LAPORAN

Disusun untuk memenuhi Ujian Akhir Semester (UAS) mata kuliah IFB-302
Keamanan Jaringan yang diberikan oleh: **Bapak Diash Firdaus, M.T.**

Disusun oleh :

1. Verenada Arsy Mardatillah (15-2023-058)
2. Aliyya Rahmawati Putri (15-2023-093)
3. Ridayanti Wardani (15-2023-168)

Kelas CC

INSTITUT TEKNOLOGI NASIONAL BANDUNG
FAKULTAS TEKNOLOGI INDUSTRI
INFORMATIKA

2025

KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga kami dapat menyelesaikan laporan ini sebagai bagian dari pemenuhan tugas akhir mata kuliah IFB-302 Keamanan Jaringan dengan judul "Penerapan Keamanan Aplikasi Web pada *Application Layer*".

Laporan ini disusun berdasarkan proyek pembuatan website dengan penerapan fitur keamanan yang berfokus pada lapisan aplikasi (*application layer*), seperti otentikasi dua faktor menggunakan Google Authenticator, pembatasan percobaan login, serta enkripsi data dalam basis data. Tujuan utama dari proyek ini adalah untuk mengaplikasikan konsep-konsep keamanan jaringan yang telah dipelajari selama perkuliahan dalam implementasi nyata berbasis web.

Dalam proses penyusunan laporan ini, kami banyak mendapatkan arahan dan dukungan dari berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Bapak Diash Firdaus, M.T., selaku dosen mata kuliah Keamanan Jaringan, atas bimbingan, ilmu, dan motivasi yang telah diberikan kepada kami.
2. Rekan-rekan kelompok, yang telah bekerja sama yang baik serta kontribusi dalam pelaksanaan proyek dan penyusunan laporan ini..

Kami menyadari bahwa dalam penyusunan dan penulisan masih banyak kekurangan dan keterbatasan. Oleh karena itu, kritik dan saran yang membangun sangat kami harapkan untuk perbaikan dan peningkatan di masa yang akan datang.

Akhir kata, semoga laporan ini dapat bermanfaat bagi pembaca, serta menjadi dokumentasi dari pembelajaran dan praktik yang telah kami lakukan. Semoga kita semua selalu diberikan kekuatan dan kesempatan untuk terus belajar dan berkarya.

Bandung, 06 Juni 2025

Kelompok 7

DAFTAR ISI

KATA PENGANTAR i
DAFTAR ISI ii
DAFTAR GAMBAR iii
DAFTAR TABEL iv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan 3
BAB II TINJAUAN PUSTAKA 4
2.1 Keamanan Jaringan pada Application Layer 4
2.2 Autentikasi Dua Faktor (2FA) 4
2.3 *Brute Force Attack* dan Sistem Pemblokiran Login 4
2.4 Enkripsi Data dalam Database 5
2.5 Studi Literatur Sejenis 5
BAB III PERANCANGAN SISTEM 6
3.1 Desain Umum Sistem 6
3.2 Struktur Folder dan File Proyek 7
3.3 Fitur Keamanan yang Diimplementasikan 9
3.4 *Tools* dan *Library* yang Digunakan 9
3.5 Perancangan Database 11
3.6 Diagram Alur Sistem 13
BAB IV IMPLEMENTASI 16
4.1 Implementasi Sistem 16
4.2 Pengujian Sistem 20
BAB V KESIMPULAN DAN SARAN 25
5.1 Kesimpulan 25
5.2 Saran 26

DAFTAR GAMBAR

Gambar 3.1 Alur Login Admin 13
Gambar 3.2 Alur Input Data oleh Pengguna 14
Gambar 4.1 Import Library pyotp 16
Gambar 4.2 Inisialisasi OTP dengan pyotp 16
Gambar 4.3 Verifikasi Kode OTP 16
Gambar 4.4 Proses Verifikasi Username dan Password 17
Gambar 4.5 Inisialisasi Session Login Setelah OTP Valid 17
Gambar 4.6 Proteksi Brute-Force 17
Gambar 4.7 Konfigurasi Timeout Session 17
Gambar 4.8 Penurunan Kunci AES dan Enkripsi RSA 18
Gambar 4.9 Enkripsi Setiap Field Laporan Keluhan 18
Gambar 4.10 enyimpanan Data Terenkripsi ke Database 18
Gambar 4.11 Pemanggilan Kunci Privat 19
Gambar 4.12 Dekripsi AES_KEY dari Bentuk Terenkripsi 19
Gambar 4.13 Dekripsi Field Terenkripsi pada Laporan 19
Gambar 4.14 Dekripsi File Bukti 19
Gambar 4.15 Halaman Login Admin 20
Gambar 4.17 Halaman input OTP 21
Gambar 4.16 Sistem menolak akses 21
Gambar 4.18 Tampilan Login Gagal 22
Gambar 4.19 Akun diblokir setelah 5 kali gagal login 22
Gambar 4.20 Data Terenkripsi di Database 22
Gambar 4.21 Dashboard Admin 23
Gambar 4.22 Daftar Pengajuan 23
Gambar 4.23 Data yang Berhasil Didekripsi 24

DAFTAR TABEL

Tabel 3.1 Struktur Folder dan File Proyek..... 7
Tabel 3.2 Tools dan Library yang Digunakan 10

BAB I
PENDAHULUAN

1.1 Latar Belakang

Keamanan sistem informasi menjadi aspek krusial dalam pengembangan aplikasi di era digital saat ini. Meskipun banyak sistem telah dilengkapi dengan proteksi pada level jaringan dan sistem operasi, kenyataannya lapisan aplikasi (*application layer*) tetap menjadi sasaran utama berbagai jenis serangan siber. Hal ini disebabkan karena lapisan aplikasi merupakan titik akhir yang berinteraksi langsung dengan pengguna, dan menjadi tempat di mana proses autentikasi, pengolahan data, serta akses ke database dilakukan.

Laporan OWASP (*Open Web Application Security Project)* menunjukkan bahwa sebagian besar kerentanan dalam sistem berasal dari kesalahan pengamanan pada lapisan aplikasi, seperti penggunaan otentikasi yang lemah, manajemen sesi yang buruk, atau tidak adanya enkripsi pada data penting. Salah satu konsekuensi dari hal tersebut adalah tingginya jumlah kasus pembobolan sistem melalui serangan *brute force, credential stuffing*, atau pencurian data yang disimpan dalam database secara *plaintext*.

