fungsi `verifyTripaySignature`. Jika hasil pengujian bernilai *false*, *endpoint* akan memutus koneksi dan memberikan respons HTTP 400 *Bad Request*. Hal ini menjamin bahwa hanya *payload* yang benar-benar berasal dari Tripay yang dapat melanjutkan eksekusi.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

1  const isSafeToProcess = await checkIdempotency(merchant_ref, rawBody);
2  if (!isSafeToProcess) {
3    console.warn(
4      `⚠️ [SECURITY WARNING] Duplicate Request dicegah untuk transaksi: ${mercha
5      nt_ref}`,
6    );
7    return NextResponse.json(
8      { success: false, message: "Conflict / Duplicate Request Detected" },
9      { status: 409 },
10     );
11   }
12   console.log(
13     "✅ [Security] Transaksi lolos verifikasi berlapis (HMAC & Redis).",
14   );

```

Gambar 4. 28 Implementasi Gerbang Validasi Idempotency Lock pada Webhook

Apabila permintaan dinyatakan otentik, permintaan tersebut akan langsung dihadapkan pada gerbang keamanan kedua yang dirancang untuk memitigasi serangan *Replay Attack* dan *Race Condition*. Seperti yang ditunjukkan pada Gambar 4.28, sistem memanggil fungsi `checkIdempotency` dengan membawa parameter `merchant_ref` dan `rawBody`. Jika fungsi Redis ini merespons dengan nilai *false*, hal tersebut mengindikasikan terdapat permintaan duplikat atau upaya pengiriman ulang *payload* yang sudah diproses. Sistem akan menolak permintaan tersebut dengan status HTTP 409 *Conflict*, sehingga mencegah terjadinya pembaruan ganda pada basis data yang dapat merugikan penyedia layanan.

```

1  const paymentId = Number(merchant_ref.split("-")[1]);
2  const payment = await prisma.payment.findUnique({
3    where: { id: paymentId },
4  });
5
6  if (!payment) {
7    return NextResponse.json(
8      { success: false, message: "Payment not found" },
9      { status: 404 },
10     );
11   }
12
13   if (payment.status === "SUCCESS") {
14     console.warn(
15       `🚫 [DB BLOCKED] Payment ID ${paymentId} sudah LUNAS di Database!
16       Replay Attack (TTL Expiry) digagalkan.`,
17     );
18     return NextResponse.json(
19       { success: true, message: "Payment already processed" },
20       { status: 200 },
21     );
22   }

```

Gambar 4. 29 Implementasi Gerbang Layer 3 (Database Absolute Lock) pada Webhook



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Apabila sebuah permintaan berhasil melewati lapisan *in-memory* Redis, sistem tidak serta-merta langsung melakukan pembaruan data. Mengantisipasi celah kerentanan arsitektural bawaan pada sistem Redis di mana kunci *Idempotency* dapat hangus setelah melewati batas waktu tertentu (*Time-to-Live / TTL Expiry*), sistem menerapkan lapisan pertahanan pamungkas (Layer 3) dengan merujuk pada *Source of Truth* atau sumber kebenaran absolut di basis data relasional (MySQL).

Berdasarkan Gambar 4.29, sistem mengekstraksi parameter ID Pembayaran dari nilai `merchant_ref` dan melakukan pencarian data tunggal secara spesifik menggunakan metode `findUnique` melalui mesin Prisma ORM. Pendekatan pembacaan data ini bertujuan untuk memverifikasi kondisi riil transaksi sesaat sebelum proses penulisan (*write*) dilakukan. Jika dari hasil kueri ditemukan bahwa status transaksi di basis data telah tercatat sebagai "SUCCESS", sistem akan langsung memutus rantai eksekusi kode (*early return pattern*).

Interupsi ini merupakan bentuk mitigasi langsung terhadap ancaman *Replay Attack* tertunda, di mana penyerang sengaja menunggu hingga penjagaan memori Redis kedaluwarsa. Sistem kemudian mencetak log peringatan keamanan [DB BLOCKED] sebagai jejak audit (*audit trail*) bagi administrator. Menariknya, pada blok pengecekan ini sistem dirancang untuk mengembalikan respons HTTP 200 OK ke server Tripay. Taktik pengembalian status sukses palsu ini adalah bentuk rekayasa *Graceful Degradation*, yang bertujuan agar algoritma *retry policy* (kebijakan pengiriman ulang) pada *payment gateway* berhenti mengeksekusi *webhook* lanjutan, sehingga menghemat utilisasi jaringan tanpa sedikit pun menembus integritas basis data.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

1  const tripayStatus = status.toUpperCase();
2  let newStatus = "PENDING";
3  if (["PAID", "SETTLED", "SUCCESS"].includes(tripayStatus))
4    newStatus = "SUCCESS";
5  else if (["EXPIRED", "FAILED", "REFUND", "UNPAID"].includes(tripayStat
6    us))
7    newStatus = "FAILED";
8
9  const updatedPayment = await prisma.payment.update({
10    where: { id: paymentId },
11    data: {
12      status: newStatus,
13      reference,
14      payload: body,
15      paidAt: newStatus === "SUCCESS" ? new Date() : null,
16    },
17  });

```

Gambar 4. 30 Implementasi Eksekusi Pembaruan Basis Data Persisten

Hanya permintaan tunggal yang telah terbukti secara konklusif melewati ketiga gerbang keamanan—yakni lolos verifikasi kriptografi HMAC, tidak terkendala *Mutex Lock* Redis, dan belum dilunasi pada basis data absolut—yang diberikan hak akses (privilege) untuk mengeksekusi logika pembaruan data persisten. Berdasarkan Gambar 4.30, tahap pertama dalam proses ini adalah standarisasi status (*State Standardization*). Mengingat penyedia layanan Tripay dapat mengembalikan berbagai terminologi status keberhasilan (seperti "PAID", "SETTLED", atau "SUCCESS"), sistem melakukan pemetaan dinamis untuk mengonversi nilai tersebut ke dalam satu format baku, yaitu "SUCCESS", guna menjaga konsistensi skema lokal. Selanjutnya, fungsi `prisma.payment.update` dipanggil untuk mengeksekusi kueri pembaruan (*Update Query*) secara aman.

Penggunaan antarmuka Object-Relational Mapping (ORM) pada tahap ini tidak hanya menyederhanakan sintaksis, tetapi juga bertindak sebagai lapis keamanan tambahan yang meniadakan risiko serangan Injeksi SQL (*SQL Injection*). Pembaruan ini merekam tiga komponen vital: status transaksi terkini, penyematan seluruh muatan JSON asli (payload) ke dalam kolom teks sebagai cadangan bukti forensik jika terjadi sengketa (*dispute*), serta pencatatan presisi penanda waktu (*timestamp*) menggunakan `new Date()` pada atribut `paidAt` untuk keperluan analisis kronologi transaksi pembayaran.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

1  if (newStatus === "SUCCESS") {
2    await prisma.testAttempt.create({
3      data: {
4        userId: payment.userId!,
5        testTypeId: payment.testTypeId,
6        paymentId: payment.id,
7        companyId: payment.companyId,
8        status: "RESERVED",
9      },
10   });
11   console.log(
12     `[System Info] Akses Token Tes berhasil dicetak untuk Transaksi ${paymentId}.`,
13   );
14 }
15
16 return NextResponse.json({ success: true, payment: updatedPayment });
17 } catch (err: any) {
18   console.error("[System Error] Terjadi kesalahan pada proses webhook:", err);
19   return NextResponse.json(
20     { success: false, message: "Internal server error" },
21     { status: 500 },
22   );
23 }
24 }

```

Gambar 4. 31 Implementasi Pencetakan Akses Token dan Penyelesaian Siklus

Sebagai tahap akhir dari siklus *webhook*, pembaruan status transaksi menjadi SUCCESS akan secara otomatis memicu pembuatan entitas baru pada tabel TestAttempt seperti yang diilustrasikan pada Gambar 4.31. Proses pencetakan akses ujian ini dilakukan dengan tingkat kepercayaan tinggi (*high confidence*), karena sistem telah memastikan secara programatik bahwa data tersebut adalah otentik (lolos HMAC) dan bukan merupakan hasil penggandaan transaksi (lolos Redis).

Setelah seluruh proses logika bisnis selesai, *endpoint* mengembalikan respons HTTP 200 OK dengan format JSON kepada server Tripay, yang menandakan bahwa proses *callback* telah dieksekusi dengan sempurna. Sebagai tambahan, sistem juga dilengkapi dengan blok penanganan *error* (catch) komprehensif yang akan mengembalikan status HTTP 500 *Internal Server Error* jika terjadi kegagalan interupsi di luar logika utama, memastikan server tidak mengalami *crash* yang fatal.

4.3.9 Deployment dan Inisialisasi Server Aplikasi

Setelah seluruh arsitektur kode dan lapisan keamanan (HMAC, *Redis Mutex*, dan *Rate Limiting*) diimplementasikan secara komprehensif, tahapan krusial terakhir sebelum masuk ke fase pengujian adalah melakukan *deployment* atau perilisan aplikasi ke lingkungan *server* produksi (*Virtual Private Server*). Proses ini bertujuan untuk mengalihkan eksekusi sistem dari lingkungan pengembangan lokal



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

(*localhost*) menuju infrastruktur publik yang dapat diakses secara global. Hal ini memastikan bahwa simulasi pengujian menggunakan instrumen *stress-test* nantinya dapat berjalan pada kondisi jaringan riil dengan segala variabel latensinya.

```

Local server
Done in 28s using prpm v10.33.0
root@srv1481027:/www/wwwroot/psychotest-app# npm exec prpm run build
> app-psychotest@0.1.0 build /www/wwwroot/psychotest-app
> prisma generate && next build

warn The configuration property 'package.json#prisma' is deprecated and will be removed in Prisma 7. Please migrate to a Prisma
config file (e.g., 'prisma.config.ts').
For more information, see: https://pris.ly/prisma-config

Environment variables loaded from .env
Prisma schema loaded from prisma/schema.prisma

✓ Generated Prisma Client (v6.19.0) to ./.node_modules/.prisma/client@6.19.0_typescript@5.9.3_typescript@5.9
.3/node_modules/.prisma/client in 1.60s

Start by importing your Prisma Client (See: https://pris.ly/d/importing-client)
Tip: Want to turn off tips and other hints? https://pris.ly/tip-4-nohints

▲ Next.js 15.5.7
- Environments: .env

Creating an optimized production build ...
✓ Compiled successfully in 3.6min
  Skipping linting
  ✓ Checking validity of types
  ✓ Collecting page data
  ✓ Generating static pages (85/85)
  ✓ Collecting build traces
  ✓ Finalizing page optimization

