UJIAN AKHIR SEMESTER

KEAMANAN INFORMASI KJ003

Nama : Arrauuf Reza Firmansyah
NIM : 20210801188

Soal :

1. Buatlah analisisnya dengan kasus yang anda tentuka sendiri
2. Buatlah aplikasinya
3. Lakukan Vulnerability Assesment terhadap aplikasi yang dibikin
4. Lakukan Pengamanan data – datanya sesuai dengan kasus yang anda tentukan sendiri

Jawaban :

1. Di era digital saat ini, banyak perusahaan atau startup menggunakan jasa freelancer untuk menunjang kebutuhan proyek jangka pendek maupun pengembangan produk. Namun, seiring bertambahnya jumlah freelancer dan kompleksitas proyek, perusahaan sering menghadapi masalah dalam pengelolaan data freelancer, distribusi proyek, serta monitoring progres pekerjaan. Tanpa sistem terpusat, informasi terkait freelancer, proyek yang mereka tangani, dan perkembangan pekerjaan menjadi sulit dipantau dan berpotensi menyebabkan keterlambatan atau miskomunikasi.

   Oleh karena itu, diperlukan sebuah sistem manajemen proyek berbasis web yang dapat digunakan untuk mengelola data freelancer, mendistribusikan proyek kepada freelancer sesuai keahlian, serta memantau perkembangan proyek secara real-time. Sistem ini juga memungkinkan adanya pembagian hak akses, di mana administrator,, dan freelancer memiliki level akses yang berbeda sesuai peran mereka. Dengan demikian, perusahaan dapat meningkatkan efisiensi manajemen proyek, mempercepat proses pengambilan keputusan, serta menjaga transparansi dan akuntabilitas pekerjaan.

2. Link github : https://github.com/rauf-ezaa/UAS-KI-Arrauuf_Reza_Firmansyah

3. Setelah website dilakukan pengujian analaisis assessment dengna menggunakan tools ptt website_scanner maka di dapatkan hasil pengujian dan analisis nya yaitu

- Missing security Headers

  Beberapa Http Security headers tidak ditemukan di response aplikasi, yakni :

  - **Content-Security-Policy (CSP)**

  ➔ Berpotensi membuat aplikasi rentan terhadap serangan **Cross-Site Scripting**
  Rekomendasi: Tambahkan header `Content-Security-Policy` pada response server

  - **X-Content-Type-Options**

  ➔ Tanpa ini, browser dapat melakukan MIME sniffing yang berisiko terhadap content-type spoofing
  Rekomendasi: Tambahkan `X-Content-Type-Options: nosniff`

  - **Strict-Transport-Security (HSTS)**

  ➔ Tanpa HSTS, pengguna dapat terkoneksi via HTTP yang rentan terhadap man-in-the-middle
  Rekomendasi: Terapkan `Strict-Transport-Security` pada semua HTTPS response

  - **Referrer-Policy**

  ➔ Untuk mengontrol informasi referer yang dikirim ke server lain
  Rekomendasi: Gunakan header `Referrer-Policy: no-referrer` atau `strict-origin-when-cross-origin`

Server Software Disclosure

- Informasi framework, library, dan software yang digunakan (Laravel, Filament, Livewire, Alpine.js, Cloudflare) terdeteksi oleh scanner
  ➔ **Risiko:** Dapat memberikan petunjuk bagi attacker untuk eksploitasi spesifik
  Rekomendasi: Minimalkan informasi yang terexpose melalui header atau meta

robots.txt Terbuka

- File robots.txt ditemukan dan dapat diakses publik
  ➔ Jika ada direktori sensitif yang diblokir crawler, justru bisa diketahui attacker
  Rekomendasi: Review isi robots.txt dan hindari menulis direktori sensitif

Login Interface Ditemukan

- Halaman login /admin/login terdeteksi
  ➔ Risiko brute force atau credential stuffing
  Rekomendasi:
  - Gunakan rate limiting
  - CAPTCHA
  - Audit credential secara berkala

Security.txt Tidak Ditemukan

- Tidak ada file .well-known/security.txt
  ➔ Ini bukan celah, tapi best practice untuk publikasi jalur kontak keamanan
  Rekomendasi: Tambahkan security.txt untuk kontak vulnerability disclosure

Kesimpulan

- Tidak ditemukan **kerentanan berisiko tinggi** seperti SQL Injection, XSS aktif, CSRF, Auth Bypass, atau Server Misconfiguration

- Temuan bersifat **informasi dan best practice**, yang disarankan untuk segera ditindaklanjuti demi memperkuat keamanan aplikasi

4. Pada Bagian keamanan sistem informasi website yang dibuat, pengamanan data dilakukan dengan cara memberikan token pada saat user akan HIT request API data project. Pengamanan bekerja dengan cara menyelipkan api_token pada request API lalu ditaruh api_token tersebut pada authorization bearer token.

   Token tersebut digunakan untuk mendapatkan data yang akan dirimkan kembali oleh server. data yang dikirim dari server berbentuk response JSON yang telah terenkripsi. Untuk membuka data enkripsi tersebut digunakan key decrypt yang akan sama seperti api_token yang dikirimkan server kepada user.

   Kesimpulannya, pengamanan data pada sistem informasi ini mencakup middleware dahulu dimana ketika user akan mengambil data project maka user akan diminta untuk mengirim api_token sebagai bukti user telah authenticated. Setelah itu, user akan dikirimkan data project dalam bentuk data yang telah ter enkripsi. Data tersebut dapat dibuka menjadi plain text biasa menggunakan kunci dekripsi yakni api_token