Berdasarkan kondisi tersebut, kami mengembangkan sebuah aplikasi berbasis web dengan fokus utama pada penerapan mekanisme keamanan di lapisan aplikasi (*application layer*) sebagai bentuk implementasi dari tugas besar mata kuliah IFB-302 Keamanan Jaringan. Proyek ini menitikberatkan pada pengamanan proses login, perlindungan data pengguna, serta mitigasi risiko serangan terhadap sistem. Adapun fitur keamanan yang kami implementasikan antara lain:

- Autentikasi dua faktor (*Two-Factor Authentication*/2FA) menggunakan *Google Authenticator* untuk proses login admin.
- Mekanisme pemblokiran sementara (*temporary lockout*) selama 1 menit apabila terjadi 5 kali kesalahan login berturut-turut, sebagai langkah pencegahan *brute force attack*.

- Enkripsi data yang tersimpan dalam basis data menggunakan algoritma tertentu, guna memastikan kerahasiaan informasi jika terjadi kebocoran data.

Dengan pendekatan ini, kami tidak hanya menerapkan konsep keamanan jaringan secara nyata, tetapi juga memastikan bahwa aplikasi memiliki fondasi keamanan yang kuat dari sisi arsitektur aplikasi. Diharapkan proyek ini dapat menjadi salah satu contoh penerapan prinsip-prinsip keamanan sejak tahap perancangan (*security by design*), serta meningkatkan kesadaran pentingnya lapisan application layer dalam menjaga keamanan sistem informasi.

1.2 Rumusan Masalah

Merujuk pada urgensi keamanan aplikasi yang telah diuraikan dalam latar belakang, maka permasalahan yang ingin diselesaikan melalui proyek pengembangan website ini dapat dirumuskan sebagai berikut:

1. Bagaimana menerapkan autentikasi dua faktor (2FA) berbasis *Google Authenticator* pada proses login admin untuk memperkuat lapisan keamanan aplikasi?
2. Bagaimana merancang mekanisme pembatasan percobaan login gagal agar dapat mencegah serangan *brute force* secara efektif?
3. Bagaimana mengimplementasikan enkripsi pada data penting yang tersimpan di dalam database untuk menjaga kerahasiaan dan integritas data?
4. Bagaimana membangun aplikasi yang secara teknis aman sekaligus mempertimbangkan prinsip-prinsip keamanan pada lapisan application layer?

1.3 Tujuan

Pengembangan aplikasi ini dilakukan dengan tujuan utama untuk meningkatkan keamanan pada lapisan aplikasi (*application layer*) melalui serangkaian mekanisme proteksi yang dirancang secara sistematis. Tujuan khusus dari proyek ini meliputi:

1. Mengimplementasikan autentikasi dua faktor (2FA) menggunakan *Google Authenticator* pada proses login admin sebagai bentuk perlindungan terhadap akses tidak sah.
2. Merancang sistem keamanan yang mampu mendeteksi dan membatasi percobaan login gagal secara berturut-turut guna mencegah serangan *brute force*.
3. Menerapkan enkripsi terhadap data sensitif di dalam database untuk memastikan kerahasiaan informasi pengguna maupun sistem.
4. Membangun aplikasi yang berfokus pada keamanan di lapisan *application layer* dengan tetap mempertimbangkan fungsionalitas dan kenyamanan pengguna.
5. Mengaplikasikan prinsip-prinsip keamanan jaringan dalam pengembangan aplikasi nyata sebagai bagian dari pembelajaran mata kuliah IFB-302 Keamanan Jaringan.

BAB II
TINJAUAN PUSTAKA

2.1 Keamanan Jaringan pada Application Layer

Application layer merupakan lapisan tertinggi dalam model TCP/IP maupun OSI, tempat aplikasi dan layanan berinteraksi langsung dengan pengguna. Di sinilah serangan siber seringkali terjadi, karena kerentanannya terhadap eksploitasi input pengguna, manipulasi permintaan, dan akses tidak sah. Oleh karena itu, penguatan keamanan pada layer ini menjadi krusial. Menurut laporan OWASP (*Open Web Application Security Project*) tahun 2023, sepuluh besar ancaman terbesar pada web sebagian besar berasal dari sisi *application layer*, seperti *Broken Authentication, Sensitive Data Exposure, dan Security Misconfiguration.*

2.2 Autentikasi Dua Faktor (2FA)

Autentikasi dua faktor (*Two-Factor Authentication*) merupakan metode pengamanan akun yang menggabungkan dua lapisan identifikasi, biasanya berupa sesuatu yang diketahui (seperti *password*) dan sesuatu yang dimiliki (seperti kode verifikasi dari aplikasi seperti *Google Authenticator*). Dalam implementasinya, 2FA mampu mengurangi risiko akses ilegal bahkan jika kredensial utama bocor. Penelitian oleh Symantec (2022) menunjukkan bahwa 2FA dapat menurunkan keberhasilan serangan phishing hingga lebih dari 95%.

2.3 *Brute Force Attack* dan Sistem Pemblokiran Login

Brute force merupakan metode serangan yang dilakukan dengan mencoba semua kemungkinan kombinasi *username* dan *password* secara terus-menerus. Salah satu cara efektif untuk mencegah serangan ini adalah dengan menerapkan sistem pemblokiran sementara jika terdapat upaya login yang gagal berulang kali. Mekanisme ini dikenal sebagai rate limiting atau account lockout. Strategi ini terbukti mampu menahan bot dan mengurangi upaya peretasan secara signifikan.

2.4 Enkripsi Data dalam Database

Enkripsi data adalah proses mengubah informasi menjadi bentuk yang tidak dapat dibaca tanpa kunci tertentu. Dalam konteks keamanan aplikasi, enkripsi database berfungsi untuk melindungi data penting seperti informasi pribadi pengguna, kredensial, dan catatan sensitif. Algoritma enkripsi yang umum digunakan di antaranya adalah AES (*Advanced Encryption Standard*) untuk data simetris dan RSA untuk enkripsi asimetris. Penerapan enkripsi dalam database penting agar jika terjadi kompromi terhadap sistem, data yang diperoleh tidak langsung dapat dieksploitasi.

2.5 Studi Literatur Sejenis

Beberapa penelitian dan pengembangan aplikasi sebelumnya juga menekankan pentingnya keamanan pada application layer. Sebagai contoh, studi oleh Rachmawati et al. (2021) menunjukkan bahwa sistem login yang menggunakan kombinasi 2FA dan pemblokiran otomatis terhadap IP mencurigakan mampu menurunkan insiden login ilegal hingga 70%. Penelitian lain oleh Setiawan dan Nugraha (2022) juga menunjukkan efektivitas enkripsi AES dalam menjaga kerahasiaan data pengguna aplikasi berbasis web.