Route (app)      Size  First Load JS
┌ ○ /            4.36 kB           161 kB

```

Gambar 4. 32 Kompilasi Aplikasi Next.js pada Lingkungan Server

Langkah pertama dalam proses *deployment* adalah melakukan kompilasi kode sumber. Seperti yang ditunjukkan pada Gambar 4.32, administrator mengeksekusi perintah kompilasi melalui terminal VPS. Proses ini menginstruksikan kerangka kerja Next.js untuk melakukan serangkaian optimasi tingkat lanjut, termasuk minifikasi kode, *tree-shaking* untuk membuang modul fungsionalitas yang tidak terpakai, dan transformasi kode TypeScript menjadi JavaScript murni yang dapat dieksekusi dengan cepat oleh mesin Node.js. Selain itu, sistem juga melakukan pra-render (*pre-rendering*) halaman statis guna memangkas waktu muat (*load time*). Indikator *Compiled successfully in 3.6min* menegaskan bahwa seluruh struktur aplikasi—termasuk modul keamanan HMAC dan integrasi Redis yang telah ditambahkan—bebas dari *syntax error*. Proses kompilasi ini sangat krusial untuk memastikan aplikasi memiliki performa dan efisiensi penggunaan memori yang optimal saat nantinya menghadapi beban tinggi (*high throughput*) pada fase simulasi serangan.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Site modification [psychotestapp] -- Time added [2026-04-21 12:22:09]

Project config	Path	/www/wwwroot/psychotest-app	
Domain Manager	Name	psychotest-app	
Mapping	Run opt	start [next start]	* Get the startup mode in package.json
URL rewrite	Port	3000	* Port of the project
Config	User	www	* No special requirements, choose www user
SSL	Node	v24.14.0	* Choose the right Node version, Install other
Load status	Remark	psychotest-app	
Service status	Boot	☒ Follow the system to start the service	
Module			
Project log			
Website log			

[Save](#)

- [Run opt]: The scripts list in package.json is read by default, or you can select the [Custom Command] option to manually enter the start command
- [Custom start]: You can select the startup file or directly enter the startup command. Supported startup methods: npm/node/pm2/yarn
- [Port]: The wrong port will lead to access to 502, if you don't know the port, you can change to the correct port after starting
- [User]: For security reasons, the www user is used by default to run, and root user running may bring security risks

Gambar 4. 33 Inisialisasi Layanan Node.js dan Manajemen Port

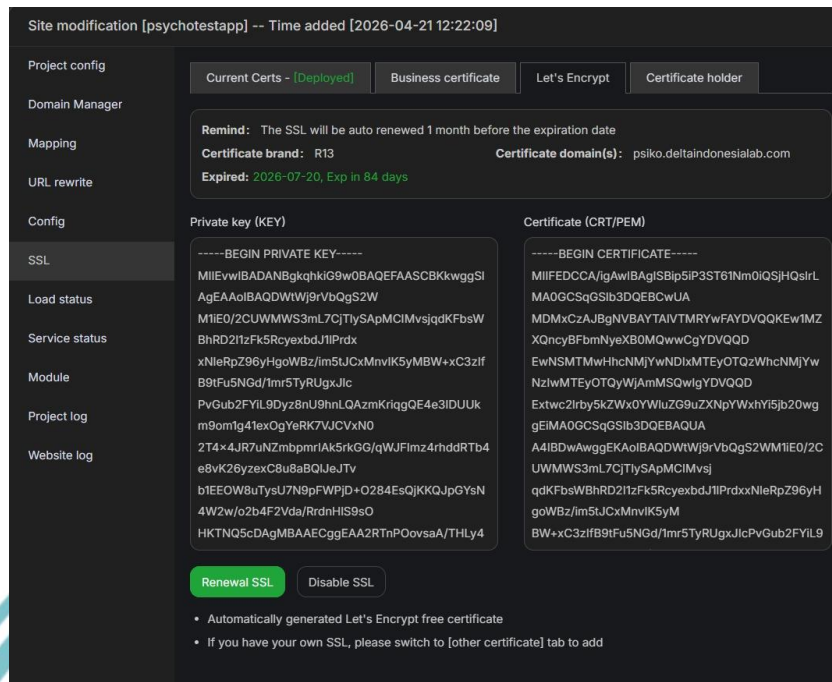
Setelah aplikasi berhasil dikompilasi, tahapan berikutnya adalah menginisialisasi sistem sebagai layanan latar belakang (*background daemon service*). Hal ini mutlak diperlukan agar aplikasi Next.js tetap aktif beroperasi (*always-on*) meskipun administrator telah menutup sesi koneksi terminal (SSH). Berdasarkan Gambar 4.33, sistem memanfaatkan manajer proyek bawaan panel kontrol untuk menjalankan perintah *startup* menggunakan *runtime* Node.js v24.14.0. Fitur manajer proses ini juga bertindak sebagai mekanisme *fail-safe* yang akan secara otomatis menghidupkan ulang (*auto-restart*) aplikasi apabila terjadi *crash* atau interupsi sistem di luar dugaan. Layanan aplikasi ini secara sengaja diisolasi untuk berjalan secara internal hanya pada *port* 3000. Pengisolasian ini merupakan praktik penguatan infrastruktur (*infrastructure hardening*) yang esensial. Dengan memblokir eksposur langsung ke *port* publik, aktor ancaman tidak dapat melakukan *bypass* untuk mengakses aplikasi secara langsung, melainkan dipaksa untuk melewati gerbang penjagaan *Reverse Proxy*.



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4. 34 Konfigurasi Transport Layer Security (SSL/TLS) dan Force HTTPS

Gerbang terakhir sekaligus fondasi perlindungan terluar dalam arsitektur *deployment* ini adalah pengamanan jalur komunikasi data, seperti yang diilustrasikan pada Gambar 4.34. Administrator mengonfigurasi *domain* publik aplikasi dan memasang sertifikat keamanan SSL/TLS (*Secure Sockets Layer / Transport Layer Security*) yang diterbitkan oleh otoritas sertifikat.

Sebagai langkah penyempurnaan infrastruktur keamanan, sistem mengaktifkan fitur *Force HTTPS* secara ketat untuk menolak semua permintaan HTTP biasa yang tidak terenkripsi. Dari perspektif keamanan informasi, implementasi HTTPS ini berfungsi sebagai perlindungan Lapis Nol (*Transport Layer Protocol Security*). Sebelum *payload webhook* dari Tripay mencapai lapisan validasi HMAC di dalam aplikasi, seluruh lalu lintas data yang melintasi internet publik telah terlebih dahulu dienkripsi secara asimetris. Mekanisme ini secara langsung menjamin pemenuhan prinsip kerahasiaan (*confidentiality*) dan integritas (*integrity*) data. Hasilnya, celah kerentanan terhadap serangan *Packet Sniffing* atau *Man-in-the-Middle* (MitM) di mana aktor ancaman mencoba menguping jaringan atau mencuri *payload JSON* sebelum tiba di *endpoint VPS*—dapat dimitigasi sepenuhnya.



4.4 Pengujian

4.4.1 Deskripsi Pengujian

Pengujian sistem keamanan pada penelitian ini bertujuan untuk mengukur, menganalisis, dan mengevaluasi efektivitas arsitektur pertahanan berlapis (*Defense-in-Depth*) yang telah diimplementasikan pada *endpoint callback payment gateway* aplikasi PsikoTest. Evaluasi difokuskan pada analisis perbandingan (komparasi) secara empiris antara sistem konvensional yang tidak memiliki mekanisme keamanan spesifik (sistem rentan), dengan sistem yang telah diperkuat menggunakan verifikasi kriptografi HMAC-SHA256, *Idempotency Key* berbasis Redis, serta *Database Absolute Lock* (sistem *secure*).

Skenario pengujian dirancang secara komprehensif untuk menyimulasikan berbagai vektor ancaman siber yang menargetkan kerentanan logika bisnis (*Business Logic Vulnerability*) pada integrasi *Application Programming Interface* (API) pihak ketiga. Fokus utama pengujian mencakup mitigasi terhadap ancaman pemalsuan identitas dan manipulasi panjang karakter komputasi (*Buffer Mismatch*), manipulasi data transmisi (*Webhook Spoofing*), serangan pengiriman ulang paket data yang sah (*Replay Attack*), eksploitasi celah waktu pemrosesan (*Race Condition*), serta evaluasi stabilitas kinerja server dari potensi kelumpuhan layanan (efek *Denial of Service*) akibat lonjakan beban konkurensi yang masif.

Guna mereplikasi serangan secara presisi dan terukur, pengujian ini memanfaatkan beberapa instrumen keamanan siber berstandar industri. Manipulasi *payload* JSON dan modifikasi parameter Header HTTP untuk skenario *Spoofing* dan *Omission* dieksekusi menggunakan Burp Suite dan Postman API. Sementara itu, pengujian ketahanan beban tinggi (*High Concurrency*) untuk membuktikan keandalan *in-memory data store* (Redis) dalam menahan serangan *Replay Attack* secara masif dieksekusi menggunakan alat uji otomatisasi Apache JMeter. Seluruh indikator keberhasilan mitigasi divalidasi berdasarkan interupsi kode status HTTP, kemunculan jejak audit pada log server (*Security Breach Logs*), serta konsistensi status akhir transaksi pada tabel Payment di basis data. Untuk memberikan batasan yang jelas mengenai parameter yang diuji, berikut adalah tabel komparasi

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



spesifikasi arsitektur antara sistem rentan dan sistem *secure* yang menjadi subjek utama dalam pengujian ini:

Tabel 4. 4 Perbandingan Arsitektur API Callback Payment Gateway

Parameter Keamanan	Sistem Rentan (<i>Vulnerable System</i>)	Sistem <i>Secure</i> (<i>Defense-in-Depth</i>)
Validasi <i>Header</i> & Integritas Data	Tidak ada. Sistem langsung mengekstrak <i>body</i> JSON (status: PAID) tanpa memverifikasi pengirim.	Verifikasi ketat menggunakan algoritma HMAC SHA-256 dan pengecekan panjang variabel (<i>Buffer Length Check</i>).
Penyimpanan Memori & <i>Idempotency</i>	Tidak ada. Setiap <i>request</i> yang masuk langsung membebani antrean koneksi basis data.	Menggunakan <i>In-Memory Database</i> (Redis) sebagai <i>Idempotency Key</i> untuk menyaring <i>request</i> duplikat di <i>layer</i> terluar.
Penguncian Basis Data (<i>Concurrency</i>)	Tidak ada. Rentan terhadap <i>Race Condition</i> yang memicu pencetakan Token Tes ganda.	Menggunakan sistem <i>Absolute Database Lock / Transaction</i> agar entitas data tetap tersinkronisasi dan mencegah duplikasi.
Ketahanan Beban Komputasi (<i>High Concurrency</i>)	Tidak ada. Serangan <i>Replay Attack</i> bervolume masif akan membanjiri koneksi basis data, menyebabkan latensi tinggi, <i>timeout</i> , hingga memicu kelumpuhan server (<i>Resource Exhaustion</i>).	Mampu menahan lonjakan <i>request</i> berkat penolakan instan di lapisan RAM (Redis), menjaga <i>throughput</i> stabil dan mencegah kelelahan sumber daya.

Berdasarkan Tabel 4.4 di atas, pengujian dirancang untuk mengekspos kelemahan fundamental pada arsitektur "Sistem Rentan" yang hanya mengandalkan logika pemrosesan dasar tanpa penyaring keamanan. Ketiadaan lapisan otentikasi

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

membuat sistem ini sangat rentan terhadap manipulasi muatan (*Spoofing*), sementara absennya penanganan konkurensi memungkinkan penyerang mengeksploitasi *Race Condition*. Sebaliknya, "Sistem *Secure*" diuji untuk membuktikan bahwa kombinasi tiga pilar utama (HMAC, Redis, dan *Database Lock*) mampu membentuk mekanisme pertahanan proaktif. Sistem *secure* tidak hanya divalidasi kemampuannya dalam menolak anomali data (melalui HMAC) dan lalu lintas duplikat (melalui Redis), tetapi juga diukur keandalannya dalam mempertahankan stabilitas operasional (*availability*) tanpa mengalami interupsi *server* saat berada di bawah tekanan serangan yang masif.

4.4.2 Prosedur Pengujian

4.4.2.1 Spesifikasi Lingkungan Pengujian

Spesifikasi lingkungan pengujian yang digunakan dalam penelitian ini dijabarkan secara rinci pada Tabel 4.2. Tabel tersebut memuat konfigurasi utama yang meliputi sistem operasi, jumlah *core* CPU, kapasitas RAM, serta ruang penyimpanan yang dialokasikan. Penelitian ini menggunakan dua lingkungan utama yang saling terhubung, yaitu mesin inang (*host machine*) yang berperan sebagai penyerang (*attacker*) sekaligus klien (*client*), serta *Virtual Private Server* (VPS) yang berfungsi sebagai server target (*target server*) pada infrastruktur awan (*cloud*).

Tabel 4. 5 Spesifikasi Virtual Machine

Nama VM	OS	CPU	Role
Laptop Attacker (Host)	Windows 11	6 Core @ 2.10GHz Storage 512 GB RAM 12 GB	Pengujian serangan (Burp Suite, JMeter) & Klien
Target Server (VPS)	Ubuntu Server 24.04.1 LTS	CPU: 1 Core RAM: 4 GB Storage: 50 GB	Web Server & API, Basis Data (MariaDB) + Atomic Mutex Lock (Redis)

Berdasarkan Tabel 4.5, perangkat *host machine* menggunakan sistem operasi



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Windows 11 dengan spesifikasi komputasi 6 *core* CPU @ 2.10 GHz, RAM 12 GB, dan ruang penyimpanan 512 GB. Kapasitas RAM dan CPU yang memadai pada perangkat ini sangat krusial untuk memastikan instrumen pengujian, khususnya Apache JMeter, dapat menggenerasi ratusan *request* secara serentak (*concurrent*) tanpa mengalami kelebihan beban (*bottleneck*) pada sisi klien. Selain itu, perangkat ini juga dikonfigurasi untuk mensimulasikan aktivitas pengguna normal melalui peramban (*browser*), serta menggunakan Burp Suite untuk memanipulasi dan memalsukan *payload* pada skenario serangan *Webhook Spoofing*.

Di sisi lain, *target server* diimplementasikan menggunakan VPS dengan sistem operasi Ubuntu Server 24.04.1 LTS. Konfigurasi server ini beroperasi dengan spesifikasi 1 *core* CPU, RAM sebesar 4 GB, dan kapasitas penyimpanan 50 GB. VPS ini bertindak sebagai infrastruktur terpusat yang menjalankan keseluruhan layanan sistem, meliputi *web server* (API Next.js), basis data MariaDB 10.11.10 untuk penyimpanan data transaksi persisten, serta layanan Redis 7.2 untuk menjalankan mekanisme *distributed lock* (*Idempotency Key*).

Pemilihan spesifikasi VPS dengan sumber daya komputasi menengah ini bertujuan untuk mereplikasi kondisi *server production* berskala standar. Konfigurasi ini sekaligus menjadi parameter yang ideal untuk mengukur efektivitas mitigasi. Dengan spesifikasi CPU dan RAM yang terbatas, implementasi keamanan menggunakan Redis harus mampu memblokir ratusan *request* duplikat (*Replay Attack*) dalam hitungan milidetik. Hal ini membuktikan bahwa mekanisme pertahanan yang diusulkan tidak hanya efektif mengamankan data, tetapi juga menjaga stabilitas kinerja *server* dari risiko kelumpuhan (*downtime*) akibat lonjakan beban pemrosesan.

4.4.2.2 Skenario Pengujian

a. Pengujian Kerentanan Manipulasi Panjang Signature (*Buffer Mismatch*)

Tujuan dari pengujian ini adalah untuk mengevaluasi keandalan mekanisme pra-validasi (*pre-validation*) dan sanitasi data pada lapisan keamanan *endpoint* server.



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Secara spesifik, pengujian ini menargetkan potensi kelemahan dalam menangani variabel memori (*buffer length*) yang tidak sesuai standar sebelum masuk ke fungsi komparasi kriptografi `crypto.timingSafeEqual` di lingkungan Node.js. Jika anomali panjang karakter ini tidak ditangani dengan ketat, penyerang dapat mengeksploitasi celah tersebut untuk mengelabui proses verifikasi. Kegagalan fungsi ini berpotensi berujung pada lolosnya otentikasi (*authentication bypass*) dan tereksekusinya manipulasi transaksi secara ilegal di dalam basis data tanpa memicu peringatan sistem.

Serangan simulasi ini akan dikirimkan menggunakan instrumen Burp Suite (melalui fitur *Repeater*). *Payload* transaksi pada *body* JSON dimanipulasi secara manual dengan mengubah nilai "status": "UNPAID" menjadi "status": "PAID". Selanjutnya, pada *header* X-Callback-Signature yang seharusnya berisi 64 karakter heksadesimal standar SHA-256, nilainya sengaja diganti dengan variasi *string* anomali (seperti "x", "123", *null*, atau karakter berlebih). Injeksi ini kemudian ditembakkan menuju *endpoint callback* pada dua kondisi sistem yang berbeda: sistem rentan (tanpa validasi panjang *buffer* awal) dan sistem *secure* (yang telah dilengkapi dengan fungsi pengecekan ukuran variabel sebelum komparasi komputasi dilakukan).

Parameter yang diamati:

1. Keandalan mekanisme penyaringan (*filtering*) server dalam mengidentifikasi dan memblokir input variabel memori yang tidak seimbang sebelum membebani siklus komputasi kriptografi utama.
2. Perbandingan kode status respons HTTP akhir antara sistem rentan yang berpotensi gagal melakukan validasi sehingga meloloskan muatan manipulatif (HTTP 200 *OK*), dengan sistem *secure* yang mampu memotong siklus permintaan secara dini (*early exit*) dan menolak secara elegan menggunakan HTTP 400 *Bad Request*.
3. Pencatatan jejak audit (*audit trail*) melalui observasi keluaran *log* pada terminal server untuk memastikan sistem berhasil mendeteksi dan mencatat anomali panjang karakter tersebut tanpa merusak logika bisnis utama.



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

b. Pengujian Simulasi Manipulasi Transmisi (Webhook Spoofing)

Tujuan dari pengujian ini adalah menyimulasikan vektor serangan pemalsuan data (*spoofing*) di mana aktor ancaman memanipulasi *payload* transaksi di tengah jalur transmisi guna mendapatkan akses layanan secara ilegal tanpa melakukan pelunasan. Mengingat penyadapan jaringan langsung (*sniffing*) pada koneksi HTTPS memerlukan eksploitasi tingkat lanjut, pengujian ini dilakukan menggunakan metode simulasi *Proof of Concept* (PoC). Penyerang memanfaatkan data *log* otentik (dengan status awal "UNPAID") beserta *header* X-Callback-Signature aslinya yang didapatkan langsung dari sistem.

Melalui instrumen Postman API, penyerang mengirimkan ulang muatan data otentik tersebut. Lalu lintas ini kemudian dicegat menggunakan proksi Burp Suite yang telah dikonfigurasi dengan aturan *Match and Replace*. Sistem proksi akan memodifikasi teks "status": "UNPAID" menjadi "status": "PAID" secara otomatis dalam hitungan milidetik sebelum diteruskan ke *endpoint callback* server target. Pengujian ini dilakukan secara komparatif untuk membandingkan respons sistem rentan (tanpa verifikasi kriptografi) dan sistem *secure* saat menerima muatan data manipulatif dengan *signature* yang tidak sinkron.

Parameter yang diamati:

1. Kemampuan fungsi verifikasi kriptografi HMAC pada server dalam mendeteksi anomali integritas data (membandingkan sistem rentan yang meloloskan transaksi dengan sistem *secure* yang secara tegas menolak melalui respons HTTP 400 Bad Request).
2. Integritas rekaman data transaksi pada basis data utama (tabel Payment) guna memastikan status tagihan tidak dapat diperbarui secara ilegal menjadi "SUCCESS".
3. Kemunculan jejak audit berupa pencetakan *log* peringatan indikasi pelanggaran keamanan (*Security Breach*) pada terminal server saat manipulasi data terdeteksi.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

c. Pengujian Kerentanan Race Condition, Replay Attack, dan Evaluasi Ketahanan Beban (*Stress Testing*)

Tujuan dari pengujian ini adalah menguji efektivitas mekanisme *Idempotency Key* berbasis Redis dan penguncian basis data absolut (*Database Absolute Lock*) dalam menangkal eksekusi transaksi ganda akibat manipulasi konkurensi dan eksploitasi jeda waktu. Selain itu, pengujian ini diekspansi untuk mengevaluasi tingkat ketahanan, stabilitas, dan efisiensi server ketika menghadapi lonjakan lalu lintas yang masif melalui pengujian beban bertahap (*Performance Testing*) guna mengamati titik kelelahan sumber daya (*resource exhaustion*) yang berpotensi memicu efek *Denial of Service* (DoS).

Pengujian ini memanfaatkan fitur simulasi pelunasan pada dashboard penyedia layanan (Tripay Sandbox) untuk memicu pengiriman webhook natural secara *real time* ke server. Payload JSON otentik dengan status "PAID" yang diterima oleh server kemudian disalin sebagai amunisi pengujian. Serangan eksploitasi dikirimkan menggunakan instrumen otomasi Apache JMeter yang dibagi menjadi tiga fase serangan komparatif antara sistem rentan (tanpa mitigasi) dan sistem *secure* (dengan verifikasi berlapis):

1. Fase Pertama: Menembakkan 100 permintaan duplikat secara serentak (*High Concurrency*) untuk memicu *Race Condition* dan *Replay Attack* pada sistem.
2. Fase Kedua: Menembakkan ulang permintaan yang sama setelah memberikan jeda waktu lebih dari 60 detik guna mengeksploitasi masa kedaluwarsa memori (*TTL Expiry*) pada Redis.
3. Fase Ketiga: Melakukan pengujian performa dan tegangan (*Load & Stress Testing*) secara bertahap. Beban lalu lintas *Replay Attack* ditingkatkan dalam tiga eskalasi parameter secara berurutan, yaitu 100 *threads*, 500 *threads*, hingga mencapai ekstrem 1.000 *threads* (masing-masing dengan *Loop Count* 10 dalam waktu 1 detik) untuk memicu kelelahan sumber daya (*resource exhaustion*).

Parameter yang diamati:

1. Kemampuan server dalam mendeteksi dan menolak permintaan duplikat



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

secara instan di tingkat memori RAM (Redis) yang ditandai dengan interupsi kode respons HTTP 409 Conflict dan HTTP 429 Too Many Requests.

2. Kemampuan lapisan pertahanan dalam (*Database Absolute Lock*) dalam menahan eksploitasi jeda waktu setelah kunci memori pada Redis terhapus secara otomatis berdasarkan batas waktu (*Time-To-Live / TTL*).
3. Jumlah total entitas hak akses (*Token Tes*) yang berhasil dicetak pada tabel TestAttempt di basis data utama untuk satu ID pembayaran yang sama.
4. Evaluasi komparatif terhadap metrik performa jaringan (Latensi dan *Throughput*) serta pemantauan lonjakan penggunaan sumber daya perangkat keras (CPU dan *Memory Usage*) saat simulasi *Stress Testing* berlangsung untuk membuktikan efisiensi arsitektur yang diusulkan.

4.4.2.3 Pengujian Kerentanan Manipulasi Panjang Signature (*Buffer Mismatch*)

Pada skenario ini, pengujian dilakukan menggunakan instrumen proksi Burp Suite (fitur *Intruder*) yang dikonfigurasi untuk memanipulasi permintaan HTTP POST menuju *endpoint* server target pada rute *api/payment/callback*. Percobaan serangan disimulasikan dalam lima variasi pengujian berdasarkan modifikasi nilai anomali yang diinjeksikan pada *header X-Callback-Signature*.

Seluruh variasi serangan menggunakan muatan data (*payload*) JSON dasar yang sama, di mana parameter status pembayaran telah dimodifikasi secara manual oleh penyerang menjadi "status": "PAID". Tabel berikut menjelaskan konfigurasi modifikasi *header* untuk setiap variasi serangan yang digunakan:

Tabel 4. 6 Konfigurasi Variasi Manipulasi Panjang Signature

No	Variasi Serangan	Nilai Header X-Callback-Signature	Tujuan Konfigurasi
1	Injeksi Karakter Sangat Pendek (<i>Short String</i>)	x	Menguji respons sistem ketika menerima nilai karakter yang sangat



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

No	Variasi Serangan	Nilai Header X-Callback-Signature	Tujuan Konfigurasi
			singkat.
2	Injeksi Karakter Berlebih (<i>Over-length String</i>)	e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855e3b0c44298fc1c149afb4c8996fb92427ae41e464 (100 karakter acak)	Menguji anomali <i>buffer overflow</i> dengan memaksakan ukuran memori di luar batas maksimal SHA-256.
3	Injeksi Format Angka Mentah (<i>Integer Mismatch</i>)	1234567890	Memicu kelemahan tipe data komputasi menggunakan format numerik tanpa <i>string</i> .
4	Injeksi Simbol Non-Heksadesimal	@#\$\$%^&*!()	Mengevaluasi penanganan karakter tidak valid pada fungsi decode HMAC.

a. Serangan Pertama (Injeksi Karakter Sangat Pendek / Short String)

Pada skenario pertama ini, serangan ditargetkan untuk menguji respons sistem ketika menerima nilai karakter *signature* yang sangat singkat. Penyerang mengonfigurasi Burp Suite *Repeater* dengan memasukkan *string* anomali (x) yang hanya berjumlah 1 karakter pada *header* X-Callback-Signature. Pada server rentan yang sama sekali tidak menerapkan mekanisme verifikasi kriptografi berlapis, manipulasi *header* ini tidak akan divalidasi, sehingga muatan data manipulatif berpotensi diloloskan begitu saja (HTTP 200 OK).

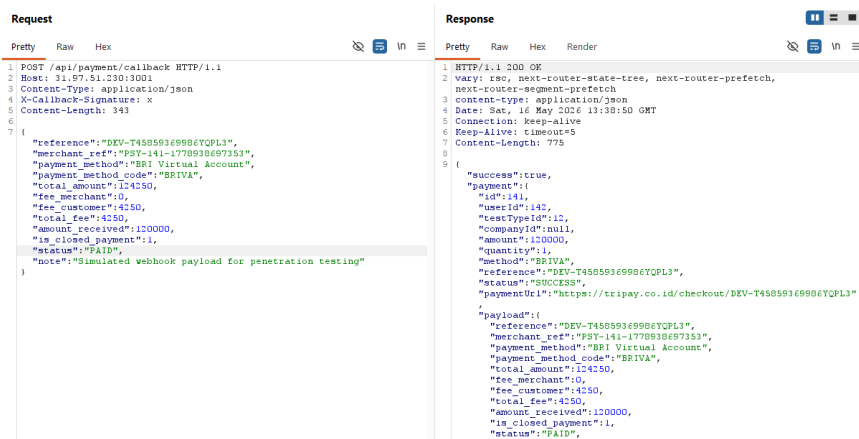
Di sisi lain, apabila server menggunakan fungsi komparasi bawaan tanpa melakukan pra-validasi panjang karakter (*length check*), hal ini dapat memicu galat komputasi yang berujung pada interupsi sistem (HTTP 500 Internal Server Error). Pengujian ini memastikan bahwa sistem *secure* yang diusulkan mampu mencegat anomali tersebut sejak awal dan secara elegan mengembalikan respons penolakan



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

HTTP 400 Bad Request tanpa mengganggu stabilitas *runtime* aplikasi.



Gambar 4. 35 Tampilan Respons HTTP 200 OK pada Sistem Rentan

Berdasarkan Gambar 4.35 di atas, eksekusi pertama dilakukan dengan mengarahkan *payload* injeksi menuju Sistem Rentan. Tangkapan layar dari instrumen Burp Suite menunjukkan bahwa permintaan HTTP POST dengan *header* X-Callback-Signature berisi satu karakter statis (x) berhasil dieksekusi oleh server. Karena ketiadaan mekanisme verifikasi kriptografi yang memadai, server langsung mempercayai muatan data secara sepihak dan merespons dengan kode status HTTP 200 OK. Hal ini mengonfirmasi bahwa server rentan tidak melakukan validasi apa pun terhadap integritas sumber pengirim.



Gambar 4. 36 Perubahan Status Transaksi Menjadi Success secara Ilegal

Dampak dari keberhasilan injeksi tersebut dibuktikan secara nyata pada tingkat penyimpanan basis data sebagaimana ditunjukkan pada Gambar 4.36. Oleh karena server secara mentah menerima nilai "status": "PAID" dari muatan JSON yang dikirimkan melalui Burp Suite, status transaksi pada tabel Payment secara otomatis diperbarui menjadi *success*. Pembaruan status ini bersifat ilegal karena tidak selaras dengan kondisi riil pada *dashboard payment gateway* Tripay, di mana transaksi tersebut sebenarnya masih berstatus belum dibayar (*unpaid*).



Hak Cipta :

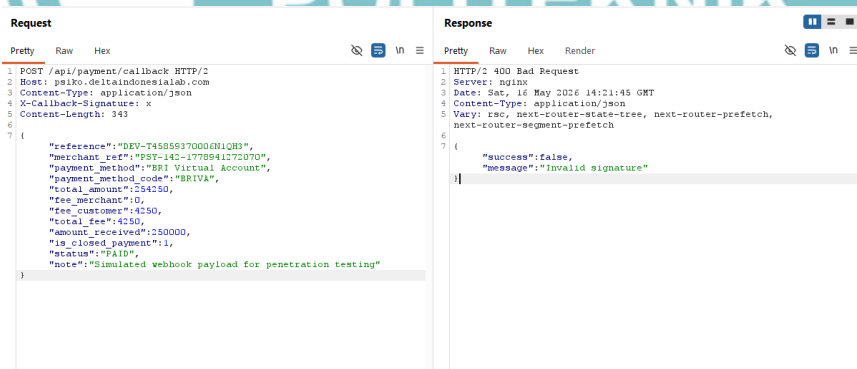
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 141.
prisma:query SELECT `psikotesting`.`Payment`.`id`, `psikotesting`.`Payment`.`user`
Id`, `psikotesting`.`Payment`.`testTypeId`, `psikotesting`.`Payment`.`companyId`,
`psikotesting`.`Payment`.`amount`, `psikotesting`.`Payment`.`quantity`, `psikotes
testing`.`Payment`.`method`, `psikotesting`.`Payment`.`reference`, `psikotesting
`.`Payment`.`status`, `psikotesting`.`Payment`.`paymentUrl`, `psikotesting`.`Paym
ent`.`payload`, `psikotesting`.`Payment`.`paidAt`, `psikotesting`.`Payment`.`expi
redAt`, `psikotesting`.`Payment`.`promoId`, `psikotesting`.`Payment`.`companyPric
ingId`, `psikotesting`.`Payment`.`createdAt`, `psikotesting`.`Payment`.`updatedAt`
` FROM `psikotesting`.`Payment` WHERE `psikotesting`.`Payment`.`reference` = ? LI
MIT ? OFFSET ?
```

Gambar 4. 37 Log Terminal Server Menunjukkan Pencetakan Token Akses Ilegal

Kerentanan ini lebih lanjut memicu eskalasi kerugian fungsional pada aplikasi. Berdasarkan catatan *log* server pada Gambar 4.37, perubahan status pembayaran menjadi *success* memicu sistem untuk mengeksekusi logika bisnis selanjutnya, yaitu mencetak dan menerbitkan akses *Token Tes* untuk nomor transaksi 141. Temuan ini merupakan bukti celah keamanan kritis (*High Impact Vulnerability*), di mana penyerang dapat memperoleh layanan digital secara penuh tanpa perlu melakukan transaksi keuangan yang sah.

Sebaliknya, pengujian dengan skenario serangan yang sama persis kemudian diarahkan menuju Sistem *Secure* guna memvalidasi kemampuan mitigasi dari arsitektur yang diusulkan.



Gambar 4. 38 Tampilan Respons HTTP 400 Bad Request pada Sistem Secure

Berdasarkan Gambar 4.38 di atas, tangkapan layar dari instrumen Burp Suite menunjukkan bahwa arsitektur keamanan yang telah diperkuat mampu mendeteksi anomali *signature* yang sangat pendek tersebut sejak tahap awal. Server secara otomatis menolak permintaan ilegal dan secara elegan mengembalikan respons HTTP 400 Bad Request disertai pesan galat "Invalid signature". Hal ini



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

mengonfirmasi bahwa server *secure* berhasil memotong siklus eksekusi di lapisan pra-validasi sebelum muatan data manipulatif tersebut menyentuh logika basis data.

```
[Webhook HTTP POST] Menerima payload untuk: PSY-142-1778941272070
[Signature Header] x
x
[BODY JSON RAW]
{"reference":"DEV-T45859370006N1QH3","merchant_ref":"PSY-142-1778941272070","payment_method":"BRI Virtual Account","payment_method_code":"BRIVA","total_amount":254250,"fee_merchant":0,"fee_customer":4250,"total_fee":4250,"amount_received":250000,"is_closed_payment":1,"status":"PATO","note":"Simulated webhook payload for penetration testing"}
[X] [SECURITY BREACH] Invalid HMAC Signature terdeteksi! Akses ditolak.
```

Gambar 4. 39 Log Terminal Sistem Secure Mendeteksi dan Memblokir Serangan

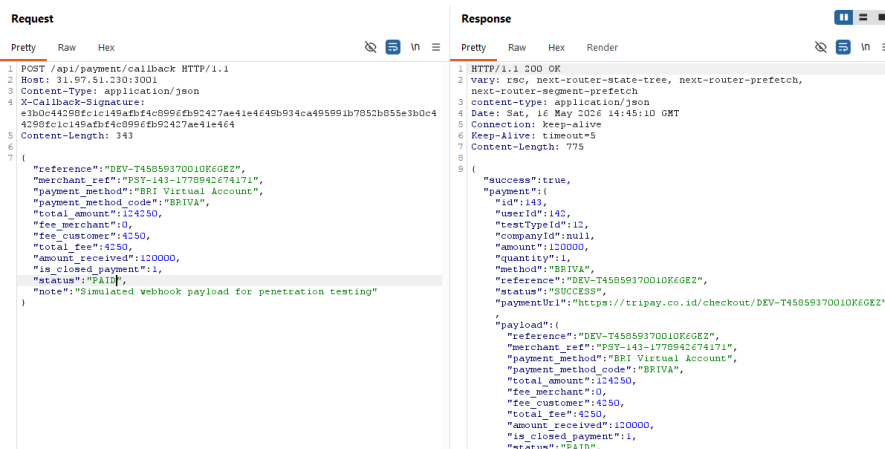
Tindakan mitigasi proaktif tersebut juga terekam dengan sangat terperinci pada sistem pemantauan server, seperti yang ditunjukkan pada Gambar 4.39. Berdasarkan *log* terminal, server mencatat secara transparan bahwa nilai *header* yang masuk hanyalah karakter tunggal (x). Sistem kemudian secara akurat memproses validasi HMAC dan langsung memunculkan peringatan interupsi [SECURITY BREACH] Invalid HMAC Signature terdeteksi! Akses ditolak.. Keberadaan *log* ini membuktikan bahwa sistem tidak hanya mampu menolak injeksi manipulasi dengan stabil, tetapi juga memiliki tingkat visibilitas dan presisi yang tinggi dalam mengidentifikasi serta mencatat setiap upaya eksploitasi logika bisnis.

b. Serangan Kedua (Injeksi Karakter Berlebih / Over-length String)

Pada skenario kedua, penyerang mencoba memicu anomali kelebihan muatan memori (*buffer overflow*). Penyerang mengonfigurasi Burp Suite *Repeater* dengan memasukkan nilai karakter heksadesimal acak yang panjangnya melampaui batas maksimal 64 karakter standar algoritma SHA-256, yaitu sebanyak 100 karakter (e3b0c44...). Pada *server* rentan, variabel yang terlalu panjang ini berpotensi diloloskan begitu saja (HTTP 200 OK) jika tidak ada validasi. Namun, apabila algoritma dipaksa melakukan alokasi memori (*memory allocation*) untuk komparasi ukuran yang tidak wajar, hal ini dapat memicu galat sistem (*HTTP 500 Internal Server Error*). Pengujian ini memastikan sistem *secure* memotong siklus permintaan (*request life-cycle*) secara dini (*early exit*) dan mengembalikan respons penolakan HTTP 400 *Bad Request* jika ukuran variabel melebihi ambang batas.

