Nama : Lukmanul Hakim
NIM : 20230801228
Matkul : Keamanan Informasi
Dosen : Dr. Hani Dewi Ariessanti B, S.KOM, M.KOM
Link GitHub : https://github.com/Lukmanul77/UAS-KSI

Analisis Studi Kasus – Keamanan Data Gaji Karyawan

1. Scope Kasus & Data Sensitif

- **Objek:** data gaji dan informasi personal (pegawai, jabatan, gaji, dsb).
- **Risiko:** data gaji adalah **High Sensitivity Level**—terbuka untuk pengguna yang tidak berhak dapat menimbulkan masalah hukum, reputasi, dan privasi.

2. Threat Modeling

Actor	Threats	Impact
Authenticated Attacker	- SQL Injection - IDOR (Insecure Direct Object Reference) - Escape ke aplikasi admin	Data salary disclosure, privilege escalation
External Attacker	- Brute-force login - CSRF/XSRF - XSS di form input user/admin - Directory Traversal	Data leak, defacement, unauthorized access
Insider Threat	- Bypassing self-service restrictions - Export data ke file (cheat)	Unauthorized sharing of sensitive data

3. Analisa Kode & Vulnerability Points

Berdasarkan struktur Laravel + Filament:

- **Authentication & Authorization**

- Pastikan guard default Laravel hanya menggunakan BCrypt. Cek .env dan config/auth.php.

- Filament memiliki role/permission. Pastikan kebijakan akses menggunakan Gates/Policies di semua operasi CRUD gaji.

- **Input Validation & SQL Injection**

Form menggunakan Filament Resource. Namun tetap verifikasi:

```
php
protected function getFormSchema(): array {
  TextInput::make('salary')->numeric()->required();
}
```

- Pastikan tidak ada raw query seperti DB::table('gaji')->whereRaw(...).

- **Mass Assignment**

- Model Salary harus mendefinisikan $fillable, bukan $guarded = [], untuk mencegah pengisian field yang tidak seharusnya.

- **IDOR**

Endpoints berbasis ID (misalnya /admin/salaries/{id}/edit) harus mengecek:

```
php
public function editableBy(User $user) {
  return $user->isAdmin() || $user->id === $this->owner_id;
}
```

- **CSRF**

Filament sudah menyertakan token CSRF, tetapi jika ada form non-Filament, pastikan:

```
blade
<form method="POST">@csrf ...</form>
```

- **XSS**

Blade secara default escape output. Namun, bila menggunakan format HTML:

```
 blade
{!! $salary->notes !!}
```

- harus sanitize input terlebih dahulu, misalnya menggunakan Purify::clean() agar mencegah script injection.

- **Rate Limiting**

Login endpoint harus dibatasi:

```
 php
'throttle:5,1', // max 5 percobaan per menit
```

4. Rekomendasi Mitigasi & Hardening

A. Authentication & Brute-force

- Aktifkan Laravel Throttle & Filament Login Limit (5 attempts per 1 minute).
- Terapkan account lock setelah 5 kali gagal berturut-turut dan notifikasi via email.

B. Least Privilege (Authorization)

- Definisikan Gate viewSalary, editSalary, deleteSalary di AuthServiceProvider.

Terapkan pada Filament Resource:

```
 php
public static function canEdit(Model $record, User $user): bool {
  return $user->hasRole('admin') || $user->id === $record->user_id;
}
```

C. Input Validation & Sanitasi

- Pastikan salary numeric & dalam rentang wajar:

  ```
   php
  ->rule('numeric|min:100000|max:1000000000')
  ```

- Sanitasi field notes:

  ```
   php
  $input = Purify::clean($request->input('notes'));
  ```

D. SQL Injection / Mass Assignment

- Gunakan Eloquent/Query Builder standard, hindari whereRaw.
- Model Salary:

```
 php
protected $fillable = ['user_id', 'salary', 'notes'];
```

E. CSRF, XSS, CSP

- Pastikan <meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self';"> diset di resources/views/layouts/app.blade.php.
- Semua request memakai token CSRF.
- Output di Blade escaped atau disanitasi.

F. Secure Storage & Data in Transit

- .env file: pastikan APP_ENV=production, APP_DEBUG=false.
- Gunakan HTTPS (TLS), pakai SSL di server.
- Enkripsi gaji (opsional): Laravel memiliki encrypt() untuk field sensitif.

G. Audit & Logging

- Implementasi Activity Log (misalnya package spatie/laravel-activitylog) untuk event: create/update/delete salary.
- Notifikasi via email untuk perubahan gaji signifikan.

5. Checklist Vulnerability Assessment

Risiko	Ada/Kekurangan	Mitigasi Disarankan
Authentication brute-force	? (belum terlihat throttle)	Laravel throttle
Authorization (IDOR)	Kemungkinan ada akses ke gaji user lain	Gunakan Gates + Policies
Input sanitasi	Validasi numeric ada; sanitasi text?	Tambahkan Purify
CSRF	Filament inklusif, tapi jika ada custom form	Pastikan @csrf
XSS / Output escape	Filament Blade auto escape	Hindari {!! !!}

SQL Injection	Jika raw query: risk high	Gunakan Eloquent
Mass assignment	Belum jelas definisi $fillable	Terapkan $fillable
Secure config & transport	.env, HTTPS?	Atur TLS, debug=false
Logging / Audit trail	Belum ada log	Pasang activitylog
CSP header	Tidak terlihat	Tambah meta CSP

6. Kesimpulan & Langkah Lanjutan

Anda sudah membangun pondasi aplikasi dengan Laravel & Filament yang kuat. Namun, untuk memenuhi standar keamanan data gaji karyawan, Anda perlu memperkuat aspek:

1. **Authentication & Rate Limiting**
2. **Authorization granular (Gates/Policies)**
3. **Input Sanitasi & Output Escape**
4. **Secure Config/Transport**
5. **Audit log & Versioning**

Setelah menerapkan mitigasi tersebut, rekomendasi selanjutnya:

- Jalankan **penetration testing** manual/otomatis (OWASP ZAP, Snyk).
- Adakan **code review** khusus keamanan.
- Lakukan **security training** bagi developer agar aware terhadap praktik aman.