BAB III
PERANCANGAN SISTEM

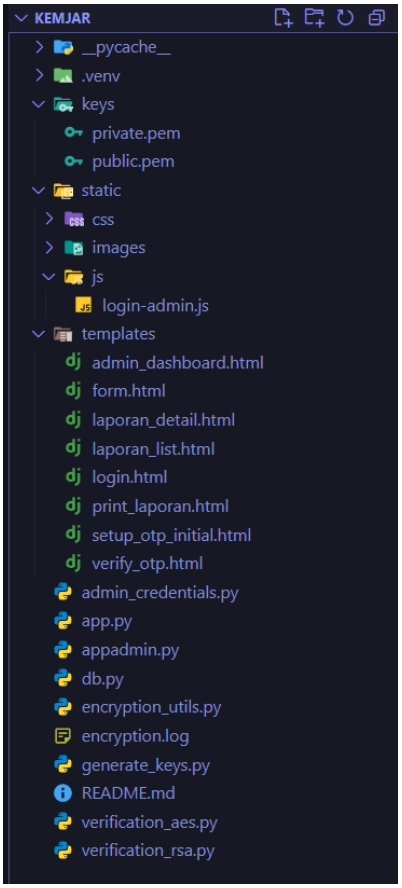
3.1 Desain Umum Sistem

Sistem yang dikembangkan merupakan aplikasi web yang berfokus pada keamanan di lapisan application layer, dengan skenario utama yaitu pengiriman data oleh user dan pengelolaan oleh admin. Untuk menjaga keamanan akses dan data, sistem menerapkan kombinasi fitur autentikasi dua faktor (2FA), proteksi *brute-force*, serta enkripsi data. Alur umum sistem dibagi menjadi dua peran utama, yaitu admin (dengan akses terbatas dan dilindungi OTP) serta user mengisi formulir yang kemudian dienkripsi sebelum dikirim ke database.

Website ini dibangun menggunakan framework Flask sebagai backend, serta HTML, CSS, dan JavaScript untuk frontend. Seluruh data yang disimpan dalam database telah melalui proses validasi dan enkripsi menggunakan algoritma RSA/AES. Admin harus melewati verifikasi kode OTP dari *Google Authenticator* untuk bisa mengakses *dashboard*.

3.2 Struktur Folder dan File Proyek

Tabel 3.1 Struktur Folder dan File Proyek

	KEMJAR ├── __pycache__ ├── .venv ├── keys │ ├── private.pem │ └── public.pem ├── static │ ├── css │ ├── images │ └── js ├── templates ├── admin_credentials.py ├── app.py ├── appadmin.py ├── db.py ├── encryption_utils.py ├── encryption.log ├── generate_keys.py ├── README.md ├── verification_aes.py └── verification_rsa.py

Struktur direktori proyek disusun secara modular agar memudahkan pengembangan dan pengelolaan.

Berikut merupakan penjelasan struktur direktori proyek secara ringkas untuk menggambarkan organisasi file dan fungsinya dalam sistem:

1. /templates

   Berisi seluruh file HTML untuk antarmuka pengguna: login.html, form.html, admin_dashboard.html, verify_otp.html, laporan_list.html, dll.

2. /static

   Berisi file statis seperti:

   - /css: gaya tampilan.
   - /images: ikon atau ilustrasi.
   - /js: login-admin.js untuk validasi sisi client.

3. /keys

   Menyimpan public.pem dan private.pem, yaitu kunci RSA yang digunakan untuk proses enkripsi dan dekripsi.

4. File utama:

   - app.py: kontrol utama aplikasi user.
   - appadmin.py: kontrol utama aplikasi admin.
   - db.py: pengelolaan koneksi dan query ke database.
   - encryption_utils.py: fungsi enkripsi dan dekripsi.
   - verification_aes.py & verification_rsa.py: enkripsi dan dekripsi data.
   - admin_credentials.py: menyimpan info login admin terenkripsi.
   - generate_keys.py: menghasilkan public-private key RSA.
   - encryption.log: mencatat aktivitas enkripsi.
   - README.md: dokumentasi singkat proyek.

3.3 Fitur Keamanan yang Diimplementasikan

Untuk meningkatkan keamanan sistem pada lapisan aplikasi, beberapa fitur berikut diimplementasikan secara terintegrasi:

1. Login Admin dengan Verifikasi OTP (*Google Authenticator*)

   Setelah memasukkan username dan password yang benar, admin harus memverifikasi kode OTP yang bersifat dinamis (berubah setiap 30 detik) melalui aplikasi *Google Authenticator*. Hal ini menambah lapisan keamanan terhadap akses tidak sah.

2. Proteksi *Brute Force (Rate Limiting)*

   Jika terjadi 5 kali kesalahan login berturut-turut, sistem akan memblokir login selama 1 menit. Mekanisme ini bertujuan untuk mencegah *brute-force attack.*

3. Enkripsi Data

   Setiap field laporan seperti judul, isi, nama, dan kontak dienkripsi menggunakan algoritma AES dalam mode GCM (Galois/Counter Mode) untuk menjamin kerahasiaan dan integritas data. Kunci AES yang digunakan kemudian diubah ke format base64 dan dienkripsi menggunakan RSA dengan skema OAEP (Optimal Asymmetric Encryption Padding).

   Pendekatan hybrid encryption ini digunakan karena RSA tidak efisien untuk mengenkripsi data dalam jumlah besar. Oleh karena itu, enkripsi utama dilakukan dengan AES yang ringan dan cepat, sementara RSA digunakan untuk melindungi kunci AES. Data hanya dapat dibaca kembali setelah kunci AES berhasil didekripsi menggunakan RSA private key yang valid.

3.4 *Tools* dan *Library* yang Digunakan

Aplikasi ini memanfaatkan berbagai *library* dan *tools* Python, baik untuk membangun aplikasi web, pengamanan data, serta interaksi dengan database. Berikut ini adalah daftar *tools* dan *library* yang digunakan beserta fungsinya:

Tabel 3.2 Tools dan Library yang Digunakan

No	*Library atau Tools*	Fungsi
1	Flask	*Framework* web Python. Digunakan untuk *routing*, *session*, dan *render* HTML dengan Jinja2.
2	Flask-Limiter	Membatasi jumlah request (*rate limiting*), seperti login *attempt* per menit.
3	PyOTP	Menghasilkan dan memverifikasi kode OTP berbasis waktu untuk autentikasi dua faktor (2FA).
4	PyCryptodome	Digunakan untuk enkripsi dan dekripsi data menggunakan AES-GCM dan RSA-OAEP. Juga SHA-256 untuk derive key.
5	PyMySQL	Menghubungkan aplikasi Python ke *database* MySQL untuk menyimpan dan membaca data.
6	filetype	Mendeteksi tipe file (image, PDF, DOCX) dari file hasil dekripsi.
7	base64	*Encoding* dan *decoding* string terenkripsi dan kunci AES sebelum/selesai dienkripsi RSA.
8	hashlib	Digunakan untuk menghasilkan AES *key* 256-bit dari SECRET_KEY dengan SHA-256.