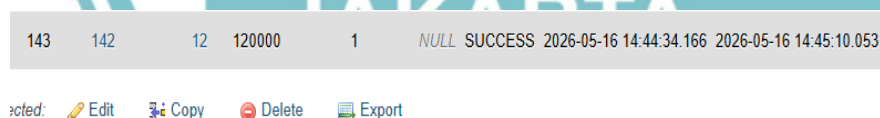


Untuk membuktikan kerentanan tersebut secara empiris, eksekusi dilakukan dengan mengarahkan *payload* injeksi *header* berlebih menuju Sistem Rentan



Gambar 4. 40 Tampilan Respons HTTP 200 OK pada Sistem Rentan (Karakter Berlebih)

Berdasarkan Gambar 4.40 di atas, tangkapan layar dari instrumen Burp Suite menunjukkan bahwa permintaan HTTP POST yang membawa *header* X-Callback-Signature dengan muatan anomali 100 karakter berhasil dieksekusi tanpa kendala oleh *server*. Ketidadaan mekanisme pengecekan batas maksimal ukuran variabel menyebabkan *server* rentan gagal mengidentifikasi anomali *over-length* tersebut, sehingga secara mentah mempercayai muatan data dan merespons dengan kode status *HTTP 200 OK*.



Gambar 4. 41 Perubahan Status Transaksi Menjadi Success secara Ilegal

Dampak dari lolosnya anomali karakter berlebih tersebut kembali terlihat secara langsung pada integritas data. Berdasarkan Gambar 4.41 di atas, status transaksi pada tabel Payment secara paksa diperbarui menjadi sukses. Hal ini menegaskan bahwa tanpa adanya pra-validasi panjang *signature*, setiap modifikasi yang diselipkan pada *body* JSON akan selalu dipercaya dan dieksekusi oleh basis data.

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

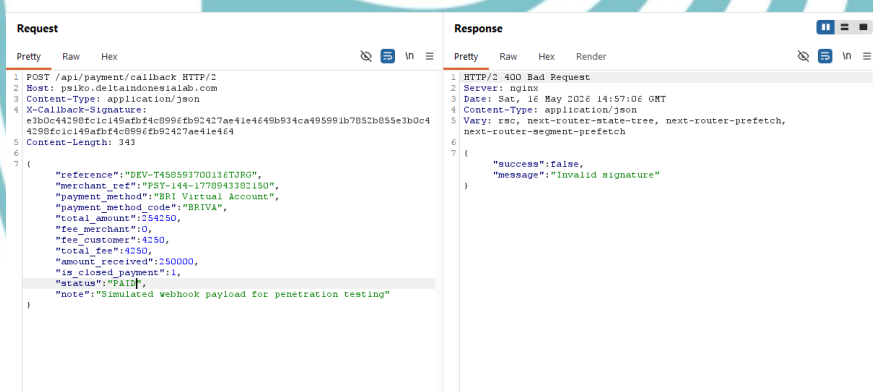
estesting`.TestAttempt`.reservedAt`, `psikotesting`.TestAttempt`.status`, `psiko
testing`.TestAttempt`.tokenId` FROM `psikotesting`.TestAttempt` WHERE `psikote
esting`.TestAttempt`.id` = ? LIMIT ? OFFSET ?
prisma:query COMMIT
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 143.

```

Gambar 4. 42 Log Terminal Server Menunjukkan Pencetakan Token Akses Ilegal

Kerentanan ini juga kembali memicu kerugian fungsional pada aplikasi. Berdasarkan catatan log pada Gambar 4.42, sistem secara otomatis meneruskan logika bisnis dengan mencetak akses token layanan bagi pengguna. Kegagalan *server* dalam menyaring muatan variabel yang tidak wajar membuktikan tingginya tingkat paparan kerentanan aplikasi terhadap manipulasi transaksi dari luar sistem.

Sebaliknya, pengujian dengan skenario injeksi karakter berlebih yang sama persis kemudian diarahkan menuju Sistem *Secure* guna memvalidasi kemampuan arsitektur pertahanan yang diusulkan.



Gambar 4. 43 Penolakan Serangan Karakter Berlebih pada Sistem Secure

Berdasarkan Gambar 4.43 di atas, tangkapan layar Burp Suite menunjukkan bahwa *server secure* mampu membedakan ukuran variabel secara presisi. Sebelum algoritma kriptografi dipaksa memproses alokasi memori komparasi yang berlebihan, sistem telah memotong *request* tersebut di lapisan awal karena panjang karakter melebihi ambang batas mutlak 64 byte. Penolakan ini dibuktikan dengan respons *HTTP 400 Bad Request* beserta pesan "Invalid signature", yang secara efektif memblokir serangan.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
[Webhook HTTP POST] Menerima payload untuk: PSY-144-1778943382150
[Signature Header] e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855e3b0c44298fc1c149afb4c8996fb92427ae41e464
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855e3b0c44298fc1c149afb4c8996fb92427ae41e464
[BODY JSON RAW]
{"reference":"DEV-T458593700136TJRG","merchant_ref":"PSY-144-1778943382150","payment_method":"BRI Virtual Account","payment_method_code":"BRIVA","total_amount":254250,"fee_merchant":0,"fee_customer":4250,"total_fee":4250,"amount_received":250000,"is_closed_payment":1,"status":"PAID","note":"Simulated webhook payload for penetration testing"}
X [SECURITY BREACH] Invalid HMAC Signature terdeteksi! Akses ditolak.
```

Gambar 4. 44 Log Terminal Sistem Secure Mendeteksi Anomali Memori

Tindakan mitigasi proaktif tersebut juga terekam pada sistem pemantauan server seperti yang terlihat pada Gambar 4.44. Log terminal menunjukkan peringatan *[SECURITY BREACH] Invalid HMAC Signature terdeteksi!*. Catatan log ini memvalidasi bahwa sistem *secure* dirancang untuk memiliki daya tahan (*resilience*) yang baik, di mana sistem tidak mengalami kelumpuhan (*crash / Error 500*) saat memproses beban parameter memori yang di luar standar, melainkan secara presisi mendeteksi dan memblokirnya sebagai sebuah ancaman siber.

c. Serangan Ketiga (Injeksi Format Angka Mentah / Integer Mismatch)

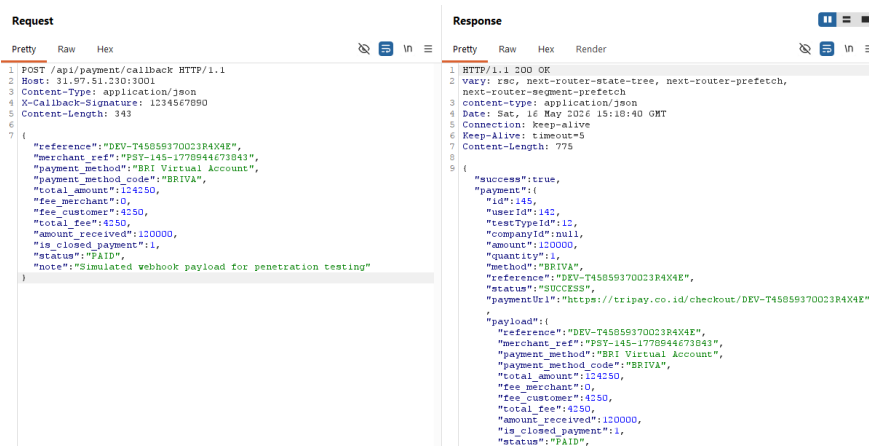
Pada skenario ketiga, pengujian difokuskan pada kelemahan tipe data komputasi (*Data Type Mismatch*). Penyerang mengonfigurasi Burp Suite *Repeater* dengan memasukkan nilai numerik mentah (1234567890) tanpa membungkusnya dalam format *string*. Mengingat fungsi kriptografi Node.js hanya menerima tipe data tertentu (*String, Buffer, ArrayBuffer, dll.*), *server* rentan dapat tertipu dan meloloskannya (HTTP 200 OK), atau mengalami kegagalan konversi tipe data memori otomatis yang menyebabkan galat sistem (*HTTP 500 Internal Server Error*). Pengujian ini memastikan *server secure* memiliki penanganan pengecualian (*Exception Handling*) yang ketat untuk menolak format yang tidak sesuai dengan HTTP 400 *Bad Request*.

Untuk membuktikan kerentanan tersebut secara empiris, eksekusi dilakukan dengan mengarahkan *payload* injeksi format numerik mentah menuju Sistem Rentan.



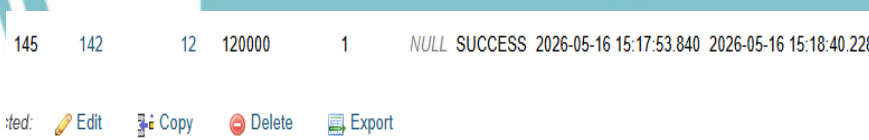
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



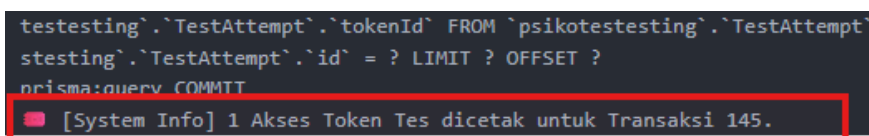
Gambar 4. 45 Tampilan Respons HTTP 200 OK pada Sistem Rentan (Injeksi Angka Mentah)

Berdasarkan Gambar 4.45 di atas, tangkapan layar dari instrumen Burp Suite menunjukkan bahwa permintaan HTTP POST dengan *header* X-Callback-Signature yang berisi tipe data angka (1234567890) berhasil dieksekusi oleh *server*. Ketiadaan validasi tipe data (*type checking*) menyebabkan *server* rentan secara otomatis mengabaikan anomali format tersebut, menerima muatan manipulasi secara utuh, dan meresponsnya dengan kode status *HTTP 200 OK*.



Gambar 4. 46 Perubahan Status Transaksi Menjadi Success secara Ilegal

Dampak dari kegagalan validasi tipe data tersebut berimplikasi langsung pada basis data. Berdasarkan Gambar 4.46, status transaksi pada tabel Payment berhasil diperbarui menjadi sukses secara sepihak. Hal ini mengonfirmasi bahwa *server* sama sekali tidak memverifikasi apakah *signature* yang dikirim merupakan format kriptografi yang sah sebelum melakukan perubahan entitas data.



Gambar 4. 47 Log Terminal Server Menunjukkan Pencetakan Token Akses Ilegal

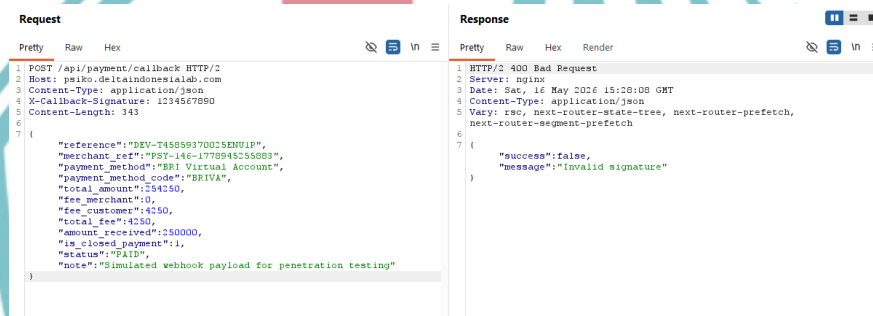


Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

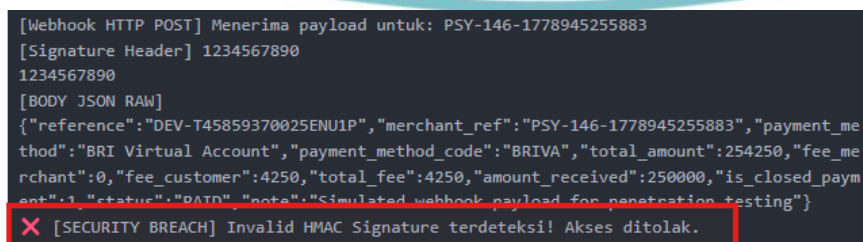
Selanjutnya, kelalaian dalam menangani anomali tipe data ini berdampak pada kebocoran fungsionalitas sistem. Berdasarkan log terminal pada Gambar 4.47, perubahan status menjadi *success* kembali memicu sistem untuk mencetak token akses layanan bagi penyerang. Kegagalan ini menunjukkan bahwa *server* rentan tidak memiliki mekanisme pertahanan *fail-safe* ketika dihadapkan pada masukan data yang menyimpang dari standar API.

Sebaliknya, pengujian injeksi angka mentah yang sama kemudian diarahkan menuju Sistem *Secure* guna mengevaluasi ketahanan penanganan pengecualian (*Exception Handling*) pada arsitektur pertahanan yang diusulkan.



Gambar 4. 48 Penolakan Serangan Injeksi Angka Mentah pada Sistem Secure

Berdasarkan Gambar 4.48 di atas, tangkapan layar Burp Suite membuktikan bahwa arsitektur *secure* memiliki validasi yang ketat. Sistem mendeteksi adanya ketidaksesuaian nilai dan format *signature* sejak awal, sehingga *request* tersebut langsung ditolak dengan mengembalikan status *HTTP 400 Bad Request*. Hal ini memastikan sistem terhindar dari potensi kelumpuhan (*Unhandled Promise Rejection*) yang biasanya terjadi saat Node.js dipaksa mengonversi tipe data secara paksa.



Gambar 4. 49 Log Terminal Sistem Secure Mendeteksi Anomali Format Data



Hak Cipta :

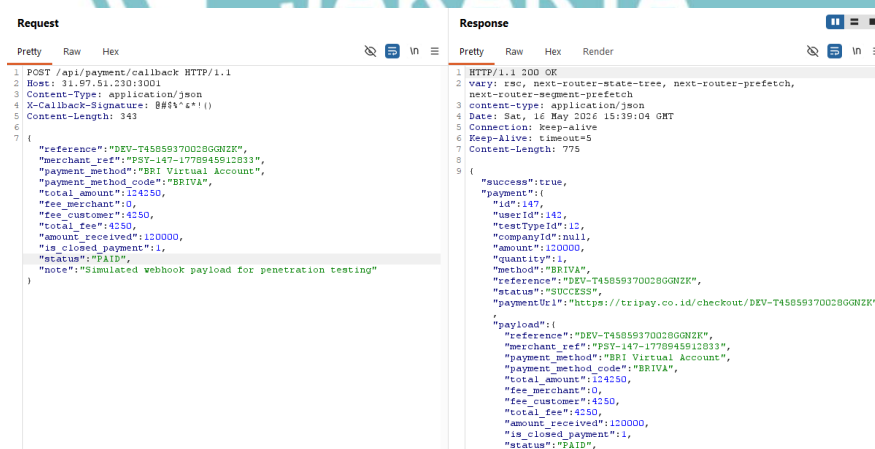
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Tindakan penolakan tersebut juga didokumentasikan dengan baik oleh sistem pemantauan server seperti yang ditunjukkan pada Gambar 4.49. Log terminal kembali mencatat peringatan *[SECURITY BREACH] Invalid HMAC Signature terdeteksi!*. Catatan ini menegaskan bahwa mekanisme keamanan yang diimplementasikan beroperasi secara konsisten dalam menolak dan mencatat segala bentuk anomali, baik berupa manipulasi panjang memori maupun ketidaksesuaian tipe data.

d. Serangan Keempat (Injeksi Simbol Non-Heksadesimal)

Algoritma HMAC SHA-256 pada umumnya menghasilkan keluaran berbasis karakter heksadesimal. Pada pengujian ini, penyerang secara sengaja menginjeksikan deretan simbol khusus (@#\$%^&*!()) sebagai muatan (*payload*) pada instrumen Burp Suite *Repeater*. Pada *server* rentan, ketiadaan sanitasi data dapat membuat muatan ilegal ini diloloskan (HTTP 200 OK), atau menyebabkan kegagalan dekode saat dipetakan ke dalam bentuk *byte* memori yang memicu galat sistem (*HTTP 500 Internal Server Error*). Pengujian ini membuktikan bahwa sistem *secure* memiliki lapis penyaring (*filter*) yang mengembalikan HTTP 400 *Bad Request* guna menolak format karakter ilegal sebelum masuk ke tahap komputasi inti.

Untuk membuktikan kerentanan tersebut secara empiris, eksekusi dilakukan dengan mengarahkan *payload* injeksi simbol non-heksadesimal menuju Sistem Rentan.



Gambar 4. 50 Tampilan Respons HTTP 200 OK pada Sistem Rentan (Injeksi Simbol)



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

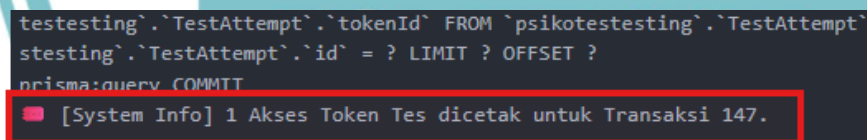
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Berdasarkan Gambar 4.50 di atas, tangkapan layar dari instrumen Burp Suite menunjukkan bahwa permintaan HTTP POST yang membawa *header* X-Callback-Signature berisi deretan simbol ilegal (@#\$%^&*!()) berhasil dieksekusi oleh *server*. Ketiadaan sanitasi data (*data sanitization*) menyebabkan *server* rentan gagal memfilter karakter yang tidak sesuai dengan format standar kriptografi, sehingga *server* tetap mempercayai muatan data dan meresponsnya dengan kode status *HTTP 200 OK*.



Gambar 4. 51 Perubahan Status Transaksi Menjadi Success secara Ilegal

Dampak fatal dari ketiadaan sanitasi data tersebut kembali berimbas pada integritas basis data. Berdasarkan Gambar 4.51, status transaksi pada tabel Payment secara keliru diperbarui menjadi sukses. Fakta ini membuktikan bahwa *server* rentan mengabaikan validasi bentuk karakter dari *signature*, sehingga modifikasi apa pun pada *body* JSON akan tetap diproses.



Gambar 4. 52 Log Terminal Server Menunjukkan Pencetakan Token Akses Ilegal

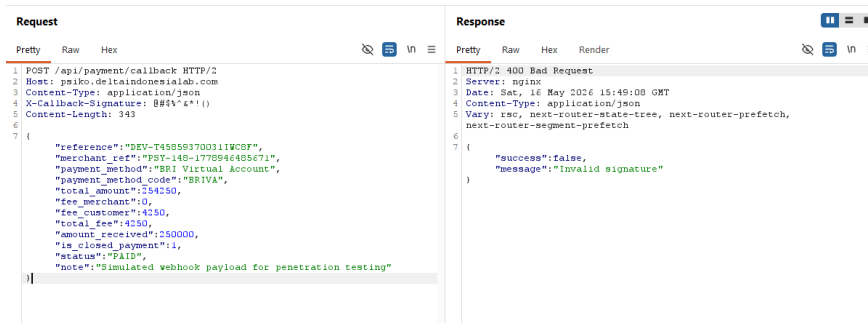
Kerentanan ini juga berujung pada eksploitasi fitur utama aplikasi. Seperti yang terlihat pada log terminal di Gambar 4.52, perubahan status menjadi sukses memicu sistem untuk mengeksekusi logika penerbitan akses token layanan bagi pengguna penyerang. Hal ini menunjukkan bahwa sistem sama sekali tidak memiliki lapisan penyaring untuk memblokir muatan yang mengandung karakter anomali.

Sebaliknya, pengujian dengan skenario injeksi simbol yang sama kemudian diarahkan menuju Sistem *Secure* guna memvalidasi kemampuan sanitasi data pada arsitektur pertahanan yang diusulkan.



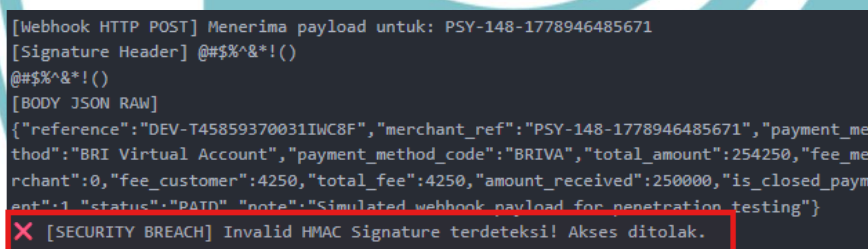
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4. 53 Penolakan Serangan Injeksi Simbol pada Sistem Secure

Berdasarkan Gambar 4.53 di atas, tangkapan layar Burp Suite membuktikan bahwa arsitektur keamanan yang telah diperkuat memiliki lapis penyaring (*filter*) yang ketat. Sistem secara otomatis mendeteksi keberadaan karakter non-heksadesimal yang tidak valid dan langsung menolak permintaan tersebut dengan status *HTTP 400 Bad Request*. Mekanisme ini mencegah *server* dari potensi kegagalan sistem (*Error 500*) yang dapat terjadi akibat proses decode karakter ilegal di dalam memori kriptografi.



Gambar 4. 54 Log Terminal Sistem Secure Mendeteksi Anomali Karakter Ilegal

Tindakan pemblokiran proaktif tersebut juga terkonfirmasi pada sistem pemantauan server seperti yang ditunjukkan pada Gambar 4.54. Log terminal mencatat secara akurat munculnya peringatan *[SECURITY BREACH] Invalid HMAC Signature terdeteksi!*. Catatan ini menegaskan efektivitas sistem keamanan *secure* dalam melakukan proses pra-validasi dan sanitasi data secara konsisten untuk melindungi integritas ekosistem aplikasi.

4.4.2.4 Pengujian Simulasi Manipulasi Transmisi (Webhook Spoofing)

Pada skenario ini, dilakukan pengujian untuk mengevaluasi perilaku arsitektur API *callback* saat menghadapi serangan *Webhook Spoofing*. Serangan ini



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

menyimulasikan kondisi di mana penyerang berhasil memanipulasi muatan data (*payload*) pembayaran dari status gagal atau belum dibayar menjadi berhasil sebelum paket tersebut diproses oleh *server* target. Pengujian difokuskan pada dua aspek utama: pertama, apakah sistem rentan dapat dikelabui oleh perubahan muatan data JSON secara sepihak; dan kedua, apakah verifikasi kriptografi HMAC-SHA256 pada sistem *secure* mampu mendeteksi kerusakan integritas data dan memblokir modifikasi ilegal tersebut.

