JIKA (Jurnal of Informatics) Universitas Muhammadiyah Tangerang *P ISSN : 2549-0710*
Vol 8, No 3, July 2024, pp 281- 287 *E ISSN : 2722-2713*
DOI : 10.31000/jika.v8i3.11198

ANALISIS DAN IMPLEMENTASI FITUR KEAMANAN APLIKASI PADA FRAMEWORK LARAVEL

Syepry Maulana Husain[1)], Lukman Azhari[2)], Muhammad Lutfhi Aksani[3)] , Suwanda Aditya Saputra[4)]

[1,2,3] Teknik Informatika Fakultas Teknik, Universitas Muhammadiyah Tangerang
[4] Teknik Informatika Fakultas Teknik, Universitas Bina Nusantara
Jl. Perintis Kemerdekaan 1/33 Cikokol Kota Tangerang

Co Responden: shevrie18@gmail.com

Article history
Received 07 Apr 2024
Revised 15 Jun 2024
Accepted 26 Jun 2024
Available online 31 Jul 2024

Keywords
Laravel,
Security,
Web,
Application,
Framework

Abstract

In an increasingly complex digital age, web application security is one of the most important aspects to consider. The Laravel framework has become a top choice in web application development thanks to the robust security features it provides. However, a deep understanding of Laravel's security features and practical implementation in the context of real web application development is still a challenge for many developers. This article provides an in-depth analysis of the security features provided by Laravel and applies them in a web application development case study. We selected several key security features, including protection against SQL injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). The analysis shows that by properly implementing Laravel's security features, web applications can be effectively protected against many common security attacks. However, challenges faced in the implementation of these security features were also revealed, including the complexity of configuration and customization required to integrate security features in complex web applications. Thus, this research provides a better understanding of web application security practices using the Laravel framework.

Riwayat
Diterima 07 Aprl 2024
Revisi 15 Jun 2024
Disetujui 26 Jun 2024
Terbit online 31 Jul 2024

Kata Kunci
Laravel,
Keamanan,
Web,
Aplikasi,
Framework

Abstrak

Dalam era digital yang semakin kompleks, keamanan aplikasi web menjadi salah satu aspek yang paling penting untuk dipertimbangkan. *Framework* Laravel telah menjadi pilihan utama dalam pengembangan aplikasi web berkat fitur-fitur keamanan yang kuat yang disediakannya. Namun demikian, pemahaman yang mendalam tentang fitur-fitur keamanan Laravel dan implementasi praktis dalam konteks pengembangan aplikasi web yang nyata masih menjadi tantangan bagi banyak pengembang. analisis mendalam terhadap fitur keamanan yang disediakan oleh Laravel dan menerapkannya dalam studi kasus pengembangan aplikasi web. Kami memilih beberapa fitur keamanan utama, termasuk proteksi terhadap *SQL injection, Cross-Site Scripting (XSS*), dan *Cross-Site Request Forgery (CSRF)*. Hasil analisis menunjukkan bahwa dengan menerapkan fitur keamanan Laravel secara tepat, aplikasi web dapat dilindungi secara efektif dari berbagai serangan keamanan yang umum. Namun, tantangan yang dihadapi dalam implementasi fitur keamanan ini juga terungkap, termasuk kompleksitas konfigurasi dan penyesuaian yang diperlukan untuk mengintegrasikan fitur-fitur keamanan dalam aplikasi web yang kompleks. Dengan demikian, penelitian ini memberikan pemahaman yang lebih baik tentang praktik keamanan aplikasi web menggunakan *framework* Laravel.

PENDAHULUAN

Aplikasi web telah menjadi infrastruktur yang vital dalam kehidupan sehari-hari, menyediakan platform untuk berbagai aktivitas mulai dari transaksi keuangan hingga pertukaran informasi penting. Namun, seiring dengan kemudahan akses dan fungsionalitas yang diberikan, aplikasi web juga menjadi target yang menarik bagi para penyerang cyber. Ancaman seperti serangan peretasan, pencurian data, dan pelanggaran privasi telah menjadi kekhawatiran yang semakin meningkat dalam pengembangan dan penggunaan aplikasi web. Oleh karena itu,

memastikan keamanan aplikasi web telah menjadi prioritas utama bagi pengembang dan organisasi yang bergerak dalam lingkungan digital.

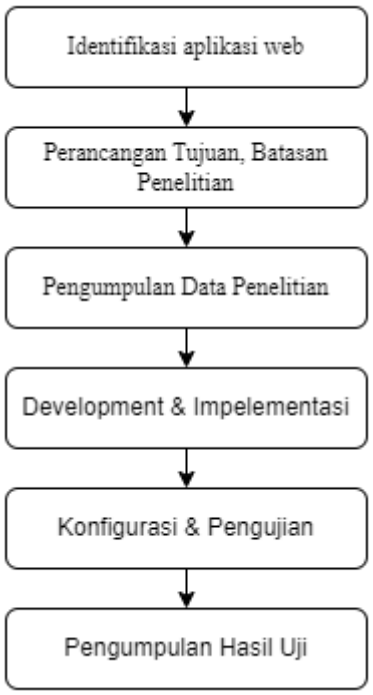
Salah satu langkah yang penting dalam memastikan keamanan aplikasi web adalah dengan menggunakan framework pengembangan yang dapat menyediakan fitur keamanan yang kuat. Dalam konteks ini, Laravel muncul sebagai salah satu framework PHP yang paling populer dan kuat dalam pengembangan aplikasi web. Dikenal karena sintaks yang jelas, struktur yang terorganisir, serta kumpulan fitur bawaan yang lengkap, Laravel tidak hanya mempercepat proses pengembangan aplikasi web tetapi juga menyediakan alat yang efektif untuk melindungi aplikasi dari berbagai serangan keamanan.