9	datetime	Mengelola waktu sesi login, aktivitas terakhir, dan pencatatan waktu laporan.
10	os & io	os: akses *file* & *environment*. io: membaca *file* sebagai *binary stream* untuk *preview/download.*
11	Jinja2	*Template engine* bawaan Flask, digunakan untuk merender halaman HTML dinamis.
12	Google Authenticator	Digunakan oleh admin (*client-side*) untuk menghasilkan OTP dari secret key yang dibuat di server.

3.5 Perancangan Database

Database yang digunakan dalam sistem ini bertujuan untuk menyimpan data laporan keluhan kesehatan dari pengguna. Sistem menggunakan MySQL sebagai database relasional dan diakses menggunakan library pymysql dari Python. Terdapat dua tabel utama dalam rancangan basis data, yaitu tabel laporan dan tabel status_kasus.

1. Tabel laporan

   Tabel ini menyimpan seluruh informasi laporan yang dikirimkan oleh pengguna melalui form pelaporan

   Struktur tabel laporan adalah sebagai berikut:

   - id (INT, AUTO_INCREMENT, PRIMARY KEY): ID laporan.
   - kategori (VARCHAR(100)): Kategori tujuan poli kesehatan yang dipilih oleh pengguna.
   - judul (TEXT): Judul keluhan kesehatan.

- isi (TEXT): Penjelasan lengkap mengenai gejala, keluhan, atau kondisi kesehatan.
- bukti (LONGBLOB): File pendukung seperti foto hasil lab, surat dokter, atau gambar kondisi fisik.
- tanggal (DATE): Tanggal kunjungan yang diinginkan.
- lokasi (VARCHAR(255)): Lokasi penyakit atau area tubuh yang dikeluhkan.
- nama (TEXT): Nama lengkap pasien.
- kontak (TEXT): Informasi kontak pasien seperti email atau nomor telepon.
- aes_key (TEXT): Kunci AES yang digunakan untuk enkripsi data.

2. Tabel status_kasus

   Tabel ini digunakan untuk mencatat status dari setiap laporan yang masuk.

   Struktur tabel status_kasus adalah sebagai berikut:

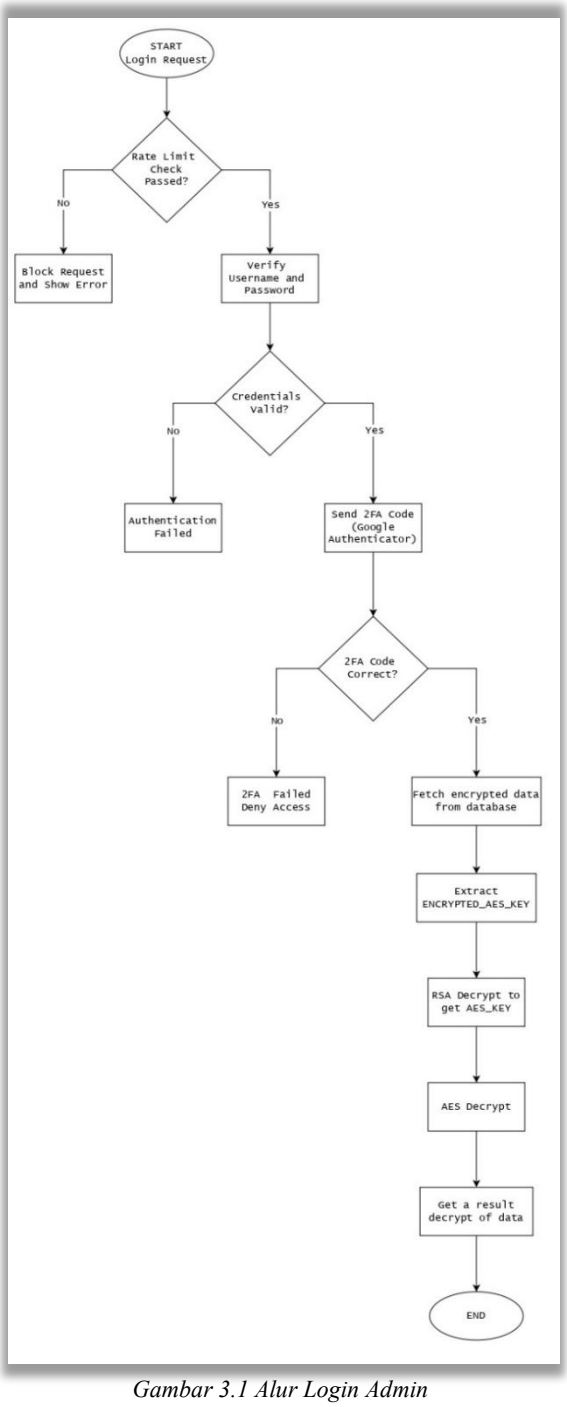
   - id (INT, AUTO_INCREMENT, PRIMARY KEY): ID status.
   - id_laporan (INT): ID laporan yang menjadi referensi.
   - status (ENUM('selesai', 'diproses', 'diajukan') DEFAULT 'diajukan'): Menyatakan pasien sedang berada dalam antrian.
   - updated_at (DATETIME DEFAULT CURRENT_TIMESTAMP): Waktu terakhir status diperbarui.

Tabel status_kasus memiliki relasi dengan tabel laporan melalui kolom id_laporan, dengan aturan penghapusan cascade apabila laporan dihapus.

3.6 Diagram Alur Sistem

Alur sistem dalam aplikasi ini dibagi menjadi dua bagian utama, yaitu alur login admin dan alur pengisian data oleh pengguna. Kedua alur tersebut telah dirancang dengan memperhatikan keamanan pada application layer, termasuk autentikasi dua faktor dan enkripsi data sebelum disimpan ke database.