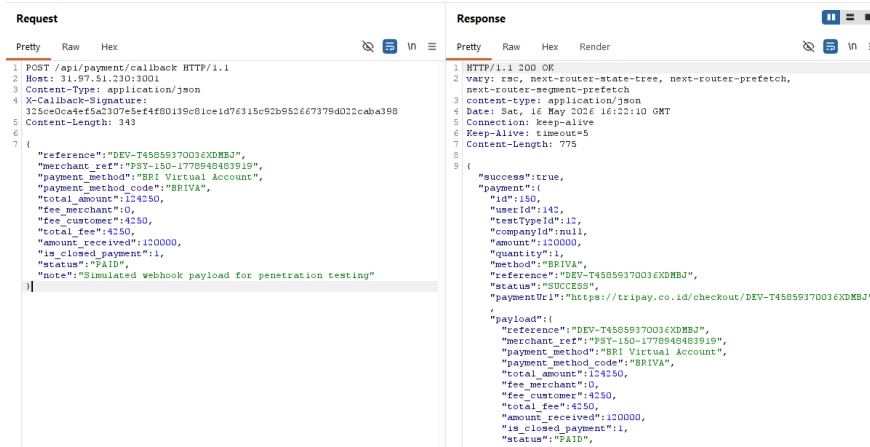
Pengujian dieksekusi menggunakan instrumen proksi Burp Suite (fitur *Repeater*). Berbeda dengan skenario kerentanan sebelumnya yang berfokus pada anomali bentuk karakter, pengujian ini menggunakan nilai *header X-Callback-Signature* asli yang secara sah dihasilkan oleh sistem *payment gateway* khusus untuk status transaksi belum dibayar (*UNPAID*). Penyerang menyalin muatan *webhook* asli tersebut beserta *header signature* bawaannya, lalu memodifikasi parameter status di dalam *body* JSON menjadi "PAID" secara manual (*on-the-fly*) sebelum mengirimkannya ke *server*. Hal ini bertujuan untuk mengevaluasi apakah *server* rentan akan menerima modifikasi tersebut secara buta, serta menguji apakah arsitektur *secure* secara komprehensif melakukan kalkulasi ulang guna mendeteksi ketidakcocokan (*mismatch*) antara *signature* status *UNPAID* dengan muatan *body* yang telah dimanipulasi.

Untuk membuktikan kerentanan tersebut secara empiris, eksekusi pertama dilakukan dengan mengarahkan muatan data yang telah dimanipulasi menuju Sistem Rentan.



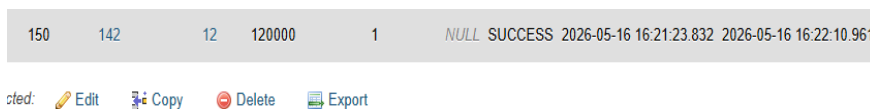
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4. 55 Tampilan Respons HTTP 200 OK pada Sistem Rentan (Webhook Spoofing)

Berdasarkan Gambar 4.55 di atas, tangkapan layar dari instrumen Burp Suite menunjukkan bahwa permintaan manipulasi *webhook* berhasil dieksekusi secara penuh tanpa adanya penolakan dari *server*. Karena *server* rentan tidak melakukan pencocokan (*hashing*) ulang antara muatan *body* JSON dengan *header* kriptografi asli yang diselundupkan, sistem secara buta mempercayai parameter "status": "PAID" yang diinjeksikan oleh penyerang. Kegagalan validasi ini dibuktikan dengan kembalian status *HTTP 200 OK* beserta muatan respons (*response body*) yang menampilkan nilai "success": true. Lebih lanjut, rincian respons tersebut memperlihatkan bahwa objek transaksi dengan nomor referensi terkait telah diperbarui oleh *server*, di mana parameter status diubah menjadi "SUCCESS" dan sistem secara otomatis mencetak cap waktu (*timestamp*) penyelesaian pada parameter *paidAt* (2026-05-16T16:22:10.960Z). Fakta ini secara mutlak mengonfirmasi bahwa *server* memproses muatan *spoofing* tersebut sebagai sebuah transaksi pembayaran yang sah, sehingga penyerang berhasil memanipulasi alur logika bisnis aplikasi.



Gambar 4. 56 Perubahan Status Transaksi Menjadi Success secara Ilegal



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Dampak fatal dari keberhasilan *spoofing* ini terekam secara jelas pada tingkat basis data sebagaimana ditunjukkan pada Gambar 4.56. Tangkapan layar dari antarmuka basis data memperlihatkan bahwa kolom status pada tabel Payment telah diperbarui secara sepihak menjadi *SUCCESS*. Perubahan ini menegaskan bahwa *server* rentan gagal melakukan validasi silang antara integritas *header signature* dengan isi muatan data. Meskipun penyerang menggunakan *header signature* asli yang secara spesifik dihasilkan oleh sistem Tripay untuk transaksi berstatus *UNPAID*, *server* secara buta tetap mengeksekusi instruksi pembaruan berdasarkan *body* JSON yang telah dimanipulasi menjadi *PAID*. Akibatnya, tercipta anomali dan inkonsistensi data yang sangat kritis; basis data sistem mencatat bahwa tagihan telah dilunasi, padahal secara faktual pada *dashboard payment gateway*, status riil transaksi tersebut masih belum dibayar. Hal ini membuktikan bahwa absennya verifikasi kriptografi yang presisi pada lapisan API secara langsung mengancam integritas penyimpanan data utama aplikasi.

```
testesting`.`TestAttempt`.`tokenId` FROM `psikotestesting`.`TestAttempt`
stesting`.`TestAttempt`.`id` = ? LIMIT ? OFFSET ?
prisma:query COMMIT
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 150.
```

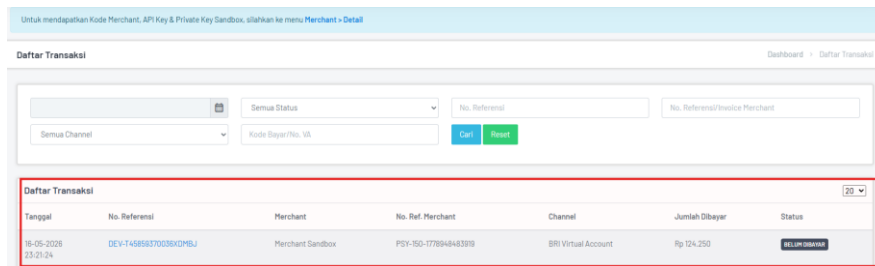
Gambar 4. 57 Log Terminal Server Menunjukkan Pencetakan Token Akses Ilegal

Sebagai eskalasi dari inkonsistensi basis data tersebut, kerentanan manipulasi transmisi ini memicu kerugian fungsional yang sangat fatal bagi penyedia layanan. Berdasarkan catatan log terminal *server* pada Gambar 4.57, perubahan status pembayaran yang dipaksakan menjadi sukses telah menipu aplikasi untuk secara otomatis mengeksekusi logika bisnis pasca-transaksi. Hal ini dibuktikan dengan terekamnya aktivitas sistem yang tertulis [System Info] 1 Akses Token Tes dicetak untuk Transaksi 150. Tereksekusinya perintah pencetakan token ini mengonfirmasi bahwa penyerang berhasil memperoleh komoditas digital berupa hak akses layanan tes secara penuh dan gratis tanpa melakukan transaksi finansial yang sah. Temuan ini secara tegas membuktikan bahwa tanpa adanya mekanisme komparasi integritas data yang presisi, seluruh alur bisnis dan kontrol akses aplikasi dapat dengan mudah dieksploitasi oleh peretas.



Hak Cipta :

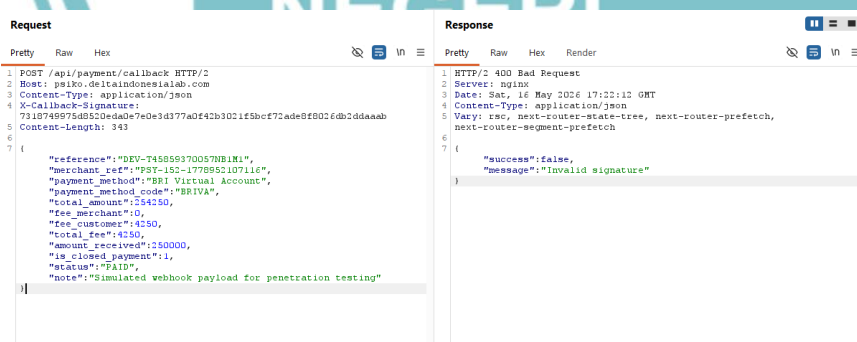
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4. 58 Status Transaksi pada Antarmuka Payment Gateway

Untuk memperkuat bukti terjadinya kerugian finansial akibat eksploitasi ini, peninjauan silang dilakukan pada *dashboard* resmi *payment gateway* Tripay. Berdasarkan Gambar 4.58 di atas, sistem Tripay secara valid mencatat bahwa nomor referensi transaksi terkait (DEV-T45859370036XDMBJ) masih berstatus "BELUM DIBAYAR". Kontradiksi mutlak antara status riil di *dashboard* resmi dengan status *SUCCESS* di dalam basis data lokal serta tercetaknya token akses menjadi bukti konklusif bahwa penyerang telah berhasil mem-*bypass* sistem pembayaran. Aplikasi PsikoTest telah memberikan hak akses layanan secara penuh tanpa adanya aliran dana riil yang masuk ke pihak penyedia layanan.

Percobaan kedua dilakukan dengan mengirimkan serangan *spoofing* dengan muatan yang sama persis menuju Sistem *Secure* guna memvalidasi kemampuan mitigasi dari arsitektur pertahanan yang diusulkan.



Gambar 4. 59 Penolakan Serangan Webhook Spoofing pada Sistem Secure

Berdasarkan Gambar 4.59 di atas, tangkapan layar Burp Suite membuktikan bahwa arsitektur keamanan yang telah diperkuat berhasil mendeteksi kerusakan integritas data secara presisi. Walaupun penyerang menyertakan *header signature* yang valid dan sah dari Tripay untuk status *UNPAID*, modifikasi secara sepihak pada *body*



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

JSON menjadi "status": "PAID" menyebabkan hasil kalkulasi *hash* di sisi *server* menjadi berbeda (*mismatch*) dari *signature* aslinya. Sistem secara otomatis dan tegas menolak permintaan tersebut dengan mengembalikan status *HTTP 400 Bad Request*. Tindakan pemblokiran ini juga diperkuat dengan kembalian respons berformat JSON {"success":false,"message":"Invalid signature"}, yang mengonfirmasi bahwa seluruh siklus logika bisnis dan interaksi ke basis data telah dihentikan secara absolut pada lapisan pra-validasi.

```
[Webhook HTTP POST] Menerima payload untuk: PSY-152-1778952107116
[Signature Header] 7318749975d8520eda0e7e0e3d377a0f42b3021f5bcf72ade8f8026db2ddaaab
7318749975d8520eda0e7e0e3d377a0f42b3021f5bcf72ade8f8026db2ddaaab
[BODY JSON RAW]
{"reference":"DEV-T45859370057NB1M1","merchant_ref":"PSY-152-1778952107116","payment_method":"BRI Virtual Account","payment_method_code":"BRIVA","total_amount":254250,"fee_merchant":0,"fee_customer":4250,"total_fee":4250,"amount_received":250000,"is_closed_payment":1,"status":"PAID","note":"Simulated webhook payload for penetration testing"}
❌ [SECURITY BREACH] Invalid HMAC Signature terdeteksi! Akses ditolak.
```

Gambar 4. 60 Log Terminal Sistem Secure Mendeteksi Modifikasi Muatan Data

Tindakan mitigasi proaktif tersebut didokumentasikan dengan sangat komprehensif pada sistem pemantauan *server* sebagaimana terlihat pada Gambar 4.60. Log terminal memperlihatkan transparansi alur verifikasi, di mana *server* merekam masuknya *header signature* bawaan (berawalan 73187499...) beserta muatan *body* JSON transaksi terkait (Nomor Referensi: DEV-T45859370057NB1M1) yang berstatus "PAID". Melalui proses verifikasi HMAC-SHA256, *server* mendeteksi adanya manipulasi dan langsung memutus akses dengan memunculkan peringatan interupsi [SECURITY BREACH] Invalid HMAC Signature terdeteksi! Akses ditolak.. Catatan log ini memberikan bukti empiris bahwa algoritma pengamanan bekerja dengan presisi tinggi; *server* tidak hanya sekadar menggagalkan serangan, tetapi juga mampu mengidentifikasi muatan ilegal, mencatat parameter anomali, dan secara mutlak memblokir eksploitasi *Webhook Spoofing* guna menjamin keutuhan ekosistem aplikasi.

4.4.2.5 Pengujian Kerentanan Race Condition, Replay Attack, dan Evaluasi Ketahanan Beban (Stress Testing)

Pada skenario ini, dilakukan pengujian untuk menguji efektivitas mekanisme *Idempotency Key* berbasis Redis dan penguncian basis data absolut (*Database*



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Absolute Lock) dalam menangkal eksekusi transaksi ganda akibat manipulasi konkurensi dan eksploitasi jeda waktu. Pengujian ini memanfaatkan fitur simulasi pelunasan pada dashboard penyedia layanan (Tripay Sandbox) untuk memicu pengiriman webhook natural secara *real-time* ke server. Payload JSON dengan status "PAID" beserta *signature* otentik yang diterima oleh server kemudian disalin sebagai amunisi pengujian.

Pengujian difokuskan pada tiga aspek utama, yaitu:

1. Apakah mekanisme *Idempotency Key* berbasis Redis mampu mendeteksi dan menolak permintaan duplikat secara instan di tingkat memori RAM dengan kode respons HTTP 409 Conflict.
2. Apakah lapisan pertahanan basis data (*Database Absolute Lock*) mampu menahan serangan jeda waktu setelah masa berlaku kunci memori (*Time-To-Live / TTL*) pada Redis habis.
3. Apakah sistem *secure* terbukti mampu mencegah anomali pencetakan hak akses (*Token Tes*) ganda pada basis data untuk satu ID pembayaran yang sama.

Seluruh rangkaian serangan eksploitasi dikirimkan menggunakan instrumen otomasi Apache JMeter. Eksekusi pengujian ini dibagi menjadi tiga fase serangan yang bertahap. Tabel 4.7 berikut menjabarkan rincian target eksploitasi beserta parameter beban yang ditembakkan pada setiap pengujian:

Tabel 4. 7 Skenario dan Parameter Beban Pengujian

Fase Pengujian	Target Kerentanan	Beban Serangan	Kondisi Awal Sistem
Fase Pertama	<i>Race Condition & Replay Attack</i> (< 60 detik)	100 <i>Threads</i> x 1 <i>Loop</i> (Total 100 <i>Requests</i>)	Basis data berstatus PENDING dan memori Redis kosong.
Fase Kedua	<i>Replay Attack</i>	100 <i>Threads</i> x 1	Basis data berstatus



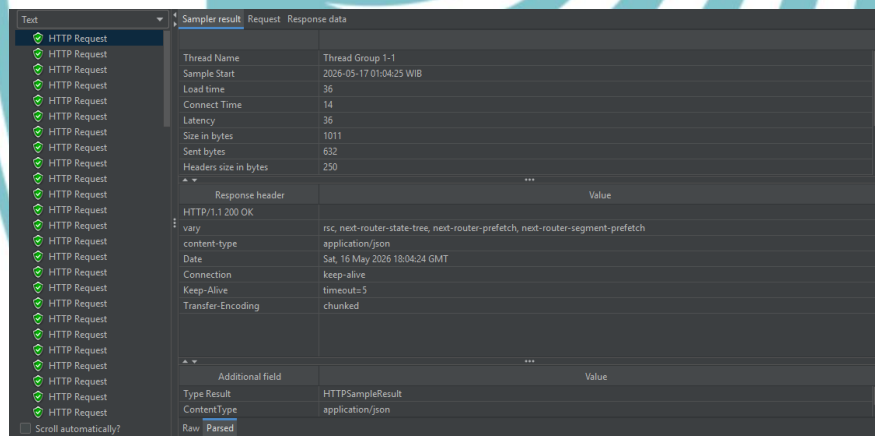
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Fase Pengujian	Target Kerentanan	Beban Serangan	Kondisi Awal Sistem
	(Eksplorasi <i>TTL Expiry</i> > 60 detik)	<i>Loop</i> (Total 100 <i>Requests</i>)	SUCCESS dan kunci Redis telah hangus.
Fase Ketiga	Uji Ketahanan Beban (<i>Stress Testing</i>)	100, 500, dan 1.000 <i>Threads</i> x 10 <i>Loop</i>	Basis data berstatus SUCCESS dan memori Redis aktif.

a. Pengujian Fase Pertama (*Race Condition & Replay Attack* < 60 Detik)

Pada percobaan pertama, serangan *High Concurrency* diarahkan menuju Sistem Rentan untuk mengevaluasi dampak ketiadaan mekanisme *Idempotency*.



Gambar 4. 61 Tampilan Respons Sistem Rentan terhadap Serangan High Concurrency

Berdasarkan Gambar 4.61 di atas, tangkapan layar dari instrumen Apache JMeter menunjukkan bahwa seluruh permintaan duplikat yang dikirimkan secara serentak (100 *threads*) berhasil dieksekusi oleh server tanpa hambatan. Karena server rentan tidak memiliki mekanisme pertahanan berbasis memori maupun penguncian basis data, sistem mengalami anomali *Race Condition* di mana setiap *thread* permintaan saling mendahului untuk memproses transaksi yang sama. Hal ini dibuktikan dengan server yang mengembalikan respons HTTP 200 OK untuk rentetan paket duplikat tersebut.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
[System Info] 1 Akses Token Tes dicetak untuk Transaksi 154.
```

Gambar 4. 62 Log Terminal Server Menunjukkan Pemrosesan Ganda

Kerentanan terhadap konkurensi lalu lintas ini terekam secara komprehensif pada sistem pemantauan server. Berdasarkan log terminal pada Gambar 4.62, server rentan secara keliru memproses logika bisnis secara berulang-ulang untuk satu identitas transaksi yang sama. Sistem mencetak riwayat log pencetakan *Token Tes* hingga puluhan kali dalam waktu sepersekian milidetik. Kegagalan ini mengonfirmasi bahwa server kehilangan kemampuan untuk melacak *state* transaksi yang sedang berjalan (*in-flight requests*).

6158	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:03:29.812	RESERVED
6159	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:24.568	RESERVED
6160	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:24.607	RESERVED
6161	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:24.606	RESERVED
6162	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:24.615	RESERVED
6163	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:24.618	RESERVED
6164	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:24.713	RESERVED
6165	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:24.718	RESERVED
6166	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:24.723	RESERVED
6167	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:24.915	RESERVED
6168	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:25.013	RESERVED
6169	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:25.013	RESERVED
6170	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:25.014	RESERVED
6171	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:25.015	RESERVED
6172	142	12	154	NULL	NULL	NULL	0 2026-05-16 18:04:25.016	RESERVED

Gambar 4. 63 Anomali Pencetakan Hak Akses Ganda pada Basis Data

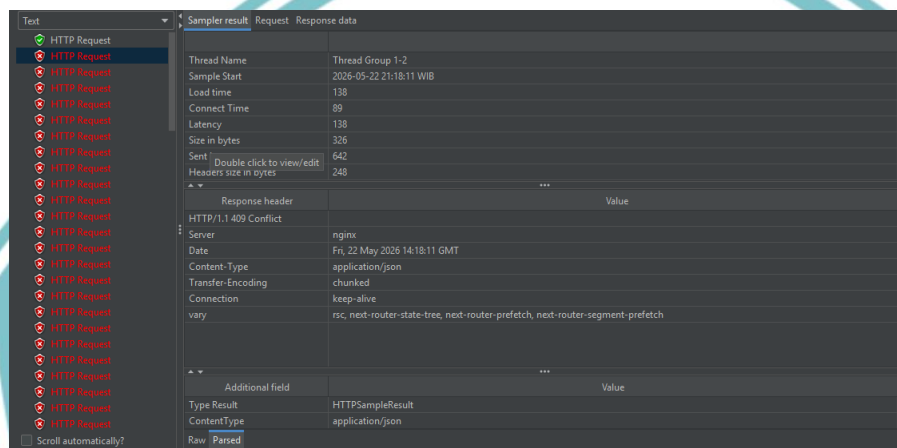
Dampak paling fatal dari eksploitasi *Race Condition* ini bermuara pada kerugian operasional dan inkonsistensi basis data. Berdasarkan Gambar 4.63 di atas, tangkapan layar dari antarmuka basis data pada tabel *TestAttempt* memperlihatkan terciptanya puluhan entitas hak akses (*Token Tes*) baru yang seluruhnya terikat pada satu ID pembayaran yang sama. Fakta ini membuktikan bahwa aplikasi sangat rentan terhadap eksploitasi pencurian komoditas digital secara masif.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Percobaan selanjutnya dilakukan dengan mengarahkan serangan menuju Sistem *Secure* guna memvalidasi efektivitas arsitektur pertahanan berlapis. Pengujian ini menggunakan metode Lingkungan Terkendali (*Controlled Environment*), di mana basis data dikondisikan pada status awal ("PENDING") dan memori Redis dalam keadaan bersih. Hal ini bertujuan menyimulasikan kondisi riil ketika 100 *payload* otentik datang saling mendahului tepat di milidetik pertama sebelum server sempat memproses apa pun.



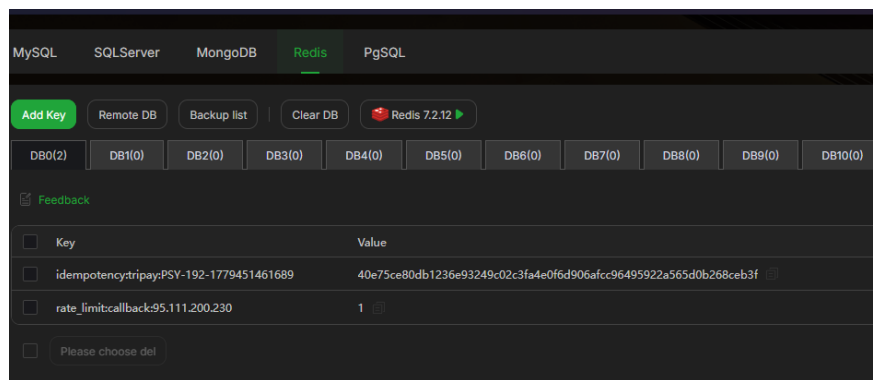
Gambar 4. 64 Penolakan Race Condition secara Presisi pada Lapisan Redis

Berdasarkan tangkapan layar Apache JMeter pada Gambar 4.64, terbukti bahwa mekanisme *Idempotency Key* beroperasi dengan presisi komputasi yang sangat tinggi. Sistem secara cerdas berhasil mengidentifikasi 1 permintaan tercepat (*first-in*) untuk diproses secara sah (ditandai dengan satu indikator respons HTTP 200 OK), dan pada milidetik yang sama, sistem langsung memblokir 99 permintaan duplikat lainnya yang datang terlambat (*late-arrivals*) dengan mengembalikan respons HTTP 409 Conflict.



Hak Cipta :

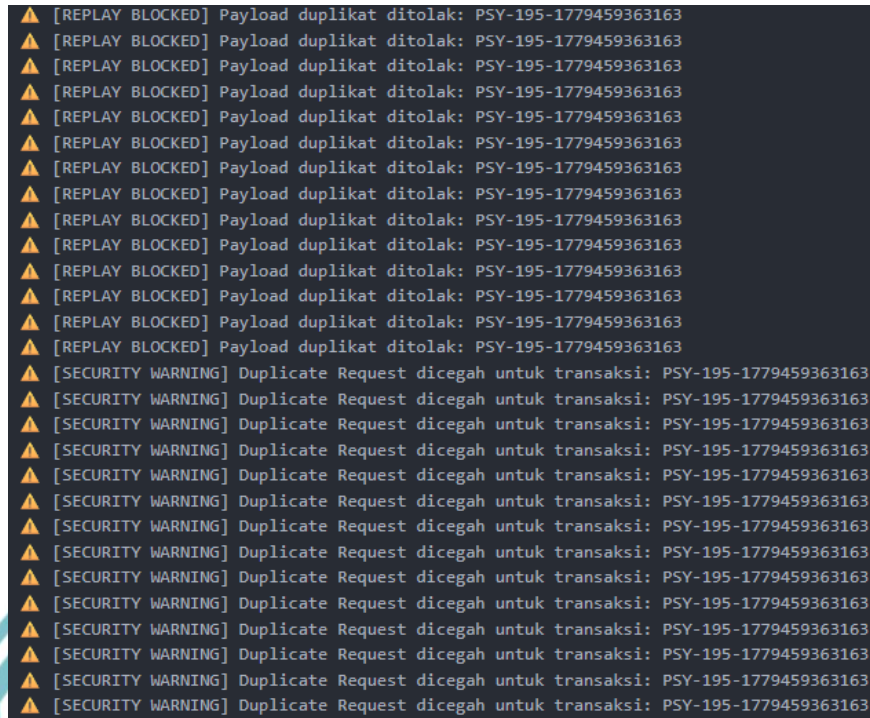
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4. 65 Pencatatan Idempotency Key dan Mitigasi Lonjakan Beban pada Redis

Keberhasilan mitigasi tersebut terekam secara konkret pada antarmuka *in-memory database* Redis. Berdasarkan Gambar 4.65, mekanisme pertahanan terbukti meredam dampak konkurensi dengan menciptakan dua entitas kunci (*keys*) krusial. Kunci *Idempotency* berhasil diciptakan oleh permintaan pertama sebagai gembok absolut, sementara kunci *Rate Limit Callback* mencatat secara akurat nilai komputasi sebesar 100 *requests* dari IP penyerang. Hal ini mengonfirmasi bahwa 99 *requests* duplikat berhasil diproses dan ditahan murni di tingkat RAM, tanpa membebani basis data utama.

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



request dicegah. Jejak audit ini menegakkan pekerja secara aktif menahan *payload* berjenis yang sama, tepat di milidetik yang sama s

Tindakan mitigasi pada tingkat memori tersebut terekam secara transparan pada log operasional server. Berdasarkan Gambar 4.66, terminal secara beruntun mencetak peringatan [REPLAY BLOCKED] Payload duplikat ditolak dan [SECURITY WARNING] Duplicate Request dicegah. Jejak audit ini menegaskan bahwa algoritma *Idempotency* bekerja secara aktif menahan *payload* berulang yang memiliki identitas transaksi yang sama, tepat di milidetik yang sama saat serangan terjadi.

Sebagai bentuk eskalasi pengujian, fase kedua dilakukan untuk menguji skenario eksploitasi masa kedaluwarsa memori (*TTL Expiry*). Permintaan duplikat yang sama ditembakkan ulang menggunakan JMeter setelah memberikan jeda waktu lebih dari 60 detik. Hal ini bertujuan mengeksploitasi celah ketika seluruh kunci *Idempotency* dan *Rate Limit* pada Redis telah hangus terhapus secara otomatis.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

LIMIT ? OFFSET ?
[DB BLOCKED] Payment ID 195 sudah LUNAS di Database! Replay Attack (TTL Expiry) dig
agalkan.