1. Alur Login Admin

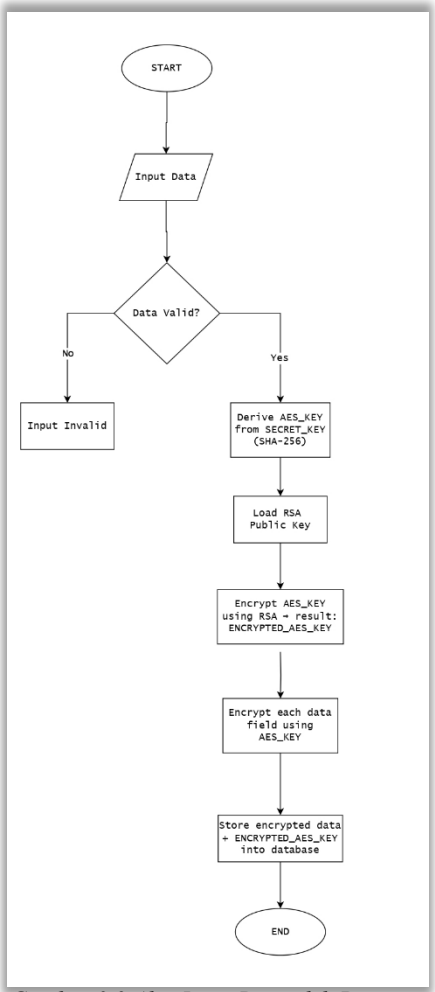


Gambar 3.1 Alur Login Admin

Proses login admin dimulai dari input username dan password. Sistem terlebih dahulu memverifikasi kredensial tersebut. Jika gagal, proses autentikasi ditolak. Apabila berhasil, sistem akan melakukan pengecekan *rate limit* untuk mencegah *brute-force*. Jika lolos, sistem mengirimkan kode 2FA (OTP) melalui *Google Authenticator.* OTP yang dimasukkan admin akan diverifikasi; jika valid, sistem mengambil data yang terenkripsi dari database.

Selanjutnya, sistem mengekstrak kunci AES terenkripsi (ENCRYPTED_AES_KEY), kemudian melakukan dekripsi RSA untuk mendapatkan AES_KEY. AES_KEY ini digunakan untuk mendekripsi data laporan yang dibutuhkan oleh admin. Jika seluruh proses berhasil, admin diberikan akses ke sistem.

2. Alur Input Data oleh Pengguna



Gambar 3.2 Alur Input Data oleh Pengguna

Proses diawali dari input data oleh pengguna, yang kemudian divalidasi oleh sistem. Jika data tidak valid, maka proses dihentikan dan ditandai sebagai “Input Invalid”.

Jika data valid, sistem akan membangkitkan AES_KEY dengan melakukan hashing terhadap SECRET_KEY menggunakan SHA-256. Setelah itu, RSA public key dimuat dan digunakan untuk mengenkripsi AES_KEY — menghasilkan ENCRYPTED_AES_KEY.

Setelah kunci simetris terenkripsi disiapkan, setiap field data seperti nama, isi laporan, dan kontak akan dienkripsi menggunakan AES_KEY. Seluruh data yang sudah dienkripsi beserta ENCRYPTED_AES_KEY kemudian disimpan ke dalam database.

BAB IV
IMPLEMENTASI

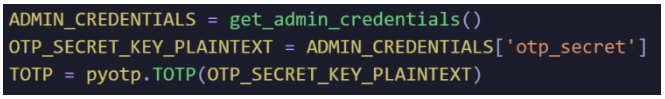
4.1 Implementasi Sistem

Sistem ini dibangun menggunakan framework Flask dan Python, dengan penekanan pada keamanan lapisan application layer. Terdapat tiga fitur keamanan utama yang diimplementasikan: autentikasi dua faktor (2FA), proteksi brute-force login, dan enkripsi data menggunakan RSA dan AES.

1. Autentikasi Dua Faktor (2FA)

   Autentikasi dua faktor diterapkan untuk proses login admin. Setelah username dan password diverifikasi, sistem meminta kode OTP yang dibangkitkan oleh aplikasi Google Authenticator. Verifikasi dilakukan menggunakan library pyotp seperti potongan berikut:

Gambar 4.1 Import Library pyotp



Gambar 4.2 Inisialisasi OTP dengan pyotp

Gambar 4.3 Verifikasi Kode OTP

Sebelum OTP diverifikasi, admin harus melewati tahapan login awal berikut:

```
if username_input == ADMIN_CREDENTIALS['username'] and \
           password_input == ADMIN_CREDENTIALS['password']:

            session.clear() # Hapus semua session lama untuk session fixation prevention
            session['otp_pending'] = True # Set flag untuk tahap OTP
            session['temp_username'] = username_input # Simpan username sementara

            flash('Username and password correct. Please enter your OTP.', 'success')
            return redirect(url_for('admin_verify_otp'))
```

Gambar 4.4 Proses Verifikasi Username dan Password

Jika OTP valid, *session* login dibuat dan aktivitas terakhir dicatat:

```
session['admin_logged_in'] = True # Set flag admin sudah Login
           session['username'] = ADMIN_CREDENTIALS['zsername'] # Simpan username admin ke session
           session['last_activity'] = datetime.utcnow().isoformat()
```

Gambar 4.5 Inisialisasi Session Login Setelah OTP Valid

2. Proteksi *Brute-Force* Login

Untuk mencegah brute-force attack, sistem menerapkan limitasi jumlah percobaan login melalui Flask-Limiter. Route /login dibatasi maksimal 5 percobaan per menit:

```
@limiter.limit("5 per minute")
```

Gambar 4.6 Proteksi Brute-Force

Selain itu, sistem juga mengatur session timeout secara eksplisit melalui konfigurasi Flask:

```
app.permanent_session_lifetime = timedelta(minutes=1)
```

Gambar 4.7 Konfigurasi Timeout Session

3. Enkripsi dan Dekripsi Data

Enkripsi data dilakukan pada saat pengguna mengirimkan laporan melalui form. Sistem menggunakan pendekatan hybrid encryption, yaitu:

- Data pengguna (judul, isi, lokasi, nama, kontak) dan file bukti dienkripsi menggunakan algoritma AES.

- AES key yang digunakan kemudian dienkripsi menggunakan algoritma RSA dan disimpan ke database.

Proses ini dilakukan agar data di database tidak dapat dibaca langsung tanpa kunci privat RSA.

Berikut adalah potongan kode enkripsi saat pengguna submit laporan:

Penurunan kunci AES (AES_KEY) dan enkripsi RSA terhadap kunci tersebut:

```
SECRET_KEY = 'kunci_rahasia_anda'
AES_KEY = derive_key(SECRET_KEY)
PUBLIC_KEY = load_public_key()

# RSA-encrypted AES key (hybrid encryption)
ENCRYPTED_AES_KEY = rsa_encrypt(base64.b64encode(AES_KEY).decode(), PUBLIC_KEY)
```

Gambar 4.8 Penurunan Kunci AES dan Enkripsi RSA

Enkripsi setiap field laporan keluhan kesehatan:

```
kategori = data['kategori']
judul = encrypt_data(data['judul'], AES_KEY)
isi = encrypt_data(data['isi'], AES_KEY)
tanggal = data['tanggal']
lokasi = encrypt_data(data['lokasi'], AES_KEY)
nama = encrypt_data(data['nama'], AES_KEY)
kontak = encrypt_data(data['kontak'], AES_KEY)
bukti = encrypt_bytes(file_bytes, AES_KEY)
```

Gambar 4.9 Enkripsi Setiap Field Laporan Keluhan

Semua data yang telah terenkripsi, termasuk ENCRYPTED_AES_KEY, disimpan ke dalam tabel laporan di database:

```
with connection.cursor() as cursor:
            query = '''
                INSERT INTO laporan
                (kategori, judul, isi, bukti, tanggal, lokasi, nama, kontak, anonim, aes_key)
                VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
            '''
            values = (
                kategori, judul, isi, bukti, tanggal, lokasi,
                nama, kontak, anonim, ENCRYPTED_AES_KEY
            )
            cursor.execute(query, values)
```

Gambar 4.10 enyimpanan Data Terenkripsi ke Database

Pengambilan dan dekripsi AES_KEY:

```
private_key = load_private_key()
```

Gambar 4.11 Pemanggilan Kunci Privat

```
aes_key_rsa_decrypted_bytes = rsa_decrypt(row['aes_key'], private_key)
aes_key = base64.b64decode(aes_key_rsa_decrypted_bytes)
```

Gambar 4.12 Dekripsi AES_KEY dari Bentuk Terenkripsi

Dekripsi setiap field laporan:

```
laporan_baru.append({
    'id': row['id'],
    'tanggal': row['tanggal'],
    'kategori': row['kategori'],
    'judul': decrypt_data(row['judul'], aes_key),
    'isi': decrypt_data(row['isi'], aes_key),
    'lokasi': decrypt_data(row['lokasi'], aes_key) if row['lokasi'] else '',
    'nama': decrypt_data(row['nama'], aes_key) if row['nama'] else '',
    'kontak': decrypt_data(row['kontak'], aes_key) if row['kontak'] else '',
    'status': row['status'],
    'updated_at': row['updated_at']
})
```

Gambar 4.13 Dekripsi Field Terenkripsi pada Laporan

File bukti juga didekripsi sebelum ditampilkan atau diunduh:

```
bukti_decrypted = decrypt_bytes(laporan_db['bukti'], aes_key) if laporan_db['bukti'] else b''
```

Gambar 4.14 Dekripsi File Bukti

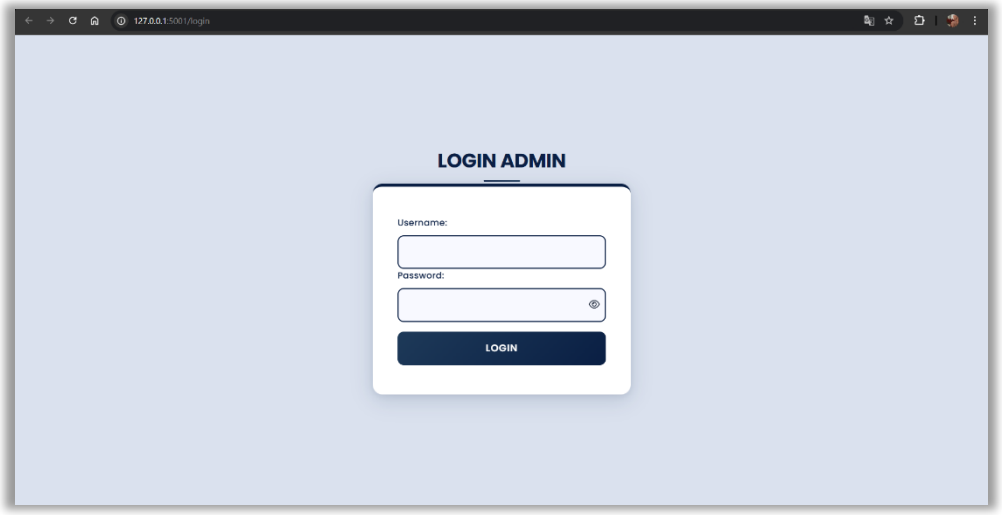
4.2 Pengujian Sistem

Pengujian dilakukan untuk mengevaluasi keberhasilan implementasi fitur keamanan yang telah dirancang pada sistem, khususnya pada lapisan application layer. Pengujian difokuskan pada beberapa aspek utama, yaitu autentikasi dua faktor (2FA), pembatasan login (rate limiting), enkripsi dan dekripsi data, serta pengelolaan akses terhadap informasi sensitif.

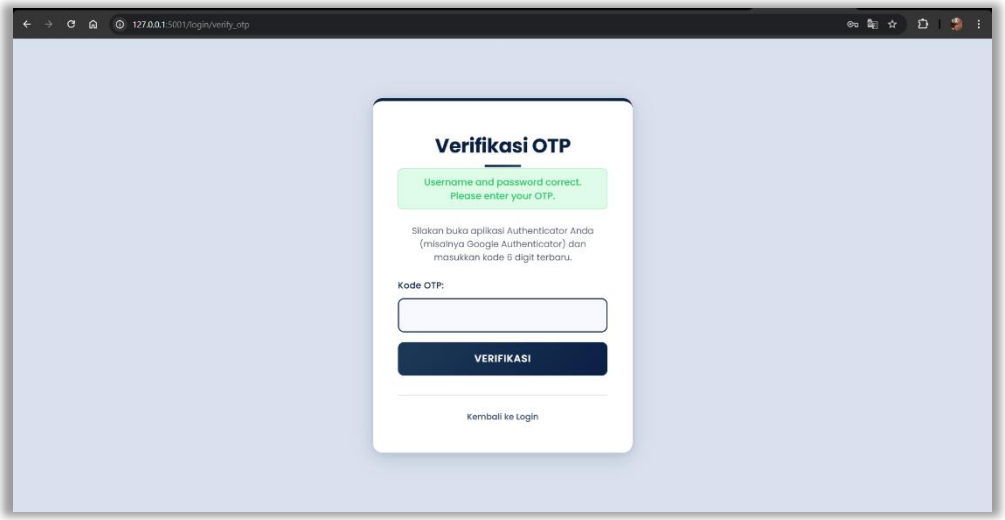
Setiap skenario diuji secara manual dengan memperhatikan tanggapan sistem terhadap masukan yang valid maupun tidak valid. Berikut ini adalah ringkasan skenario pengujian dan hasil yang diperoleh:

1. Autentikasi Admin dengan OTP

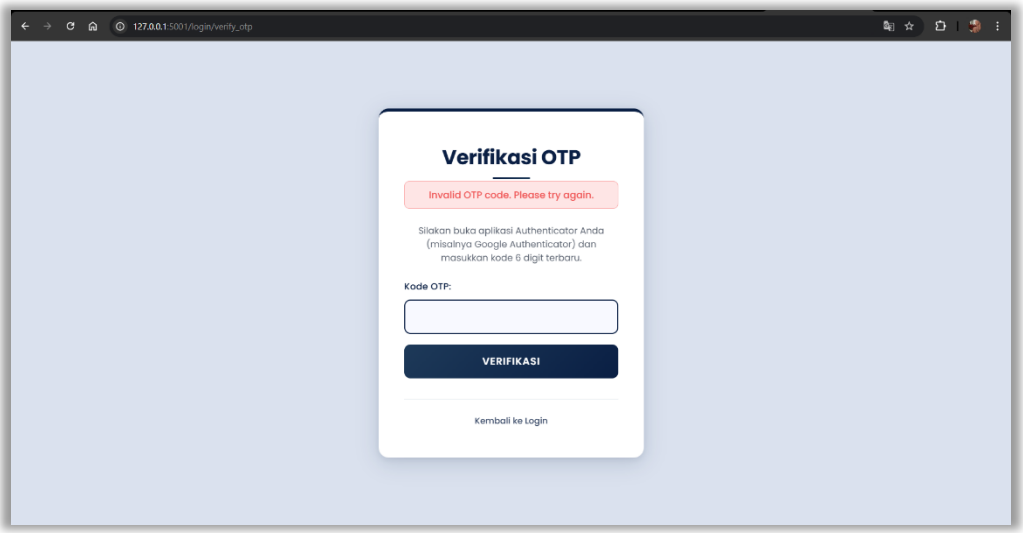
   Pengujian dilakukan terhadap proses login admin menggunakan *username* dan *password* yang benar, diikuti oleh kode OTP dari aplikasi *Google Authenticator*. Sistem berhasil memverifikasi OTP yang valid dan memberikan akses ke dashboard. Jika OTP yang dimasukkan salah atau tidak sesuai, sistem menolak akses dan menampilkan peringatan.



Gambar 4.15 Halaman Login Admin



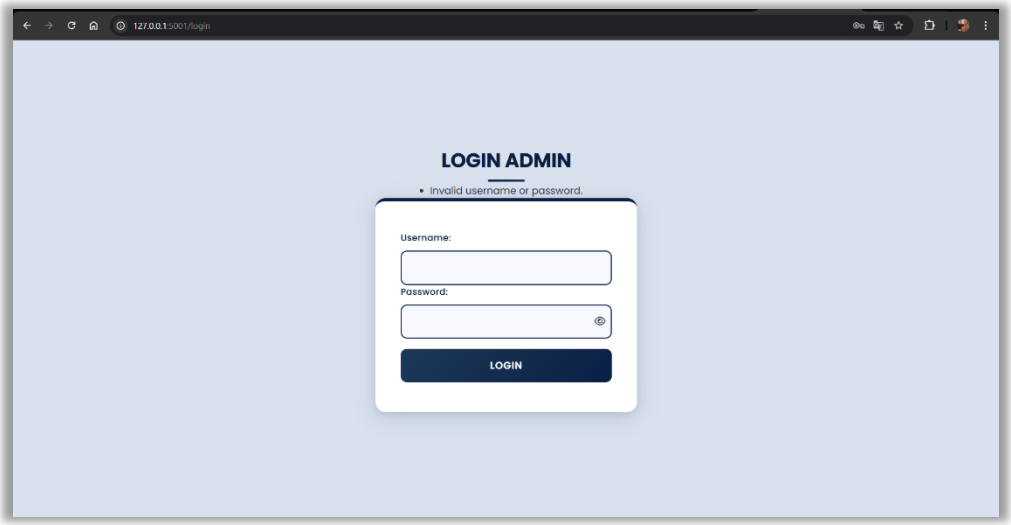
Gambar 4.16 Halaman input OTP



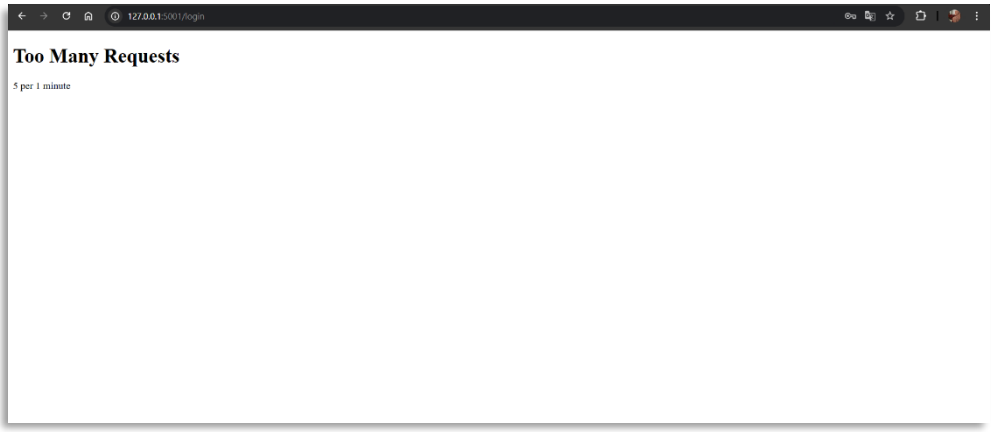
Gambar 4.17 Sistem menolak akses

2. Proteksi Terhadap Brute Force Login

   Admin mencoba melakukan login menggunakan password yang salah sebanyak lima kali berturut-turut. Setelah mencapai batas yang ditentukan, sistem menolak login dan memblokir percobaan akses selama satu menit. Setelah periode blokir berakhir, sistem mengizinkan login kembali dengan percobaan baru.