```

Gambar 4. 67 Penolakan Serangan Jeda Waktu (TTL Expiry) pada Lapisan Database

Berdasarkan Gambar 4.67, meskipun lapisan pertahanan pertama (Redis) telah berhasil dilewati akibat ketiadaan kunci di memori, muatan duplikat tersebut tetap gagal mengeksploitasi sistem berkat mekanisme *Database Absolute Lock* (lapisan pertahanan dalam). Log terminal secara transparan mencatat interupsi: [DB BLOCKED] Payment ID 195 sudah LUNAS di Database! Replay Attack (TTL Expiry) digagalkan. Server memvalidasi status final pada basis data, menyadari bahwa tagihan telah berstatus SUCCESS, dan membatalkan pencetakan token ganda secara absolut.

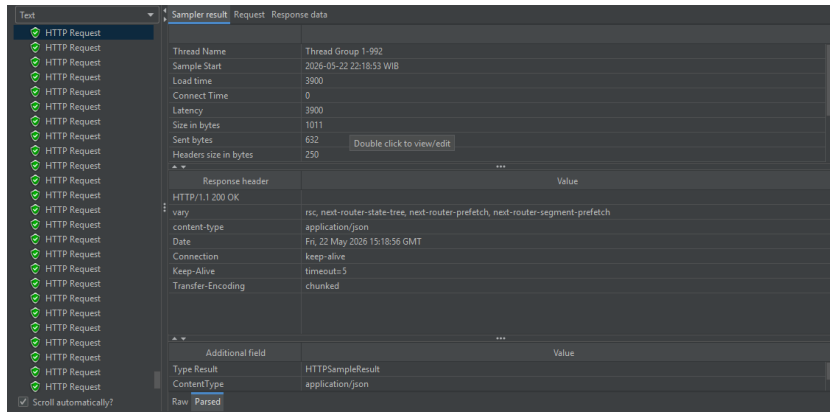
c. Pengujian Fase Ketiga (Pengujian Performa dan Tegangan / *Load & Stress Testing*)

Pengujian fase ketiga dirancang sebagai *Load & Stress Testing* untuk mengevaluasi ketersediaan layanan dan keandalan sistem ketika dihadapkan pada ancaman *Replay Attack* bervolume masif yang berpotensi memicu efek kelumpuhan (*Denial of Service*). Mengacu pada prinsip pengujian performa dan tegangan, serangan tidak dilakukan secara konstan, melainkan menggunakan metode bertahap (100, 500, hingga 1.000 *threads* dengan *Loop Count* 10) untuk mengamati titik degradasi performa dan memicu kelelahan sumber daya (*resource exhaustion*) pada perangkat keras server (VPS).



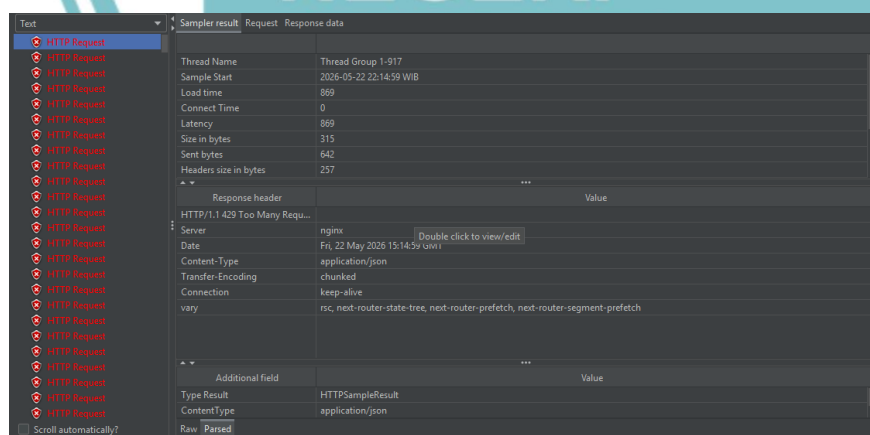
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4. 68 Ilusi Ketahanan dan Bencana Resource Exhaustion pada Sistem Rentan

Pada Sistem Rentan, ketiadaan mekanisme pembatasan laju (*Rate Limiter*) menyebabkan puluhan ribu permintaan diteruskan secara mentah menuju logika basis data. Berdasarkan Gambar 4.68, sekilas server terlihat mampu menangani gempuran tersebut yang ditandai dengan dominasi respons HTTP 200 OK. Namun, hal ini merupakan ilusi ketahanan (*False Resilience*) yang berakibat sangat fatal. Keberhasilan pengembalian respons tersebut berarti Node.js secara paksa mengeksekusi 10.000 kueri duplikat ke dalam basis data. Akibatnya, sistem mengalami *resource exhaustion* (kelelahan sumber daya); basis data dibanjiri oleh ribuan pencetakan Token Tes sampah (*data bloat*), dan metrik utilisasi CPU perangkat keras melonjak tajam untuk memproses instruksi berulang tersebut.



Gambar 4. 69 Stabilitas Pemulihan Layanan dan Pencegahan Efek DoS pada Sistem Secure



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Kondisi operasional yang sangat kontras terekam pada Sistem *Secure* (Gambar 4.69). Ketika dibombardir oleh beban ekstrem yang sama, mekanisme *Rate Limiter* berbasis Redis langsung mengidentifikasi lonjakan lalu lintas tersebut sebagai anomali. Server secara instan memotong siklus eksekusi di pintu depan (*early exit*) dengan mengembalikan kode HTTP 429 Too Many Requests. Pemrosesan berbasis memori ini mencegah puluhan ribu paket serangan menyentuh basis data, sehingga menjaga stabilitas layanan dan menghindarkan server dari kelumpuhan sistem.

Gambar 4. 70 Deteksi dan Interupsi Serangan Masif pada Terminal Server

Keberhasilan pemblokiran ini dibuktikan lebih lanjut melalui sistem pemantauan aplikasi. Berdasarkan Gambar 4.70, terminal server memunculkan peringatan interupsi [DDoS BLOCKED] Terlalu banyak request dari IP: 103.121.180.53 secara masif. Penggunaan label peringatan "DDoS" pada log ini merupakan indikator perlindungan bawaan yang menandakan bahwa sistem secara aktif meredam serbuan *Replay Attack* masif dari satu alamat IP yang berisiko melumpuhkan server (efek DoS). Catatan log ini memberikan bukti empiris bahwa



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

arsitektur keamanan mampu mendeteksi dan memotong siklus eksekusi ancaman tepat di garda terdepan. Rincian komparasi dampak beban masif ini terhadap metrik performa (*Throughput*, Latensi, CPU, dan *Memory*) akan dibahas secara kuantitatif pada bagian Data Hasil Pengujian.

4.4.3 Data Hasil Pengujian

4.4.3.1 Data Hasil Pengujian Kerentanan Manipulasi Panjang Signature (Buffer Mismatch)

Pengujian kerentanan manipulasi panjang *signature* (*Buffer Mismatch*) bertujuan untuk mengevaluasi tingkat ketahanan stabilitas *runtime* server dan keandalan penanganan pengecualian (*Exception Handling*) ketika dihadapkan pada muatan data yang tidak standar. Pengujian ini secara spesifik menargetkan celah kelemahan pada fungsi komparasi kriptografi tingkat sistem (seperti `crypto.timingSafeEqual` pada ekosistem Node.js) yang berpotensi memicu galat komputasi (`TypeError`) apabila menerima input dua variabel dengan ukuran alokasi memori (*buffer length*) yang tidak seimbang.

Berdasarkan simulasi serangan yang telah dieksekusi menggunakan instrumen Burp Suite (fitur Repeater), diperoleh data hasil respons server terhadap empat variasi injeksi anomali. Keempat variasi eksploitasi tersebut mencakup injeksi karakter sangat pendek (*short string*), karakter berlebih yang melampaui batas arsitektur SHA-256 (*over-length*), format angka mentah (*integer mismatch*), serta deretan simbol non-heksadesimal. Pengamatan pada data hasil pengujian ini difokuskan pada dua metrik evaluasi utama: pertama, perbandingan kode status HTTP yang dikembalikan oleh server sebagai representasi tingkat keberhasilan mekanisme mitigasi; dan kedua, dampak langsung dari lolosnya *payload* manipulatif tersebut terhadap integritas pembaruan status transaksi pada tabel basis data utama.

Tabel 4.8 di bawah ini menyajikan rekapitulasi komparatif data observasi antara arsitektur sistem rentan (tanpa pra-validasi panjang karakter) dan arsitektur sistem *secure* saat memproses muatan *payload* dengan nilai *signature* yang telah dimanipulasi:



Tabel 4. 8 Data Hasil Pengamatan Kerentanan Manipulasi Panjang Signature

No	Variasi Manipulasi <i>Signature</i>	Sistem Target	Respons HTTP	Status Mitigasi
1	Injeksi Karakter Sangat Pendek (<i>Short String</i>)	Sistem Rentan	200 OK	Basis data menjadi <i>SUCCESS</i> (Mitigasi Gagal)
		Sistem Secure	400 <i>Bad Request.</i>	Basis data tidak berubah (Mitigasi Berhasil)
2	Injeksi Karakter Berlebih (<i>Over-length</i> <i>100 char</i>)	Sistem Rentan	200 OK	Basis data menjadi <i>SUCCESS</i> (Mitigasi Gagal)
		Sistem Secure	400 <i>Bad Request</i>	Basis data tidak berubah (Mitigasi Berhasil)
3	Injeksi Format Angka Mentah (<i>Integer</i> <i>Mismatch</i>)	Sistem Rentan	200 OK	Basis data menjadi <i>SUCCESS</i> (Mitigasi Gagal)
		Sistem Secure	400 <i>Bad Request</i>	Basis data tidak berubah (Mitigasi Berhasil)
4	Injeksi Simbol Non- Heksadesimal (<i>@#\$\$%^&!()*</i>)	Sistem Rentan	200 OK	Basis data menjadi <i>SUCCESS</i> (Mitigasi Gagal)

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

No	Variasi Manipulasi <i>Signature</i>	Sistem Target	Respons HTTP	Status Mitigasi
		Sistem <i>Secure</i>	400 <i>Bad Request</i>	Basis data tidak berubah (Mitigasi Berhasil)

Berdasarkan data komparatif pada Tabel 4.8 di atas, terlihat jelas disparitas performa keamanan yang sangat kontras antara kedua arsitektur server yang diuji. Pada Sistem Rentan, seluruh variasi serangan (100%) terbukti berhasil menembus barisan pertahanan API tanpa adanya hambatan logika. Ketiadaan fungsi pengecekan ukuran variabel di awal menyebabkan server rentan secara sepihak memproses seluruh muatan data tanpa memverifikasi keabsahan format *signature*. Dampak dari kelalaian ini berujung pada terjadinya kebocoran fungsionalitas kritis, di mana status pembayaran pada tabel basis data diperbarui secara ilegal menjadi sukses dan memicu sistem untuk menerbitkan token layanan secara cuma-cuma kepada penyerang.

Sebaliknya, pada Sistem *Secure*, rasio serangan yang lolos berhasil ditekan secara mutlak hingga mencapai angka 0%. Implementasi fungsi pra-validasi ukuran dan bentuk karakter mampu mengisolasi seluruh pola anomali tepat di lapisan terdepan siklus permintaan (*request life-cycle*). Server *secure* terbukti memiliki keandalan *Exception Handling* yang sangat baik, di mana sistem tidak mengalami kelumpuhan operasional (*crash*) ataupun interupsi *runtime* saat dipaksa membaca format data yang menyimpang, melainkan langsung memutus aliran eksekusi secara aman (*early exit*).

Sebagai bukti pembandingan yang konkret terhadap hasil klasifikasi tersebut, berikut adalah representasi keluaran (*output*) pesan respons balik berformat JSON yang dikembalikan oleh masing-masing arsitektur server saat menerima injeksi data dari instrumen Burp Suite:



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Representasi Respons pada Sistem Rentan (Gagal Mitigasi):

```
HTTP/1.1 200 OK

content-type: application/json

{
  "success": true,
  "payment": {
    "id": 174,
    "userId": 142,
    "testTypeId": 12,
    "amount": 120000,
    "method": "BRIVA",
    "reference": "DEV-T45859370322Q5EAV",
    "status": "SUCCESS",
    "payload": {
      "status": "PAID",
      "note": "Simulated webhook payload for penetration
testing"
    },
    "paidAt": "2026-05-17T11:29:33.316Z"
  }
}
```



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Representasi Respons pada Sistem Secure (Berhasil Mitigasi):

```
HTTP/2 400 Bad Request

Content-Type: application/json

{

  "success": false,

  "message": "Invalid signature"

}
```

Melalui peninjauan struktur kode respons di atas, perbedaan penanganan lalu lintas data ilegal menjadi semakin valid. Sistem rentan memberikan umpan balik berupa status HTTP 200 OK disertai pengembalian objek data payment baru secara lengkap. Keberadaan atribut "status": "SUCCESS" dan tercatatnya stempel waktu pelunasan secara otomatis pada atribut "paidAt" di dalam respons menjadi bukti mutlak bahwa muatan data manipulatif berstatus pelunasan palsu telah dianggap sebagai transaksi yang sah oleh server rentan.

Sementara itu, arsitektur sistem *secure* secara konsisten memutus siklus eksekusi dan langsung mengembalikan penolakan tegas secara murni melalui kode status HTTP 400 Bad Request. Penolakan ini dipertegas dengan pengembalian format JSON yang rapi berupa "success": false dan pesan galat "message": "Invalid signature". *Output* tersebut menegaskan bahwa arsitektur pertahanan yang diusulkan berhasil mengamankan logika bisnis aplikasi dan mengisolasi permintaan ilegal seketika itu juga pada lapisan terdepan jaringan guna mencegah risiko eksploitasi *Buffer Mismatch*.

Selain menganalisis struktur respons HTTP dan integritas basis data, pengujian ini juga mengevaluasi aspek efisiensi komputasi server melalui pengukuran Waktu Respons (*Response Time*). Parameter ini diukur secara empiris dalam satuan milidetik (ms) untuk membandingkan seberapa cepat kedua arsitektur menangani



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

dan menggagalkan muatan anomali.

Tabel 4. 9 Komparasi Waktu Respons Pemrosesan Anomali

Variasi Serangan	Waktu Respons Rentan (200 OK)	Waktu Respons <i>Secure</i> (400 Bad Request)
Karakter Sangat Pendek	20.7 ms	58.8 ms
Karakter Berlebih (100 char)	23.7 ms	58.7 ms
Angka Mentah (<i>Integer</i>)	22.5 ms	65.5 ms
Simbol Non-Heksadesimal	38.8 ms	69.8 ms

Berdasarkan komparasi data pada Tabel 4.9, yang merupakan nilai rata-rata (mean) dari 10 kali pengujian untuk setiap variasi serangan, sistem rentan mencatat waktu respons yang lebih rendah, yaitu berkisar antara 20,7 ms hingga 38,8 ms. Hal ini terjadi karena server rentan langsung memproses permintaan yang masuk tanpa melakukan validasi panjang signature maupun inspeksi payload secara memadai. Meskipun menghasilkan waktu respons yang lebih cepat, arsitektur ini terbukti gagal menjaga integritas logika bisnis karena seluruh payload manipulatif tetap diterima dan berhasil memodifikasi status transaksi pada basis data.

Sebaliknya, sistem secure mencatat waktu respons yang relatif lebih tinggi, yaitu berkisar antara 58,7 ms hingga 69,8 ms. Peningkatan waktu pemrosesan tersebut merupakan konsekuensi dari penerapan mekanisme keamanan tambahan yang meliputi pra-validasi panjang karakter, pemeriksaan format signature, serta verifikasi HMAC sebelum permintaan diproses lebih lanjut. Meskipun terdapat penambahan latensi dalam skala milidetik, seluruh variasi serangan berhasil ditolak melalui respons HTTP 400 Bad Request sehingga payload manipulatif tidak pernah mencapai lapisan logika bisnis maupun basis data. Hasil ini menunjukkan bahwa tambahan waktu pemrosesan yang terjadi merupakan security overhead yang wajar dan sebanding dengan peningkatan tingkat keamanan sistem yang diperoleh.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4.4.3.2 Data Hasil Pengujian Simulasi Manipulasi Transmisi (Webhook Spoofing)

Pengujian simulasi manipulasi transmisi (*Webhook Spoofing*) bertujuan untuk mengevaluasi perilaku arsitektur API *callback* saat menghadapi vektor serangan pemalsuan data di tengah jalur transmisi. Pada skenario ini, eksploitasi disimulasikan dengan mengambil *header* otentik X-Callback-Signature dari transaksi yang secara riil masih berstatus "UNPAID". Aktor ancaman kemudian memanfaatkan *signature* asli tersebut dan menyandingkannya dengan muatan parameter *body* JSON yang telah dimodifikasi secara sepihak (*on-the-fly*), yakni mengubah parameter "status": "UNPAID" menjadi "status": "PAID", sebelum meneruskannya ke server target.

Pengamatan pada pengujian ini difokuskan pada kapabilitas sistem dalam melakukan validasi silang (pencocokan ulang nilai *hash*) antara *header* kriptografi yang diselundupkan dengan muatan data yang telah dimanipulasi. Secara konseptual, modifikasi sekecil apa pun pada muatan data akan menghasilkan nilai komputasi kriptografi yang sepenuhnya berbeda (dikenal sebagai efek *Avalanche* pada algoritma SHA-256). Pengujian ini akan membuktikan secara empiris apakah server mampu mendeteksi ketidaksesuaian tersebut, serta mengevaluasi dampaknya terhadap integritas pembaruan status pada tabel basis data utama.