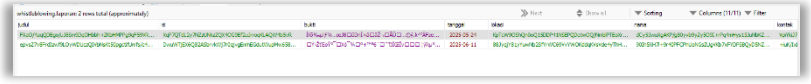
Gambar 4.18 Tampilan Login Gagal



Gambar 4.19 Akun diblokir setelah 5 kali gagal login

3. Penyimpanan Data dalam Bentuk Terenkripsi

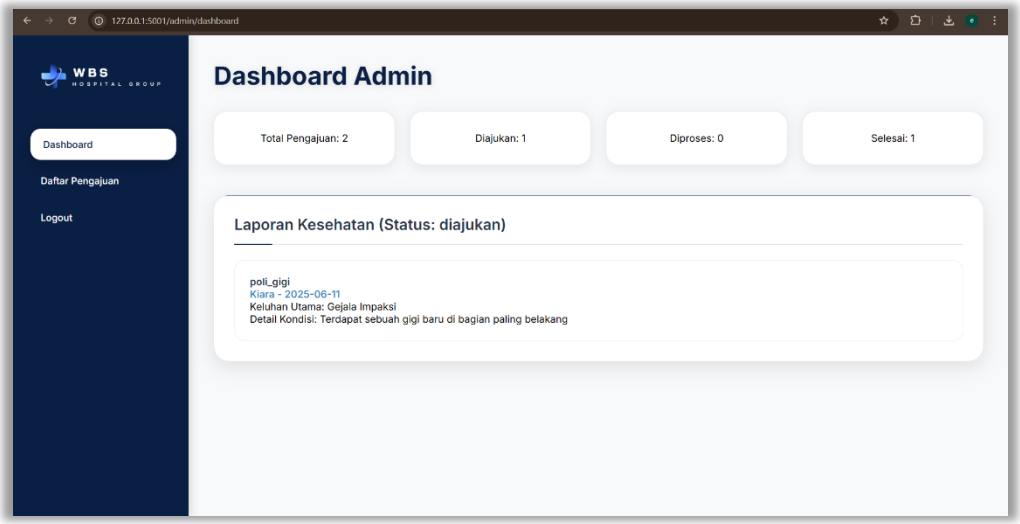
Data laporan yang dikirimkan melalui form pelaporan oleh pengguna disimpan di database dalam bentuk terenkripsi. Field seperti judul laporan, isi, nama, dan kontak tidak dapat dibaca secara langsung melalui database karena disimpan dalam bentuk ciphertext. Hal ini menunjukkan bahwa sistem berhasil menjaga kerahasiaan data pada saat penyimpanan.



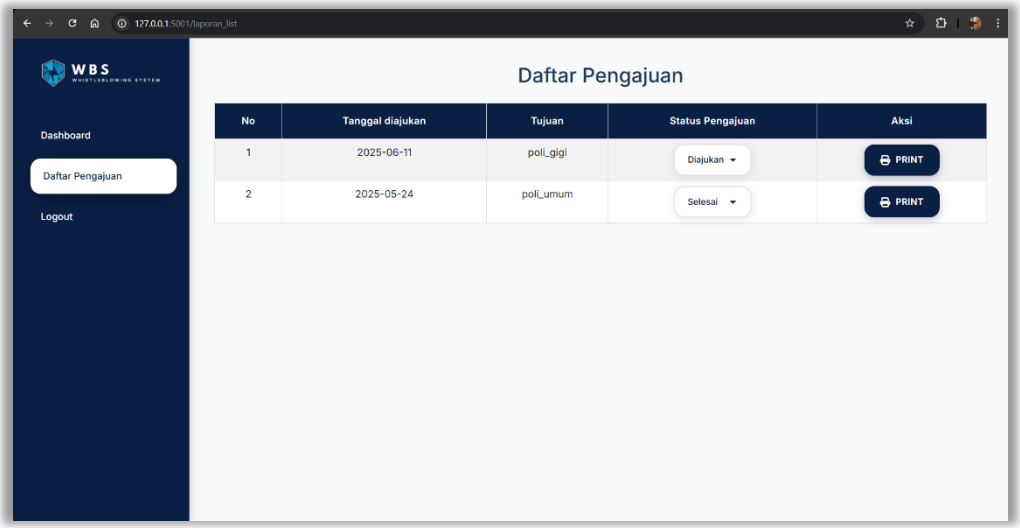
Gambar 4.20 Data Terenkripsi di Database

4. Dekripsi dan Penampilan Data di Dashboard Admin

Admin yang telah berhasil login dan melakukan verifikasi OTP dapat melihat daftar laporan pengguna dalam bentuk asli. Proses dekripsi dilakukan di sisi backend menggunakan private key RSA dan AES_KEY, sebelum ditampilkan dalam antarmuka dashboard. Seluruh field yang dienkripsi dapat ditampilkan kembali dengan benar.



Gambar 4.21 Dashboard Admin



Gambar 4.22 Daftar Pengajuan

Gambar 4.23 Data yang Berhasil Didekripsi

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Proyek pengembangan sistem ini berhasil menunjukkan penerapan keamanan yang kuat pada lapisan application layer. Melalui proses perancangan, implementasi, dan pengujian, sistem telah dilengkapi dengan sejumlah mekanisme pengamanan yang mampu melindungi data dan membatasi akses tidak sah.

Autentikasi dua faktor (*Two-Factor Authentication*/2FA) menggunakan aplikasi *Google Authenticator* terbukti efektif dalam memperkuat proses login admin. Dengan adanya tahapan verifikasi OTP setelah memasukkan *username* dan *password*, peluang akses oleh pihak yang tidak sah dapat ditekan secara signifikan. Selain itu, sistem juga dilengkapi dengan fitur pembatasan login (*rate limiting*), yang mencegah serangan *brute-force* dengan cara memblokir sementara akun setelah lima kali percobaan login yang gagal secara beruntun.

Dari sisi perlindungan data, sistem mengimplementasikan enkripsi *hybrid* menggunakan kombinasi RSA dan AES. Semua data laporan yang dikirimkan oleh pengguna, termasuk file bukti, disimpan dalam bentuk terenkripsi dan hanya dapat diakses kembali melalui proses dekripsi yang memerlukan kunci privat. Hal ini membuktikan bahwa kerahasiaan dan integritas data tetap terjaga meskipun terjadi potensi akses langsung terhadap database.

Secara keseluruhan, sistem yang dibangun telah memenuhi tujuan utama proyek, yaitu menghadirkan aplikasi web dengan keamanan berbasis *application layer* yang tangguh, dapat diandalkan, dan berfungsi sesuai dengan yang dirancang.

5.2 Saran

Meskipun sistem telah berjalan dengan baik, masih terdapat beberapa hal yang dapat dikembangkan di masa mendatang untuk meningkatkan kualitas dan skalabilitasnya.

Pertama, sistem dapat dilengkapi dengan enkripsi tambahan untuk file bukti dengan mode enkripsi yang lebih aman seperti AES-GCM yang mendukung integritas data. Kedua, penambahan fitur audit log untuk mencatat seluruh aktivitas admin selama berada di dashboard dapat menjadi bagian penting dalam menjaga akuntabilitas sistem.

Selain itu, integrasi dengan layanan notifikasi email atau SMS juga dapat dipertimbangkan, khususnya untuk menginformasikan upaya login yang mencurigakan atau laporan masuk baru. Fitur reCAPTCHA juga dapat diterapkan pada halaman login untuk mencegah serangan otomatis oleh bot.

Akhirnya, pengembangan sistem agar dapat mendukung multi-admin dan penerapan kontrol akses berbasis peran (*Role-Based Access Control*) akan sangat bermanfaat untuk implementasi di lingkungan nyata yang lebih kompleks.