Tabel 4.10 di bawah ini menyajikan rekapitulasi komparatif data observasi antara arsitektur sistem rentan dan arsitektur sistem *secure* saat memproses muatan *webhook spoofing*:

Tabel 4. 10 Data Hasil Pengamatan Simulasi Manipulasi (Webhook Spoofing)

Variasi Serangan	Sistem Target	Respons HTTP	Dampak pada Sistem & Status Eksekusi
Pemalsuan Status Transaksi	Sistem Rentan	200 OK	Basis data menjadi <i>SUCCESS</i>



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Variasi Serangan	Sistem Target	Respons HTTP	Dampak pada Sistem & Status Eksekusi
			(Mitigasi Gagal)
	Sistem Secure	400 <i>Bad Request.</i>	Basis data tidak berubah (Mitigasi Berhasil)

Sebagai bukti pembandingan yang konkret terhadap hasil klasifikasi tersebut, berikut adalah representasi keluaran (*output*) pesan respons balik berformat JSON yang dikembalikan oleh masing-masing arsitektur server saat menerima injeksi pemalsuan data dari instrumen Burp Suite:

Representasi Respons pada Sistem Rentan (Gagal Mitigasi):

```
HTTP/1.1 200 OK
content-type: application/json

{
  "success": true,
  "payment": {
    "id": 174,
    "userId": 142,
    "testTypeId": 12,
    "amount": 120000,
    "method": "BRIVA",
    "reference": "DEV-T75854320322T5YJK",
    "status": "SUCCESS",
```



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
"payload": {
  "status": "PAID",
  "note": "Simulated webhook payload for penetration
testing"
},
"paidAt": "2026-05-17T11:29:33.316Z"
}
```

Representasi Respons pada Sistem Secure (Berhasil Mitigasi):

```
HTTP/2 400 Bad Request
Content-Type: application/json

{
  "success": false,
  "message": "Invalid signature"
}
```

Berdasarkan data komparatif pada Tabel 4.11 dan representasi respons di atas, terbukti bahwa serangan *Webhook Spoofing* memiliki potensi eksploitasi yang sangat destruktif apabila server tidak menerapkan verifikasi integritas data yang presisi. Pada Sistem Rentan, injeksi manipulasi berhasil dieksekusi secara penuh yang ditandai dengan pengembalian respons HTTP 200 OK. Kegagalan sistem rentan terletak pada ketiadaan proses validasi silang; server secara buta mempercayai *header signature* otentik bawaan tanpa melakukan kalkulasi HMAC ulang terhadap muatan *body* JSON yang telah dimodifikasi. Akibatnya, sistem memproses data manipulatif tersebut sebagai instruksi pelunasan yang sah, memperbarui status tagihan di basis data menjadi *SUCCESS*, dan mencetak token



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

akses layanan secara otomatis. Dalam konteks operasional nyata, hal ini berpotensi menimbulkan kerugian finansial masif bagi penyedia layanan.

Sebaliknya, pada Sistem *Secure*, mekanisme verifikasi kriptografi HMAC-SHA256 terbukti mampu mendeteksi kerusakan integritas data dengan tingkat presisi yang absolut. Walaupun penyerang dengan sengaja menyertakan *header signature* yang sah dari *payment gateway* (yang sebenarnya di-generasikan secara spesifik untuk status *UNPAID*), modifikasi atribut "status": "PAID" di dalam *body* JSON menyebabkan hasil kalkulasi *hash* baru di sisi server menjadi tidak cocok (*mismatch*) dengan *signature* bawaannya. Sistem secara tegas memutus siklus pemrosesan logika bisnis, menolak berinteraksi dengan basis data pada lapisan pra-validasi, dan mengembalikan respons HTTP 400 Bad Request beserta pesan "message": "Invalid signature". Evaluasi komparatif ini memberikan konfirmasi bahwa arsitektur keamanan yang diusulkan mampu bertindak sebagai perisai yang kebal terhadap upaya pemalsuan muatan transmisi.

Sebagai parameter evaluasi tambahan, pengujian ini juga mencatat perbandingan Waktu Respons (*Response Time*) menggunakan instrumen Burp Suite dalam satuan milidetik (ms). Berikut adalah data komparasi waktu respons antara kedua arsitektur yang dieksekusi dalam tiga kali percobaan saat memproses muatan (*payload*) *Webhook Spoofing*:

Tabel 4. 11 Komparasi Waktu Respons pada Skenario Webhook Spoofing

Pengujian	Sistem Rentan (200 OK)	Sistem Secure (400 Bad Request)
Uji 1	21 ms	59 ms
Uji 2	20 ms	60 ms
Uji 3	19 ms	61 ms
Uji 4	22 ms	61 ms
Uji 5	20 ms	63 ms
Uji 6	21 ms	64 ms
Uji 7	25 ms	68 ms
Uji 8	19 ms	70 ms



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Pengujian	Sistem Rentan (200 OK)	Sistem Secure (400 Bad Request)
Uji 9	20 ms	73 ms
Uji 10	23 ms	73 ms
Rata-rata (Mean)	21 ms	65,2 ms

Berdasarkan Tabel 4.11, pengukuran waktu respons dilakukan sebanyak 10 kali percobaan berturut-turut menggunakan karakteristik muatan (*payload*) *Webhook Spoofing* yang setara pada kedua arsitektur sistem. Hasil pengujian menunjukkan bahwa sistem rentan mencatatkan rata-rata waktu respons yang sangat singkat, yaitu sebesar 21 ms, sedangkan sistem *secure* mencatatkan rata-rata waktu respons yang sedikit lebih tinggi, yaitu sebesar 65,2 ms.

Selisih waktu komputasi tersebut merepresentasikan *Security Overhead* yang wajar dan mutlak diperlukan demi keandalan sistem. Pada sistem rentan, peladen (*server*) secara buta menelan data manipulatif tanpa melakukan validasi kriptografi apapun, lalu secara langsung meneruskannya ke operasi pembaruan basis data. Pada kondisi pengujian tunggal tanpa hambatan antrean (kondisi *idle*), proses eksekusi langsung ini berjalan sangat cepat sehingga menghasilkan waktu respons yang sangat rendah. Namun, kecepatan komputasi ini menghasilkan konsekuensi yang fatal, yaitu lolosnya eksploitasi pemalsuan status transaksi.

Sebaliknya, peningkatan waktu respons pada sistem *secure* terjadi karena adanya kewajiban dari prosesor (CPU) untuk mengalokasikan siklus komputasinya guna mengeksekusi algoritma kriptografi HMAC-SHA256 di lapisan *middleware*. Sistem memerlukan waktu tersebut untuk mengkalkulasi ulang *hash* dari muatan data sebelum dapat membandingkannya dengan *signature* asli. Meskipun menimbulkan sedikit penambahan waktu (*overhead*), komputasi matematika ini terbukti sukses mengidentifikasi anomali pemalsuan data. Setelah teridentifikasi sebagai *Spoofing*, sistem langsung menghentikan siklus pemrosesan di memori RAM (*early exit*) dengan mengembalikan respons HTTP 400 *Bad Request*. Kompromi performa waktu sepersekian milidetik ini merupakan pertukaran (*trade-off*) yang sangat sepadan dan efisien demi menjaga integritas basis data secara mutlak dari manipulasi finansial.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4.4.3.3 Data Hasil Pengujian Kerentanan Race Condition, Replay Attack, dan Evaluasi Ketahanan Beban (Stress Testing)

Pengujian pada skenario ketiga ini dibagi menjadi tiga fase eksploitasi menggunakan instrumen otomasi Apache JMeter. Guna menyajikan hasil pengujian secara objektif, data metrik performa komputasi disandingkan secara langsung dalam satu tabel komparasi beserta pencatatan dampaknya pada tingkat basis data.

a. Data Hasil Pengujian Fase Pertama (Race Condition & Replay Attack < 60 Detik)

Pada pengujian fase pertama, simulasi serangan *High Concurrency* dilakukan menggunakan instrumen Apache JMeter. Simulasi dieksekusi dengan menembakkan muatan duplikat sebanyak 100 *threads* secara serentak dalam kurun waktu 1 detik. Pengujian ini dilakukan sebanyak 5 kali siklus percobaan untuk membandingkan stabilitas performa antara Sistem Rentan dan Sistem Secure pada kondisi awal yang identik, yaitu status transaksi di basis data dikondisikan *PENDING* dan memori Redis dalam keadaan kosong pada setiap awal pengujian.

Tabel 4.12 di bawah ini merangkum komparasi metrik performa komputasi dari instrumen JMeter untuk mengobservasi tingkat latensi dan kapasitas penyelesaian (*Throughput*) server secara langsung:

Tabel 4. 12 Data Hasil Komparasi Fase Pertama (Beban 100 Threads < 60 Detik)

Sistem Target	Uji	Avg Latency (ms)	Throughput (req/s)
Sistem Rentan	1	1857	24.1
	2	2457	23.9
	3	3162	20.8
	4	3762	20.7
	5	3887	17.9
Rentan (Mean)	-	3025	21.5
Sistem Secure	1	917	21.5



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

	2	1261	24.2
	3	1109	24.9
	4	1164	22.3
	5	1122	23.8
Secure (Mean)	-	1115	23.9

Berdasarkan data pada Tabel 4.12 yang diperoleh dari lima kali pengujian berturut-turut menggunakan beban 100 *threads* secara simultan, Sistem Rentan menunjukkan rata-rata latensi sebesar 3.025 ms dengan *throughput* rata-rata 21,5 *requests/second*. Terlihat adanya peningkatan latensi yang cukup signifikan dari pengujian pertama hingga pengujian kelima, yaitu dari 1.857 ms menjadi 3.887 ms. Pada saat yang sama, nilai *throughput* mengalami penurunan dari 24,1 *requests/second* menjadi 17,9 *requests/second*. Kondisi ini menunjukkan bahwa seluruh permintaan duplikat diproses secara bersamaan hingga mencapai basis data, sehingga memicu antrean transaksi (*Disk I/O*) yang semakin panjang dan meningkatkan beban pemrosesan server. Dampaknya, sistem mencatat *Error Percentage* sebesar 0,00% karena seluruh permintaan diterima, namun menghasilkan 100 token ilegal pada satu transaksi yang sama sebagai indikasi terjadinya *Race Condition*.

Sebaliknya, Sistem Secure mencatat rata-rata latensi yang jauh lebih efisien, yakni sebesar 1.115 ms dengan *throughput* rata-rata 23,9 *requests/second*. Nilai latensi pada setiap pengujian cenderung lebih rendah dan relatif stabil dibandingkan Sistem Rentan, sementara *throughput* tetap terjaga pada kisaran 22,3 hingga 24,9 *requests/second*. Hasil ini menunjukkan bahwa mekanisme *Redis Lock* berhasil membatasi akses bersamaan terhadap transaksi yang sama dengan hanya mengizinkan satu permintaan untuk diproses, sedangkan 99 permintaan lainnya seketika ditolak menggunakan respons HTTP 409 *Conflict*. Efektivitas mekanisme tersebut tercermin dari *Error Percentage* sebesar 99,00% yang menunjukkan keberhasilan penolakan permintaan duplikat, serta hanya tercetak satu token valid pada basis data tanpa adanya duplikasi data maupun anomali *Race Condition*.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

b. Data Hasil Pengujian Fase Kedua (Replay Attack - Eksploitasi TTL Expiry > 60 Detik)

Sebagai bentuk eskalasi pengujian, simulasi serangan *Replay Attack* tahap kedua dilakukan dengan memberikan jeda waktu lebih dari 60 detik sebelum muatan dikirimkan ulang. Kondisi ini bertujuan untuk mengeksploitasi arsitektur penyangkal terluar, di mana kunci *Idempotency* pada memori Redis dipastikan telah hangus terhapus oleh sistem kedaluwarsa otomatis (*TTL Expiry*). Pada fase ini, pengujian difokuskan pada Sistem *Secure* untuk mengevaluasi keandalan lapisan pertahanan terdalam, yakni *Database Absolute Lock*, ketika sistem mendapati status transaksi telah bernilai *SUCCESS*.

Tabel 4.13 di bawah ini merangkum metrik kinerja jaringan dari instrumen JMeter saat 100 *threads* permintaan duplikat dikirimkan serentak untuk menembus lapisan basis data:

Tabel 4. 13 Data Hasil Pengamatan Fase Kedua (Beban 100 Threads > 60 Detik)

Sistem Target	Persentase Penolakan (<i>Error %</i>)	Metrik Kinerja (Latensi & <i>Throughput</i>)	Dampak Pencetakan di Basis Data
Sistem <i>Secure</i>	100.00%	Rata-rata Latensi: 1.672 ms <i>Throughput</i> : 24.1 req/s	0 Token Baru (Blokir Absolut)

Berdasarkan pemaparan data pada Tabel 4.14, instrumen JMeter mencatatkan persentase penolakan (*Error Percentage*) secara mutlak sebesar 100.00%. Angka ini mendeskripsikan bahwa meskipun ke-100 muatan duplikat tersebut berhasil melewati penjagaan memori Redis yang telah kosong, seluruh muatan tersebut berhasil dicegat secara komprehensif pada tingkat basis data relasional. Lapisan *Database Absolute Lock* secara presisi mendeteksi bahwa identitas transaksi tersebut telah berstatus *SUCCESS*, sehingga sistem langsung memutus siklus eksekusi dan mengembalikan status penolakan HTTP 409 Conflict. Secara fungsional, tindakan mitigasi ini memberikan hasil yang sempurna, di mana pencetakan entitas Token Tes ganda dapat digagalkan sepenuhnya (0 token baru).



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Secara performa komputasi, proses penolakan pada tingkat basis data diselesaikan dengan rata-rata waktu respons (*Average Latency*) sebesar 1.672 ms dan kapasitas *Throughput* 24.1 *requests/second*. Secara teoretis, terjadinya pelambatan waktu respons pada fase ini jika dibandingkan dengan Fase Pertama (917 ms) merupakan hal yang sangat wajar dan dapat dibenarkan. Hal ini mengonfirmasi bahwa eksekusi validasi yang dilakukan pada tingkat Disk I/O (*Harddisk/SSD*) di basis data relasional membutuhkan waktu komputasi yang lebih lama dibandingkan dengan penolakan instan di tingkat memori RAM (Redis).

c. Data Hasil Pengujian Fase Ketiga (Pengujian Performa dan Tegangan / Load & Stress Testing)

Pengujian fase ketiga dirancang sebagai pembuktian ketahanan sistem terhadap ancaman *Replay Attack* bervolume masif yang berpotensi memicu efek kelumpuhan layanan (*Denial of Service*). Pengujian ini membandingkan secara langsung performa Sistem Rentan dan Sistem *Secure* dengan menembakkan muatan secara bertahap dalam tiga eskalasi, yakni 100, 500, dan 1.000 *threads* yang masing-masing diulang sebanyak 10 kali (*Loop Count 10*).

Untuk menyajikan analisis yang komprehensif, pencatatan data tidak hanya difokuskan pada metrik instrumen jaringan (JMeter), melainkan turut merekam metrik utilisasi perangkat keras server (VPS) untuk mengamati fenomena kelelahan sumber daya (*Resource Exhaustion*). Metrik server ini membandingkan kondisi sebelum serangan (*Idle*) dan saat menerima beban puncak (*Peak*). Indikator *Load Average* digunakan untuk merepresentasikan rata-rata panjang antrean proses pada CPU (pada arsitektur 1 *Core*, nilai *Load Average* > 1.00 menandakan bahwa CPU mengalami kelebihan beban dan proses harus mengantre). Rangkuman hasil pengujian dijabarkan pada Tabel 4.14 berikut:

Tabel 4. 14 Komparasi Hasil Pengujian Ketahanan Beban (Fase Ketiga)

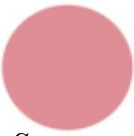
Eskalasi Beban	Sistem Target	Metrik Jaringan	Utilisasi Server (VPS 1 Core)
Tahap 1 (100	Sistem Rentan	Rata-rata Latensi:	Sebelum (<i>Idle</i>):



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Eskalasi Beban	Sistem Target	Metrik Jaringan	Utilisasi Server (VPS 1 Core)
<i>Threads</i> x 10) Total: 1.000 Req		2.418 ms <i>Throughput</i> : 36.1 req/s Penolakan (<i>Error</i>): 0.00%	CPU 0,7% <i>Load Avg</i> 0,03 Puncak (<i>Peak</i>): CPU 100% RAM 1,73 GB <i>Load Avg</i> 1,87
	 Sistem <i>Secure</i>	Rata-rata Latensi: 910 ms <i>Throughput</i> : 96.0 req/s Penolakan (<i>Error</i>): 99.90%	Sebelum (<i>Idle</i>): CPU 1,3% <i>Load Avg</i> 0,02 Puncak (<i>Peak</i>): CPU 100% RAM 1,67 GB <i>Load Avg</i> 0,77
Tahap 2 (500 <i>Threads</i> x 10) Total: 5.000 Req	Sistem Rentan	Rata-rata Latensi: 9.187 ms <i>Throughput</i> : 43.0 req/s Penolakan (<i>Error</i>): 8.14%	Sebelum (<i>Idle</i>): CPU 2,0% <i>Load Avg</i> 0,06 Puncak (<i>Peak</i>): CPU 100% RAM 1,81 GB <i>Load Avg</i> 2,20
	Sistem <i>Secure</i>	Rata-rata Latensi: 2.860 ms <i>Throughput</i> : 102.3 req/s Penolakan (<i>Error</i>): 99.98%	Sebelum (<i>Idle</i>): CPU 1,3% <i>Load Avg</i> 0,14 Puncak (<i>Peak</i>): CPU 100% RAM 1,75 GB <i>Load Avg</i> 1,13
Tahap 3 (1000 <i>Threads</i> x 10) Total: 10.000 Req	Sistem Rentan	Rata-rata Latensi: 13.632 ms <i>Throughput</i> : 58.0	Sebelum (<i>Idle</i>): CPU 1,3% <i>Load Avg</i> 0,37



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Eskalasi Beban	Sistem Target	Metrik Jaringan	Utilisasi Server (VPS 1 Core)
		req/s Penolakan (<i>Error</i>): 22.67%	Puncak (<i>Peak</i>): CPU 100% RAM 1,91 GB <i>Load Avg</i> 2,24
	Sistem <i>Secure</i>	Rata-rata Latensi: 5.369 ms <i>Throughput</i> : 94.1 req/s Penolakan (<i>Error</i>): 99.99%	Sebelum (<i>Idle</i>): CPU 2,0% <i>Load Avg</i> 0,06 Puncak (<i>Peak</i>): CPU 100% RAM 1,83 GB <i>Load Avg</i> 1,67

Berdasarkan Tabel 4.14, terlihat transisi performa dan beban server yang sangat kontras antara kondisi *idle* dan saat menerima serangan secara masif. Pada keadaan normal sebelum serangan, kedua sistem berada pada titik operasi yang sangat sehat dengan penggunaan CPU di bawah 2,0% dan *Load Average* mendekati angka 0.

Namun, seiring naiknya eskalasi serangan pada Sistem Rentan, server dipaksa untuk terus-menerus membuka koneksi antrean ke basis data. Hal ini memicu efek leher botol (*bottleneck*) yang tercermin dari rata-rata latensi yang melonjak drastis hingga 13.632 ms pada Tahap 3. Peningkatan waktu respons ini berbanding lurus dengan kelelahan perangkat keras (*hardware*). Penggunaan memori terus merangkak naik menyentuh 1,91 GB, CPU tertahan di angka 100%, dan indikator *Load Average* membengkak hingga menembus angka 2,24 (mengindikasikan antrean proses telah melampaui lebih dari dua kali lipat kapasitas ideal VPS 1 Core). Akibat terjadinya *Resource Exhaustion* tersebut, Sistem Rentan terbukti mengalami kelumpuhan layanan (*Denial of Service*) dengan rasio kegagalan jaringan mencapai 22,67% karena server terpaksa memutus koneksi secara sepihak (*Connection Timeout*).

Sebaliknya, arsitektur Sistem Secure menunjukkan daya tahan yang jauh lebih tangguh terhadap *Stress Testing*. Meskipun metrik CPU sama-sama menyentuh



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

100% akibat komputasi kriptografi (HMAC), beban antrian instruksi berhasil dikelola dengan sangat baik. Hal ini dibuktikan dari indikator *Load Average* yang terdapat di angka 1,67 pada eskalasi tertinggi (Tahap 3), jauh lebih rendah dibandingkan sistem rentan. Berkat mekanisme pemblokiran *fail-fast* berbasis memori, sistem terhindar dari tumpukan antrian basis data.

Efisiensi komputasi ini menghasilkan rata-rata latensi yang berhasil ditekan di angka 5.369 ms pada Tahap 3, serta didukung oleh kapasitas penyelesaian muatan (*Throughput*) yang konstan dan optimal di kisaran 94 hingga 102 *requests/second*. Persentase *Error* sebesar 99,99% yang terekam sepenuhnya merupakan bukti penolakan aktif secara sistematis melalui HTTP 409 *Conflict* dan 429 *Too Many Requests*. Secara komprehensif, Sistem Secure terbukti sukses memproses 1 muatan secara sah dan memblokir 9.999 muatan duplikat sisanya tanpa membahayakan ketersediaan (*availability*) layanan API.

4.4.4 Analisis Data Pengujian

4.4.4.1 Analisis Data Pengujian Kerentanan Manipulasi Panjang Signature (Buffer Mismatch)

Berdasarkan data hasil pengujian kerentanan manipulasi ukuran alokasi memori (*Buffer Mismatch*), dapat dievaluasi tingkat ketahanan stabilitas *runtime* server ketika dihadapkan pada muatan data yang menyimpang. Pengujian ini membedah keandalan penanganan pengecualian (*Exception Handling*) di ekosistem Node.js dengan menggunakan empat variasi eksploitasi: karakter sangat pendek (*short string*), karakter berlebih (*over-length*), format angka mentah (*integer mismatch*), dan simbol non-heksadesimal.

Terlihat disparitas performa keamanan yang sangat kontras antara kedua arsitektur. Pada Sistem Rentan, seluruh variasi serangan (100%) terbukti berhasil menembus barisan pertahanan *Application Programming Interface* (API) tanpa hambatan. Ketidadaan fungsi pengecekan ukuran variabel (*length check*) dan sanitasi di awal menyebabkan server secara sepihak memproses seluruh muatan data. Dampaknya berujung pada kebocoran fungsionalitas kritis, di mana status transaksi diperbarui secara ilegal menjadi SUCCESS di dalam basis data (direpresentasikan oleh



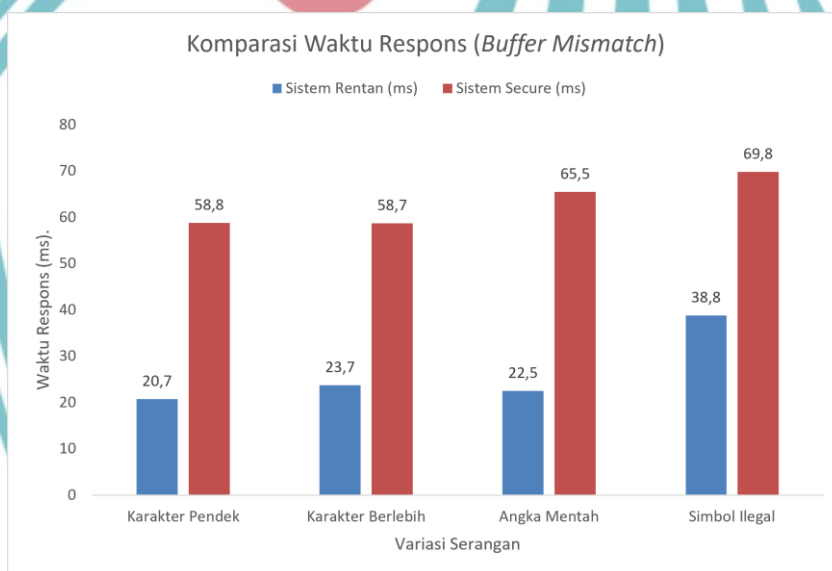
© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

kembalian HTTP 200 OK), serta memicu pencetakan token layanan secara cuma-cuma kepada penyerang.

Sebaliknya, pada Sistem Secure, rasio serangan yang lolos berhasil ditekan secara mutlak hingga mencapai angka 0%. Implementasi fungsi pra-validasi (*pre-validation*) mampu mengisolasi seluruh pola anomali tepat di lapisan terdepan jaringan. Sistem *secure* terbukti memiliki keandalan penanganan pengecualian yang sangat baik, di mana sistem tidak mengalami kelumpuhan operasional (*crash*) saat dipaksa membaca format data yang menyimpang, melainkan memutus aliran eksekusi secara aman (*early exit*) yang ditandai dengan pengembalian status HTTP 400 *Bad Request*.



Gambar 4. 71 Komparasi Waktu Respons (Buffer Mismatch)

Dampak dari ketiadaan maupun ketersediaan lapisan pra-validasi ini berkorelasi langsung terhadap efisiensi waktu komputasi server. Merujuk pada visualisasi Gambar 4.71 yang diambil dari nilai rata-rata 10 kali pengujian, Sistem Rentan mencatatkan waktu respons yang cenderung lebih rendah dan bervariasi secara fluktuatif di rentang 20,7 ms hingga 38,8 ms. Kecepatan ini bukan merupakan sebuah keunggulan, melainkan indikasi fatal bahwa server sama sekali tidak melakukan inspeksi muatan (*payload inspection*). Server langsung meneruskan muatan ilegal tersebut ke dalam antrean basis data. Keterlambatan tertinggi pada Sistem Rentan hanya terjadi pada injeksi simbol non-heksadesimal (38,8 ms), di



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

mana server sedikit mengalami *overhead* operasional saat basis data mencoba mengurai karakter ilegal tersebut.

Sebaliknya, arsitektur Sistem Secure mencatatkan waktu respons yang secara konsisten lebih tinggi, yakni berada pada rentang 58,7 ms hingga 69,8 ms. Peningkatan metrik latensi pada sistem yang aman ini dikenal sebagai fenomena kompensasi keamanan (*Security Overhead*). Sistem *secure* secara presisi mengevaluasi tipe data (menggagalkan angka mentah), memverifikasi panjang karakter (menggagalkan anomali ukuran), dan melakukan sanitasi (menggagalkan simbol ilegal) di tingkat memori (RAM) pada lapisan *middleware*.

Proses inspeksi berlapis di pintu masuk jaringan inilah yang mengonsumsi tambahan waktu pemrosesan sekitar puluhan milidetik. Namun, dalam konteks keamanan siber, *Security Overhead* ini merupakan kompensasi yang sangat efisien dan wajar (*acceptable constraint*). Sistem secara mutlak berhasil menukar tambahan waktu tersebut dengan tingkat keberhasilan mitigasi ancaman sebesar 100%, sehingga menjaga integritas basis data tetap bersih dari modifikasi instruksi yang ilegal.

4.4.4.2 Analisis Data Pengujian Simulasi Manipulasi Transmisi (Webhook Spoofing)

Berdasarkan data hasil pengujian simulasi manipulasi transmisi (*Webhook Spoofing*) yang merujuk pada Tabel 4.10 (Data Hasil Pengamatan) dan Tabel 4.11 (Komparasi Waktu Respons), dapat dievaluasi kapabilitas arsitektur *Application Programming Interface* (API) dalam mendeteksi vektor serangan pemalsuan data di tengah jalur transmisi. Pengujian ini mengukur tingkat presisi server saat menerima muatan transaksi (*payload*) yang telah dimodifikasi secara sepihak (*on-the-fly*) menjadi berstatus pelunasan, namun diselubungi menggunakan *header* kriptografi X-Callback-Signature yang otentik.

Pada sistem rentan, pengujian menunjukkan terjadinya eksploitasi fungsionalitas yang sangat fatal. Sistem secara buta mempercayai *header signature* bawaan tanpa melakukan validasi silang terhadap isi muatan *body* JSON yang telah dimanipulasi. Ketidadaan proses kalkulasi *hash* ulang ini menyebabkan server mengembalikan



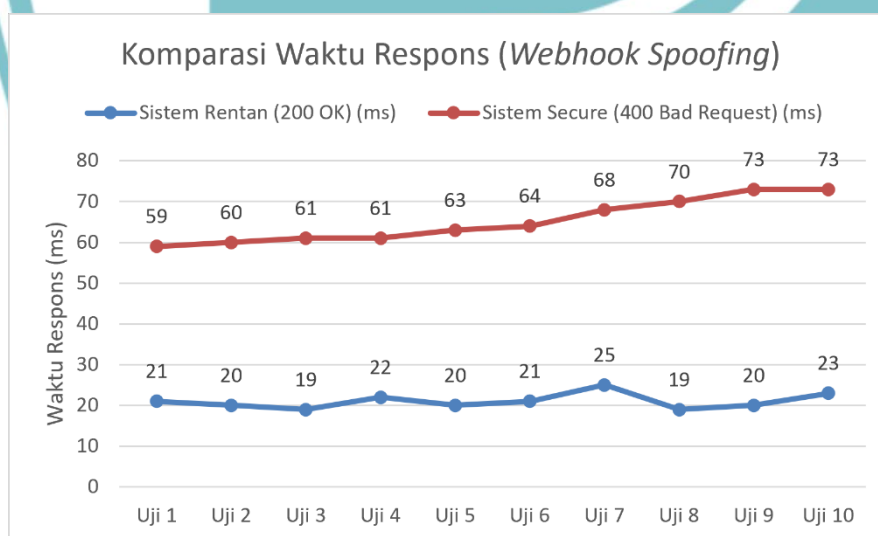
© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

respons HTTP 200 OK dan memperbarui status tagihan di dalam basis data menjadi SUCCESS. Lebih jauh, lolosnya instruksi pelunasan ilegal ini memicu tereksekusinya logika bisnis pasca-transaksi, yakni pencetakan token akses layanan bagi penyerang. Kegagalan ini mengonfirmasi bahwa tanpa verifikasi integritas data yang presisi, seluruh ekosistem aplikasi dan komoditas digital dapat dengan mudah dirampas (*bypass*) tanpa adanya aliran dana yang sah.

Sebaliknya, pada sistem *secure*, mekanisme verifikasi kriptografi HMAC-SHA256 terbukti menjadi perisai yang absolut. Sistem memanfaatkan sifat efek *Avalanche* (*Avalanche Effect*) pada algoritma SHA-256, di mana perubahan sekecil apa pun pada muatan data (dari UNPAID menjadi PAID) akan menghasilkan nilai komputasi yang sepenuhnya berbeda secara radikal. Server *secure* secara presisi mendeteksi ketidaksesuaian (*mismatch*) antara hasil kalkulasi ulang dengan *signature* yang diselundupkan. Intervensi proaktif ini ditandai dengan penghentian seluruh siklus logika bisnis, pengembalian status penolakan HTTP 400 *Bad Request*, dan pencetakan jejak audit [SECURITY BREACH] pada terminal server.



Gambar 4. 72 Komparasi Waktu Respons pada Pengujian Webhook Spoofing

Berdasarkan visualisasi pada Gambar 4.72, terlihat perbedaan pola waktu respons yang jelas antara Sistem Rentan dan Sistem Secure dalam menghadapi skenario *Webhook Spoofing*. Sistem Rentan mencatatkan rata-rata waktu respons yang sangat singkat, yaitu sebesar 20,6 ms, sedangkan Sistem Secure mencatatkan rata-



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

rata sebesar 65,2 ms. Seluruh pengujian menunjukkan bahwa waktu respons Sistem Secure sedikit lebih tinggi dibandingkan Sistem Rentan. Hal ini mengindikasikan adanya waktu komputasi ekstra (*Security Overhead*) yang wajar dan mutlak dibutuhkan untuk menjalankan mekanisme validasi keamanan kriptografi.

Pada Sistem Rentan, waktu respons terpantau sangat rendah pada rentang 19 ms hingga 25 ms. Fenomena ini menunjukkan bahwa peladen (*server*) secara buta menerima setiap muatan *spoofing* dan langsung memprosesnya hingga ke lapisan basis data tanpa adanya mekanisme validasi silang antara *header signature* dan isi *payload*. Pada kondisi pengujian tunggal tanpa beban antrean (*idle*), eksekusi basis data ini berjalan sangat cepat. Namun, kecepatan pemrosesan ini merupakan celah kerentanan yang sangat fatal, karena *server* mengeksekusi pembaruan status transaksi secara ilegal berdasarkan data yang telah dimanipulasi.

Sebaliknya, Sistem Secure menunjukkan pola waktu respons di rentang 59 ms hingga 73 ms. Peningkatan latensi sekitar 40 milidetik ini terjadi karena prosesor (CPU) memerlukan alokasi waktu untuk mengkalkulasi ulang algoritma kriptografi HMAC-SHA256 pada tahap awal pemrosesan. Ketika ketidaksesuaian (*mismatch*) teridentifikasi, sistem langsung menghentikan siklus eksekusi di tingkat memori RAM dan mengembalikan respons HTTP 400 *Bad Request* tanpa meneruskan permintaan ke lapisan aplikasi yang lebih dalam. Mekanisme *fail-fast* tersebut mengisolasi ancaman murni di garda terdepan.

Hasil analisis ini mengonfirmasi bahwa penerapan verifikasi integritas berbasis HMAC mutlak diperlukan meskipun menimbulkan sedikit penambahan waktu pemrosesan (*trade-off* performa). Kompromi waktu sepersekian milidetik ini merupakan harga yang sangat sepadan dan krusial demi mencegah *Webhook Spoofing*. Dengan memblokir permintaan tidak sah secara instan di pintu masuk jaringan, sistem berhasil menjaga integritas basis data secara mutlak dan mencegah kerugian akibat manipulasi status transaksi komoditas digital.



Hak Cipta :

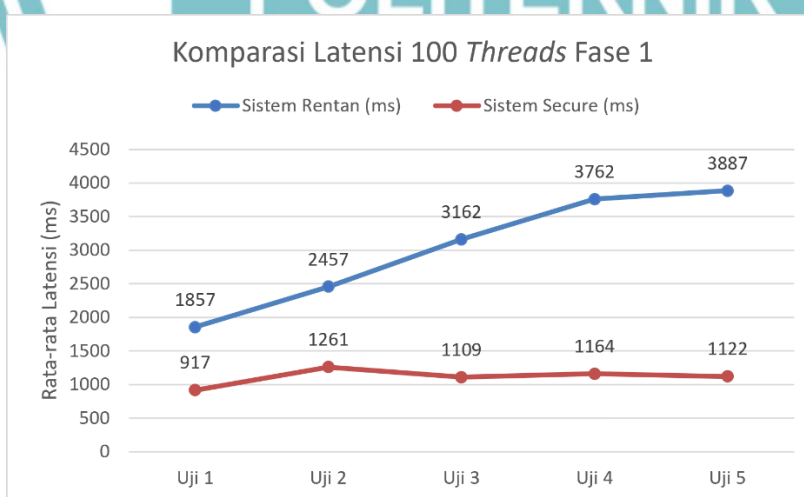
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4.4.4.3 Analisis Data Pengujian Kerentanan Race Condition, Replay Attack, dan Evaluasi Ketahanan Beban (Stress Testing)

Berdasarkan komparasi data hasil pengujian pada Skenario Ketiga, analisis pada fase ini mengevaluasi ketahanan arsitektur secara menyeluruh terhadap manipulasi konkurensi (*Race Condition*), eksploitasi jeda waktu (*Time-To-Live Expiry*), hingga kelelahan sumber daya (*Resource Exhaustion*). Pengujian ini menguji efektivitas mekanisme *Idempotency Key* dan *Rate Limiter* berbasis Redis, serta lapisan pertahanan dalam berupa *Database Absolute Lock*. Guna menyajikan analisis yang komprehensif, evaluasi dibagi berdasarkan tiga fase pengujian:

a. Analisis Fase Pertama (*Race Condition & Replay Attack* < 60 Detik)

Pada fase pengujian konkurensi tingkat tinggi (*High Concurrency*) yang dieksekusi dalam 5 siklus beruntun, ketiadaan mekanisme *Idempotency* pada sistem rentan memicu terjadinya *Race Condition* yang fatal. Karena server rentan kehilangan kemampuan melacak status transaksi yang sedang berjalan (*in-flight requests*), seluruh 100 permintaan duplikat di setiap siklusnya memenangkan kompetisi waktu tempuh dan memaksa basis data mencetak 100 entitas hak akses (Token Tes) secara berulang untuk satu transaksi yang sama.



Gambar 4. 73 Komparasi Tren Latensi pada Beban Serentak 100 Threads

Berdasarkan Gambar 4.73, terlihat adanya tren peningkatan latensi yang konsisten pada Sistem Rentan (garis biru), di mana rata-rata waktu respons meningkat dari



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

1.857 ms pada Uji 1 menjadi 3.887 ms pada Uji 5. Kenaikan latensi tersebut menunjukkan bahwa semakin banyak permintaan duplikat yang diproses secara bersamaan, semakin besar pula beban yang harus ditanggung oleh basis data dan aplikasi. Kondisi ini mengindikasikan terjadinya penumpukan antrian permintaan (request queue) akibat tidak adanya mekanisme pengendalian konkurensi, sehingga performa sistem mengalami degradasi secara bertahap selama pengujian berlangsung.

Sebaliknya, Sistem Secure (garis merah) menunjukkan pola latensi yang jauh lebih stabil dengan nilai yang berada pada kisaran 917 ms hingga 1.261 ms dan rata-rata sebesar 1.115 ms. Stabilitas ini menunjukkan bahwa mekanisme Idempotency Key dan Redis Lock mampu membatasi pemrosesan hanya pada satu permintaan yang sah, sementara permintaan duplikat lainnya ditolak lebih awal pada lapisan middleware. Dengan berkurangnya akses yang tidak perlu ke basis data, beban pemrosesan server dapat dikendalikan dengan lebih efektif sehingga waktu respons tetap terjaga meskipun sistem menerima tekanan serangan secara simultan.

b. Analisis Fase Kedua (*Replay Attack* - Eksploitasi *TTL Expiry* > 60 Detik)

Hasil pengujian pada Fase Kedua menunjukkan bahwa mekanisme keamanan yang diusulkan tidak hanya bergantung pada satu lapisan pertahanan tunggal, melainkan menerapkan prinsip pertahanan berlapis (*defense in depth*). Pada fase ini, serangan *Replay Attack* sengaja dilakukan setelah masa berlaku *Idempotency Key* berakhir sehingga lapisan *Redis Lock* tidak lagi dapat digunakan sebagai mekanisme penyaringan awal. Secara teoritis, kondisi tersebut berpotensi membuka kembali peluang eksploitasi apabila sistem hanya mengandalkan mekanisme penguncian pada memori. Namun hasil pengujian menunjukkan bahwa seluruh permintaan duplikat tetap berhasil ditolak dengan tingkat keberhasilan mitigasi sebesar 100%.

Keberhasilan tersebut mengindikasikan bahwa Database Absolute Lock berfungsi sebagai lapisan pertahanan sekunder yang efektif dalam menjaga integritas transaksi. Ketika permintaan duplikat berhasil melewati lapisan *Redis* akibat berakhirnya *TTL*, sistem masih melakukan validasi status transaksi pada basis data sebelum menjalankan logika bisnis utama. Status transaksi yang telah berubah



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

menjadi SUCCESS menyebabkan seluruh permintaan lanjutan ditolak secara otomatis menggunakan respons HTTP 409 Conflict. Dengan demikian, tidak ditemukan pencetakan token tambahan maupun perubahan data yang tidak sah pada basis data.

Dari perspektif arsitektur keamanan, hasil ini membuktikan bahwa kombinasi antara Redis Lock dan Database Absolute Lock mampu membentuk mekanisme pengamanan berjenjang yang saling melengkapi. Redis berperan sebagai lapisan pertahanan cepat (fail-fast layer) untuk menangani serangan dalam rentang waktu aktif, sedangkan basis data berfungsi sebagai sumber kebenaran akhir (source of truth) ketika lapisan memori tidak lagi tersedia. Pendekatan ini meningkatkan ketahanan sistem terhadap berbagai variasi Replay Attack, termasuk skenario eksploitasi yang dilakukan setelah masa berlaku kunci idempotensi berakhir.

Meskipun proses validasi pada basis data menghasilkan waktu respons yang lebih tinggi dibandingkan validasi pada memori Redis, peningkatan latensi tersebut merupakan konsekuensi yang wajar dari proses pemeriksaan data persisten. Dalam konteks keamanan transaksi pembayaran, tambahan waktu pemrosesan tersebut dapat diterima karena berhasil ditukar dengan jaminan integritas data dan pencegahan duplikasi transaksi secara mutlak. Oleh karena itu, hasil pengujian Fase Kedua memberikan bukti bahwa arsitektur keamanan yang diusulkan tetap mampu mempertahankan konsistensi dan keandalan sistem meskipun lapisan pertahanan terluar telah dinonaktifkan oleh mekanisme kedaluwarsa waktu.

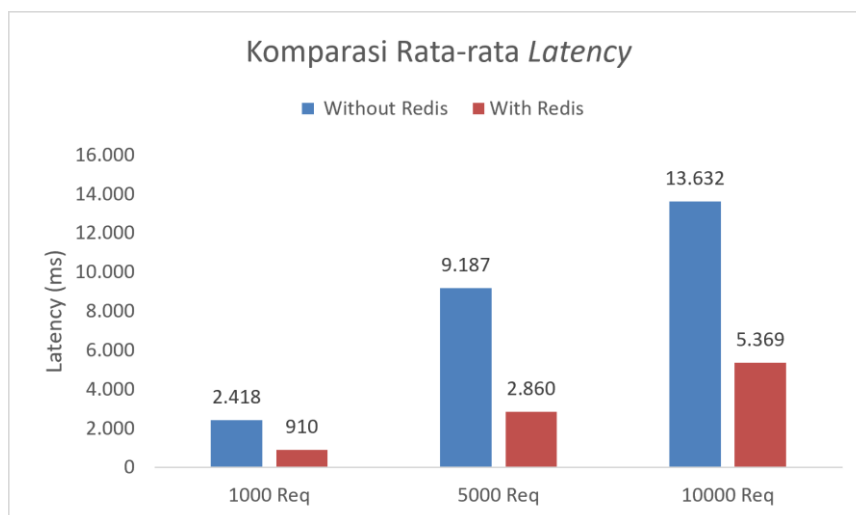
c. Analisis Fase Ketiga (Pengujian Performa dan Tegangan / Load & Stress Testing)

Dampak paling signifikan dari arsitektur *secure* terlihat secara empiris ketika sistem dihadapkan pada uji tegangan masif mulai dari 100 hingga mencapai beban puncak 10.000 *threads*. Evaluasi komprehensif pada fase ini dibedah melalui tiga indikator utama: kinerja jaringan, utilisasi perangkat keras, dan distribusi mitigasi.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4. 74 Komparasi Waktu Respons (Latensi) pada Uji Ketahanan Beban

Berdasarkan Gambar 4.74, terlihat bahwa peningkatan jumlah permintaan menyebabkan kenaikan latensi pada kedua skenario pengujian. Namun, sistem yang tidak menggunakan Redis mengalami peningkatan latensi yang jauh lebih signifikan dibandingkan sistem yang menggunakan Redis.

Pada beban 1.000 permintaan, rata-rata latensi sistem tanpa Redis mencapai 2.418 ms, sedangkan sistem dengan Redis hanya 910 ms atau lebih rendah sekitar 62,4%. Ketika beban meningkat menjadi 5.000 permintaan, latensi sistem tanpa Redis naik menjadi 9.187 ms, sementara sistem dengan Redis berada pada 2.860 ms, menunjukkan penurunan latensi sekitar 68,9%.

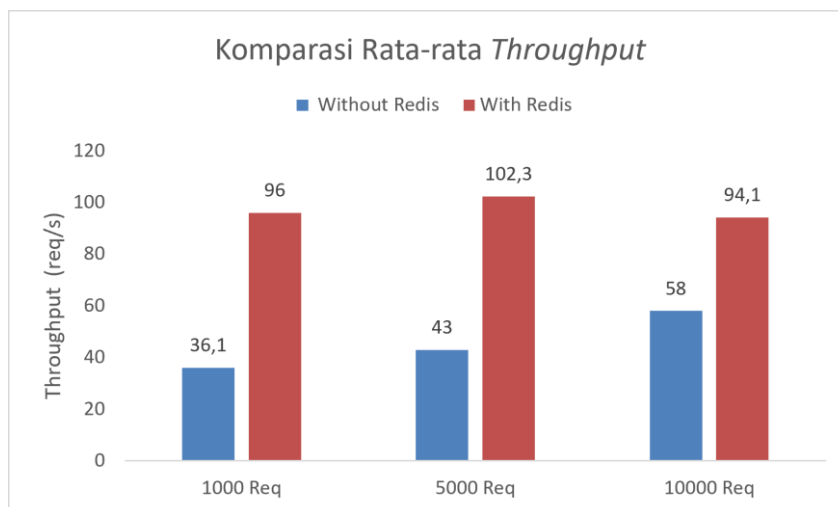
Pada beban tertinggi yaitu 10.000 permintaan, latensi sistem tanpa Redis mencapai 13.632 ms, sedangkan sistem dengan Redis sebesar 5.369 ms. Hasil ini menunjukkan bahwa penerapan Redis mampu menurunkan latensi sekitar 60,6% dibandingkan sistem tanpa mekanisme cache.

Kondisi tersebut terjadi karena Redis menyimpan data yang sering diakses pada memori sehingga permintaan tidak selalu diteruskan ke basis data. Akibatnya jumlah operasi baca-tulis ke basis data dapat dikurangi, beban Disk I/O menjadi lebih ringan, dan antrian proses pada server dapat diminimalkan. Dengan demikian, sistem mampu mempertahankan waktu respons yang lebih stabil meskipun menerima eskalasi beban yang tinggi.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4. 75 Komparasi Kapasitas Penyelesaian (Throughput) pada Uji Ketahanan Beban

Berdasarkan Gambar 4.75, terlihat bahwa peningkatan jumlah permintaan memengaruhi kapasitas penyelesaian sistem pada kedua skenario pengujian. Sistem tanpa Redis hanya mampu menghasilkan throughput sebesar 36,1 req/s pada beban 1.000 permintaan, meningkat menjadi 43 req/s pada 5.000 permintaan, dan mencapai 58 req/s pada 10.000 permintaan. Namun, peningkatan tersebut tidak sebanding dengan eskalasi jumlah permintaan yang diterima, sehingga menunjukkan adanya keterbatasan kapasitas pemrosesan pada server.

Sebaliknya, sistem yang menerapkan Redis mampu mempertahankan throughput yang jauh lebih tinggi, yaitu sebesar 96 req/s pada beban 1.000 permintaan, 102,3 req/s pada beban 5.000 permintaan, dan 94,1 req/s pada beban 10.000 permintaan. Hasil tersebut menunjukkan bahwa penggunaan Redis mampu meningkatkan kapasitas penyelesaian permintaan hingga lebih dari dua kali lipat dibandingkan sistem tanpa mekanisme cache.

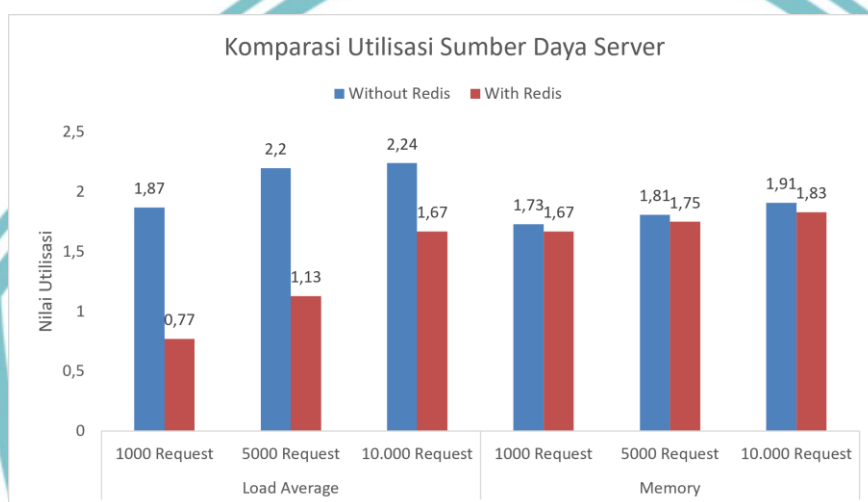
Perbedaan performa ini terjadi karena Redis menyimpan data yang sering diakses di memori sehingga jumlah akses langsung ke basis data dapat dikurangi. Akibatnya, waktu tunggu proses menjadi lebih rendah dan server dapat menyelesaikan lebih banyak permintaan dalam satuan waktu yang sama.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Selain itu, pada skenario tanpa Redis tercatat sebesar 22,67% koneksi mengalami *connection timeout* akibat meningkatnya waktu tunggu respons server. Sebaliknya, persentase interupsi sebesar 99,99% pada sistem secure berasal dari mekanisme mitigasi yang secara aktif menolak permintaan berlebih menggunakan kode respons HTTP 429 (*Too Many Requests*) dan HTTP 409 (*Conflict*), sehingga mencerminkan keberhasilan mekanisme pertahanan dalam mengendalikan lalu lintas yang masuk, bukan kegagalan layanan.



Gambar 4. 76 Komparasi Utilisasi Sumber Daya Server (*Resource Exhaustion*)

Berdasarkan Gambar 4.76, terlihat bahwa implementasi Redis dan middleware mitigasi memberikan dampak terhadap efisiensi penggunaan sumber daya server, khususnya pada metrik Load Average. Pada skenario tanpa Redis, nilai Load Average meningkat dari 1,87 pada beban 1.000 permintaan menjadi 2,24 pada beban 10.000 permintaan. Nilai tersebut menunjukkan bahwa jumlah proses yang menunggu dieksekusi semakin tinggi seiring peningkatan beban kerja server.

Sebaliknya, sistem yang menerapkan Redis mampu mempertahankan nilai Load Average yang lebih rendah, yaitu 0,77 pada beban 1.000 permintaan, 1,13 pada beban 5.000 permintaan, dan 1,67 pada beban 10.000 permintaan. Hasil ini menunjukkan bahwa mekanisme caching berhasil mengurangi beban pemrosesan server dengan meminimalkan akses langsung ke basis data sehingga antrean proses dapat ditekan.

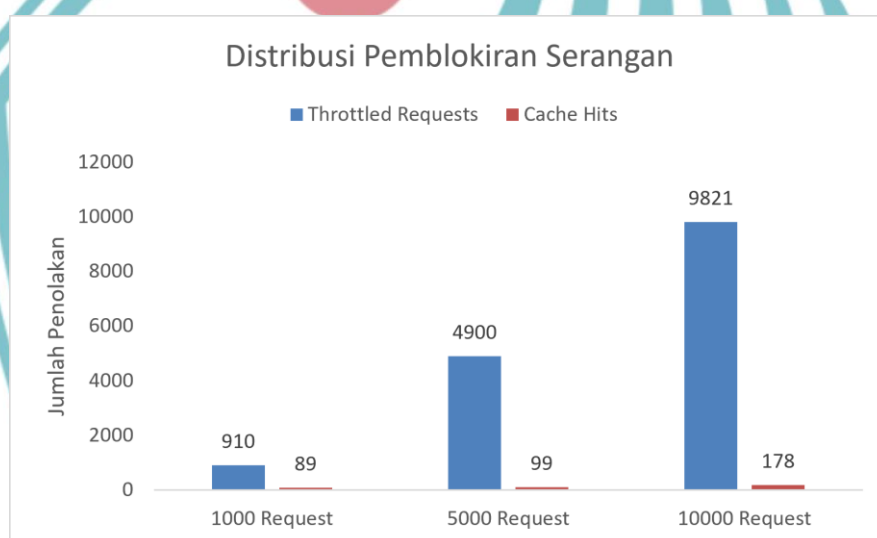


Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Pada metrik penggunaan memori, kedua skenario menunjukkan pola peningkatan yang relatif serupa seiring bertambahnya jumlah permintaan. Namun demikian, sistem dengan Redis tetap mencatat penggunaan memori yang sedikit lebih rendah dibandingkan sistem tanpa Redis. Pada beban tertinggi, penggunaan memori sistem tanpa Redis mencapai 1,91 GB, sedangkan sistem dengan Redis sebesar 1,83 GB.

Temuan ini mengindikasikan bahwa manfaat utama implementasi Redis pada penelitian ini lebih terlihat pada pengurangan beban proses (*Load Average*) dibandingkan penghematan memori. Dengan beban komputasi yang lebih rendah, server mampu mempertahankan performa yang lebih stabil ketika menghadapi eskalasi jumlah permintaan.



Gambar 4. 77 Distribusi Pemblokiran Serangan

Sebagai pelengkap, efektivitas arsitektur Defense-in-Depth dapat dievaluasi melalui distribusi pemblokiran permintaan yang ditunjukkan pada Gambar 4.77. Data pada grafik diperoleh dari log aplikasi yang mencatat respons HTTP 429 (*Too Many Requests*) yang dihasilkan oleh mekanisme Rate Limiter dan HTTP 409 (*Conflict*) yang dihasilkan oleh lapisan Idempotency Redis.

Hasil pengujian menunjukkan bahwa sebagian besar permintaan anomali berhasil dihentikan oleh lapisan pertama, yaitu Rate Limiter. Pada beban 1.000, 5.000, dan 10.000 permintaan, jumlah request yang diblokir berturut-turut mencapai 910, 4.900, dan 9.821 permintaan. Temuan ini menunjukkan bahwa mekanisme



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

pembatasan laju berhasil mencegah mayoritas lalu lintas berlebih sebelum mencapai lapisan aplikasi maupun basis data.

Sementara itu, lapisan kedua berupa Idempotency Redis berfungsi sebagai pengaman tambahan untuk mendeteksi permintaan duplikat yang masih lolos dari lapisan sebelumnya. Jumlah *cache hit* yang tercatat meningkat dari 89 pada beban 1.000 permintaan menjadi 178 pada beban 10.000 permintaan. Nilai tersebut menunjukkan bahwa masih terdapat sejumlah kecil permintaan berulang yang berhasil mencapai aplikasi, namun dapat ditolak secara efektif tanpa perlu mengakses basis data.

Pada skenario 10.000 permintaan, dari total 9.999 permintaan yang ditolak, sebanyak 9.821 permintaan (98,22%) dihentikan oleh Rate Limiter dan 178 permintaan (1,78%) ditangani oleh Idempotency Redis. Distribusi ini menunjukkan bahwa lapisan pertahanan pertama bertanggung jawab terhadap sebagian besar proses mitigasi, sedangkan lapisan Redis berperan sebagai mekanisme validasi tambahan untuk mencegah terjadinya transaksi ganda (*race condition*) dan pemrosesan ulang data yang sama.

Kunci idempotensi pada Redis dikonfigurasi menggunakan mekanisme *Time-To-Live* (TTL) selama 60 detik. Durasi tersebut dipilih untuk mengakomodasi proses *retry* yang sah dari layanan pembayaran sekaligus memastikan bahwa data kunci yang sudah tidak diperlukan dapat dihapus secara otomatis. Apabila terdapat permintaan duplikat setelah masa TTL berakhir, maka lapisan Database Absolute Lock tetap berfungsi sebagai pertahanan terakhir untuk menjamin konsistensi data transaksi.

BAB V PENUTUP

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

5.1 Kesimpulan

Penelitian ini bertujuan untuk menganalisis dan memitigasi kerentanan pada *Callback Payment Gateway* Website PsikoTest terhadap serangan *Replay Attack* dan *Webhook Spoofing*. Permasalahan utama yang ingin diselesaikan adalah rentannya arsitektur asinkron terhadap manipulasi logika bisnis dan penggandaan transaksi (*Double Spending*) akibat ketiadaan mekanisme validasi kriptografis dan penanganan konkurensi data.

Berdasarkan hasil perancangan dan serangkaian pengujian komparatif yang telah dilakukan, diperoleh kesimpulan sebagai berikut:

1. Mekanisme pertahanan berlapis (*Defense-in-Depth*) berhasil dirancang dan diimplementasikan secara komprehensif menggunakan verifikasi kriptografi HMAC-SHA256 pada lapisan pertama, mekanisme *Idempotency Key* berbasis Redis pada lapisan kedua, serta didukung *Database Absolute Lock*. Pengujian pra-validasi (*Buffer Mismatch*) menunjukkan bahwa sistem secara mutlak (100%) mampu mengisolasi empat variasi anomali format muatan secara presisi (karakter pendek, *over-length*, *integer*, dan simbol) dengan memotong siklus eksekusi di awal melalui respons HTTP 400 *Bad Request*, menjaga stabilitas *runtime* aplikasi.
2. Sistem terbukti sangat efektif dalam memitigasi serangan pemalsuan identitas dan eksploitasi logika. Pada pengujian *Webhook Spoofing*, lapisan HMAC berhasil mendeteksi dan menolak 100% modifikasi *payload* ilegal dengan waktu respons yang sangat efisien (rata-rata 65,2 ms), melindungi integritas basis data dari manipulasi sepihak. Pada skenario *Replay Attack*, algoritma *fail-fast* Redis beroperasi secara presisi dengan memproses 1 transaksi sah dan seketika memblokir 99 *request* duplikat pesaing di tingkat memori RAM (penolakan 99,00%). Lebih lanjut, pertahanan lapis ketiga (*Database Lock*) sukses memblokir eksploitasi jeda waktu secara mutlak (100%) sesaat setelah kunci memori Redis kedaluwarsa (*TTL Expiry* > 60 detik).



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

3. Penggunaan *In-Memory Data Store* (Redis) memberikan dampak positif yang sangat signifikan terhadap efisiensi dan stabilitas server saat menghadapi uji tegangan tinggi (*Load & Stress Testing*). Berkat mekanisme penolakan instan oleh *Rate Limiter* dan *Idempotency*, pada beban puncak 10.000 *threads*, sistem *secure* mampu menekan rata-rata latensi hingga 60,6% lebih rendah (5.369 ms) dibandingkan sistem rentan (13.632 ms). Server terbukti sukses mempertahankan *throughput* yang konstan di kisaran 94,1 req/s dan meredam *Load Average* di angka 1,67. Hal ini menghindarkan infrastruktur dari antrean *bottleneck* basis data dan kelumpuhan layanan (*Resource Exhaustion*) yang terjadi pada arsitektur konvensional.

5.2 Saran

Beberapa saran yang dapat dilakukan pada penelitian dan pengembangan selanjutnya, antara lain:

1. Sistem keamanan dapat dikembangkan lebih lanjut dengan mengimplementasikan mekanisme *Message Broker* (seperti Apache Kafka atau RabbitMQ) untuk mengelola antrean pembaruan data secara asinkron (*queueing*) apabila aplikasi dihadapkan pada lalu lintas *enterprise* berskala besar.
2. Perlu dipertimbangkan pengujian stabilitas dan penerapan arsitektur *Redis Cluster* atau *Redis Sentinel* guna meningkatkan ketersediaan layanan (*High Availability*) dan skalabilitas manajemen kunci memori jika API diimplementasikan secara terdistribusi pada beberapa *server*.
3. Sistem pemantauan (*monitoring*) pada terminal aplikasi dapat diintegrasikan dengan fitur notifikasi otomatis (*Alerting System*) berbasis *bot* Telegram atau surel (*email*) untuk memberikan peringatan *real-time* kepada administrator ketika terdeteksi indikasi anomali masif, seperti serangan DDoS atau pemalsuan *signature*.
4. Eksplorasi lebih lanjut dapat difokuskan dengan membandingkan performa efisiensi algoritma enkripsi asimetris (seperti RSA atau ECDSA) sebagai pendamping/alternatif HMAC-SHA256 untuk meminimalisasi risiko penyebaran



© Hak Cipta milik Politeknik Negeri Jakarta

Private Key tunggal, dengan catatan penyedia layanan *Payment Gateway* telah mendukung adopsi protokol tersebut.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



DAFTAR PUSTAKA

- Al-Fama (2024) 'Navigating the Rising Tide: Attack Trends in Financial Services', *State of the Internet / Security*, 10(5), pp. 1–36.
- Al-Omari, A. H. (2025) 'Penetration Testing of a Merchant Payment Platform; Systemic Vulnerabilities and Compliance-Centric Mitigation', *2025 IEEE International Conference on Industrial Technology (ICIT)*, pp. 1–6.
doi:<https://doi.org/10.1109/ICIT64950.2025.11049241>.
- Alulia, A. W. et al. (2023) 'Peran Krusial Jaringan Komputer dan Basis Data dalam Era Digital', *JUSTINFO (Jurnal Sistem Informasi dan Teknologi Informasi)*, 1(1), pp. 1–18.
- Baihaqie, M. R. (2021) 'Analisis dan Perancangan Sistem Pembayaran Pajak dan Retribusi Daerah secara Non-Tunai dengan Menggunakan Layanan Payment Gateway', in *Seminar Nasional Sistem Informasi (SEMNASIF)*. Yogyakarta, pp. 101–119.
- Chaudhari, B. and Verma, S. C. G. (2024) 'Achieving High-Speed Data Consistency in Financial Microservices Platforms Using NoSQL Technologies', *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, 4(2), pp. 751–758.
- Endra, H. et al. (2025) 'Memperkuat Autentikasi dan Integritas Data REST-API Menggunakan Token HMAC SHA-256', *Jurnal Minfo Polgan*, 13(2), pp. 2189–2196.
- Kurchenko, A. (2025) 'Optimization of Microservices Architecture Performance in High-Load Systems', *The American Journal of Engineering and Technology*, pp. 123–131.
- Joshi, P. K. (2024) 'Redis Cache Optimization for Payment Gateways in the Cloud', *IJETCSIT*, 4(1), pp. 28–3



Kurniawan, A., Nursyabani, N. and Matin, I. M. M. (2025) 'Analisis Keamanan Website Kota Depok Menggunakan Metode Vulnerability Assesment', *Jurnal Multinetics*, 1(1), pp. 62–73.

Mahadik, D. et al. (2025) 'To Study Webhooks for an Event-Driven Integration Mechanism', *Alochana Journal*, 14(11), pp. 35–39.

Obaid, M. A. et al. (2024) 'Exploring IoT and Blockchain: A Comprehensive Survey on Security, Integration Strategies, Applications and Future Research Directions', *Big Data and Cognitive Computing*, 8(12), p. 174.

Qiu, Z. et al. (2022) 'A Deep Study of the Effects and Fixes of Server-Side Request Races in Web Applications', in *19th International Conference on Mining Software Repositories (MSR '22)*. Pittsburgh, PA, USA: ACM, pp. 744–756.

Rainita, N. P. A. et al. (2023) 'Analisis Perbandingan Vulnerability Scanning Pada Website DVWA Menggunakan OWASP Nikto Dan Burpsuite', *Jurnal Informatika Dan Teknologi Komputer (JITEK)*, 3(2), pp. 89–97

Ranjan, P. et al. (2022) 'Threat Modeling and Risk Assessment of APIs in Fintech Applications', *ESP Journal of Engineering & Technology Advancements*, 2(2), pp. 44–61.

Salt Security (2025) *State of API Security Report, Q1 2025*. Palo Alto: Salt Security.

Shrikant, S. B. (2025) 'Enhancing Retry Policies for Webhook Delivery in Fintech Platforms', *IEEE*, pp. 1–6.

Taofik, I. et al. (2023) 'Implementasi JSON Web Token (JWT) untuk Authentication Data pada Aplikasi Bayeue Dengan Algoritma HMAC SHA-256', *Jurnal Tekno Kompak*, 17(1), pp. 1–6.

Uskono, B. M. et al. (2024) 'Analisis Keamanan Aplikasi Fintech di Indonesia: Studi Kasus OVO, GoPay, ShopeePay dan Dana', *Jurnal JIFORTY (Jurnal Ilmiah Informatika dan Teknologi)*, 5(2), pp. 177–186



Wang, Y. (2024) 'Optimizing Payment Systems with Microservices and Event- Driven Architecture: The Case of Mollie Platform', *Journal of Industrial Engineering and Applied Science*, 2(1), pp. 56–67.

Wahana, A. S. et al. (2025) 'Deteksi Dini Fraud pada Layanan Keuangan Digital Menggunakan Metode Random Forest', *Indonesian Journal on Data Science*, 3(2), pp. 97–107.

Hak Cipta :

1. Dilarang menyalin sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :

a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.

b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta

2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



DAFTAR RIWAYAT HIDUP



Jibrin Khairy Akram di Jakarta pada tanggal 29 April 2004. Penulis mulai menempuh pendidikan dasar di SDN Harapan Baru II Kota Bekasi pada tahun 2010. Setelah itu, penulis melanjutkan pendidikan ke SMPN 44 Kota Bekasi pada tahun 2016. Pada tahun 2019, penulis melanjutkan pendidikan menengah kejuruan di SMKN 5 Kota Bekasi. Setelah menyelesaikan pendidikan menengah kejuruan, penulis melanjutkan pendidikan tinggi pada tahun 2022 di Politeknik Negeri Jakarta, Jurusan Teknik Informatika dan Komputer, Program Studi Teknik Multimedia dan Jaringan.

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta





© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta





LAMPIRAN

empat: Daring (Online)

Tanggal Wawancara: 15 Januari 2026

Sumber: Pihak Klinik Yuliarpan Medika (dr. piet)

Lampiran 1 Tabel Wawancara

No	Pertanyaan	Jawaban
1	Apa kendala terbesar yang dokter rasakan dengan sistem pemesanan tes psikotes lewat WhatsApp yang berjalan sekarang?	Kendala utamanya adalah inefisiensi operasional. Admin diharuskan melayani pemesanan satu per satu, kemudian memeriksa mutasi rekening secara manual untuk mencocokkan bukti transfer. Proses ini seringkali membuat admin kewalahan ketika volume pendaftar meningkat, serta menimbulkan risiko bukti transfer terlewat atau terselip.
2	Solusi seperti apa yang diharapkan dari pengembangan website baru (cashless) yang terotomatisasi tersebut?	Solusi utama yang diharapkan adalah implementasi fitur pembayaran non-tunai (<i>cashless</i>) yang terotomatisasi. Pengguna dapat melakukan pembayaran melalui <i>Payment Gateway</i> dan status transaksi terverifikasi secara langsung (<i>real-time</i>) oleh sistem, sehingga meniadakan kebutuhan validasi bukti transfer manual ke admin. Tujuannya adalah untuk mewujudkan otomatisasi penuh pada proses transaksi.

Hak Cipta :

- Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
- Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

No	Pertanyaan	Jawaban
3.	Mengingat sistem akan berjalan secara otomatis tanpa verifikasi manual admin, apakah terdapat kekhawatiran khusus terkait validitas pembayaran?	Kekhawatiran utama terletak pada aspek keamanan validasi. Berbeda dengan mekanisme manual di mana admin dapat memeriksa mutasi bank secara langsung, sistem otomatis memiliki risiko manipulasi data di mana status pembayaran dapat dipalsukan seolah-olah telah lunas. Oleh karena itu, sistem yang dikembangkan harus memiliki mekanisme yang mampu menjamin otentikasi dan keaslian setiap notifikasi pembayaran yang masuk.
4.	Bagaimana harapan dokter jika nanti terjadi gangguan jaringan saat user melakukan pembayaran?	Prioritas utamanya adalah mencegah kerugian finansial maupun operasional. Sistem harus menjamin tidak terjadi duplikasi pencatatan di mana satu kali pembayaran tercatat dua kali, atau sebaliknya. Akurasi dan presisi data keuangan sangat krusial karena berdampak langsung pada pendapatan perusahaan.



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

No	Pertanyaan	Jawaban
5.	Apakah nantinya admin masih bersedia melakukan pengecekan manual jika ada selisih data antara uang di bank dan status di website?	Harapannya, proses rekonsiliasi manual tidak lagi diperlukan. Tujuan utama otomatisasi sistem adalah untuk mengurangi beban kerja administratif. Oleh karena itu, sistem diharapkan dapat menjamin konsistensi data yang mutlak; apabila status di website tercatat 'Lunas', maka dana harus dipastikan telah masuk dengan nominal yang tepat, tanpa adanya selisih data.

Lampiran 2 Dokumentasi Wawancara





© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

ampiran 3 Surat Pengantar Observasi



KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI POLITEKNIK NEGERI JAKARTA

Jalan Prof. Dr. G. A. Siwabessy, Kampus UI, Depok 16425
Telepon (021) 7270036, Hunting, Fax (021) 7270034
Laman: <http://www.pnj.ac.id> Posel: humas@pnj.ac.id

Nomor : 70/DST/PL3.A.9/B/KM.07.00/2026
Perihal : Permohonan Izin Observasi

19 Januari 2026

Kepada Yth.

dr. Piet Y, MARS, FISQua dan Untung Supriyadi, S.Kom., M.Pd.

Dokter Klinik Yuliarpan Medika

Jalan Raya Setu No.149, Cibuntu, Kecamatan Cibitung, Kabupaten Bekasi, Jawa Barat 17520

Dengan hormat,

Sehubungan dengan mata kuliah Skripsi yang dilaksanakan pada semester 8 (delapan) Program Studi Teknik Multimedia dan Jaringan Jurusan Teknik Informatika dan Komputer Politeknik Negeri Jakarta. Dengan ini kami mohon kesediaan Bapak/Ibu agar dapat mengizinkan mahasiswa kami untuk melakukan observasi di **Klinik Yuliarpan Medika** dengan judul penelitian "Analisis Mitigasi Serangan Replay Attack dan Webhook Spoofing pada Callback Payment Gateway Website PsikoTest Menggunakan Mekanisme Idempotency Key Berbasis Redis dan Verifikasi HMAC".

Tugas mata kuliah ini bertujuan untuk menambah wawasan terkait dengan aplikasi teori yang sudah dipelajari di Kampus dengan kondisi lapangan sebagai wadah pembelajaran dan menambah informasi mengenai mata kuliah tersebut. Adapun berikut adalah nama mahasiswa kami:

No.	Nama dan Nim	Semester/ Program Studi	Nomor Telepon dan Email	Tujuan Observasi
1	Jibrin Khairy Akram 2207421011	8 / Teknik Multimedia dan Jaringan	085966250075 jibrin.khairy.akram.tik22@mhsw.pnj.ac.id	Ingin mengetahui data-data penunjang dan wawancara untuk Skripsi

Demikian surat ini kami buat, atas perhatian dan kerjasama Bapak/Ibu kami ucapkan terima kasih.



Ketua Jurusan,

Dr. Anita Hidayati, S.Kom., M.Kom.
NIP. 197908032003122003

Tembusan :

1. Direktur;
2. Wakil Direktur Bidang Akademik;
3. Kepala Bagian Keuangan dan Umum
4. Kasubbag. Umum Politeknik Negeri Jakarta.

NEGERI JAKARTA



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

ampiran 4 Surat Izin Perusahaan Uji Coba Sistem



KLINIK YULIARPAN MEDIKA

Jl. Raya Setu No. 149, Cibuntu, Kec. Cibitung, Kabupaten Bekasi, Jawa Barat 17520

Dokter Umum
Dokter Gigi
Bidan
Laboratorium
Medical Check Up

SURAT KETERANGAN PENGGUNAAN DAN UJI COBA SISTEM

Yang bertanda tangan di bawah ini:

Nama : dr. Piet Y, MARS, FISQua.

Jabatan : Dokter

Instansi : Klinik Yuliarpan Medika

Dengan ini menerangkan bahwa sistem/aplikasi berbasis web "Psychotest Website" telah diimplementasikan, digunakan, dan diuji coba di lingkungan Klinik Yuliarpan Medika.

Berdasarkan hasil penggunaan dan uji coba yang telah dilakukan, sistem dapat digunakan untuk mendukung kegiatan operasional dan masih dalam tahap pengembangan serta penyempurnaan lebih lanjut, khususnya pada aspek keamanan sistem dan data.

Demikian surat keterangan ini dibuat untuk digunakan sebagaimana mestinya.

Bekas, 24 Mei 2026



dr. Piet Y, MARS, FISQua