Namun demikian, meskipun Laravel menyediakan sejumlah fitur keamanan bawaan, pengembang masih perlu memahami dan menerapkan praktik keamanan terbaik untuk memastikan bahwa aplikasi yang mereka bangun memiliki tingkat keamanan yang optimal. Analisis mendalam terhadap fitur keamanan yang disediakan oleh Laravel serta implementasi praktis dalam pengembangan aplikasi web yang nyata menjadi langkah penting dalam memperkuat pertahanan keamanan aplikasi. Studi kasus yang mendemonstrasikan penerapan fitur keamanan Laravel dalam situasi yang real dapat memberikan wawasan yang berharga bagi pengembang dalam menghadapi tantangan keamanan yang kompleks.

Dengan menggabungkan analisis fitur keamanan Laravel dan implementasi dalam studi kasus pengembangan aplikasi web, penelitian ini bertujuan untuk memberikan pemahaman yang komprehensif tentang cara efektif menggunakan fitur keamanan Laravel untuk menciptakan aplikasi web yang aman dan andal. Diharapkan bahwa temuan dari penelitian ini akan memberikan panduan praktis bagi pengembang dalam upaya mereka untuk membangun aplikasi web yang tahan terhadap serangan peretasan dan pelanggaran keamanan, serta memperkuat kepercayaan pengguna dalam lingkungan digital yang semakin kompleks ini.

METODE PENELITIAN

Penelitian ini menggunakan model penelitian studi kasus (*case study research model*) yang memberikan kesempatan bagi peneliti untuk memahami dengan lebih baik konteks pengembangan aplikasi web, tantangan yang dihadapi oleh pengembang, dan faktor-faktor yang mempengaruhi keputusan mereka dalam mengimplementasikan fitur keamanan. Ini dapat membantu peneliti untuk menghasilkan rekomendasi yang lebih relevan dan bermanfaat bagi praktisi serta dapat memberikan wawasan yang mendalam dan relevan bagi pengembangan aplikasi web yang lebih aman dan andal.



Gambar 1. Tahapan Penelitian

Berikut ini tahapan dalam penelitian yang dilakukan:

1. Identifikasi aplikasi web yang akan menjadi studi kasus.
2. Perancanaan tujuan penelitian, batasan studi, seperti lingkup aplikasi, fitur keamanan yang akan diimplementasikan, dan jenis serangan yang akan diuji.
3. Pengumpulan data awal tentang aplikasi web yang akan diteliti, termasuk dokumentasi pengembangan, spesifikasi keamanan, dan riwayat serangan

4. Implementasi fitur keamanan Laravel yang telah ditentukan ke dalam aplikasi web.
5. Konfigurasi dan Pengujian fitur keamanan.
6. Pengumpulan data selama proses implementasi fitur keamanan, termasuk log pengujian, tangkapan layar, dan catatan implementasi.

HASIL DAN PEMBAHASAN

Hasil analisis keamanan web tanpa fitur keamanan pada framework Laravel berdasarkan serangan atau ancaman yang sering terjadi pada aplikasi web sebagai berikut:

a. **Proteksi terhadap SQL Injection:**
Implementasi proteksi terhadap SQL Injection dalam aplikasi web tanpa menggunakan fitur keamanan dari Laravel, kami menemukan bahwa tidak ada langkah-langkah khusus yang diambil untuk mencegah serangan SQL Injection. Parameter binding tidak digunakan secara otomatis, meninggalkan aplikasi rentan terhadap jenis serangan ini.

b. **Proteksi terhadap Cross-Site Scripting (XSS)**
Ditemukan bahwa tidak ada fitur otomatis dalam framework yang menghindari serangan XSS. Penggunaan output data dalam HTML tidak diawasi secara ketat, sehingga memungkinkan potensi serangan XSS jika input tidak divalidasi dengan benar.

c. **Proteksi terhadap Cross-Site Request Forgery (CSRF)**
Implementasi fitur CSRF Protection dan menemukan bahwa tidak ada langkah-langkah yang diambil untuk melindungi aplikasi dari serangan CSRF. Token CSRF tidak disertakan dalam form, dan tidak ada verifikasi CSRF yang dilakukan oleh aplikasi.

Aplikasi web tanpa fitur keamanan dari framework Laravel rentan terhadap serangan keamanan yang umum, seperti SQL Injection, Cross-Site Scripting (XSS), dan Cross-Site Request Forgery (CSRF). Kekurangan ini meninggalkan aplikasi web rentan terhadap ancaman keamanan, dan langkah-langkah perlindungan yang sesuai harus diimplementasikan untuk meningkatkan keamanan aplikasi.

Untuk menerapkan fitur keamanan web pada Laravel melibatkan sejumlah langkah yang perlu diambil untuk memastikan bahwa aplikasi web yang dikembangkan menggunakan framework ini memiliki tingkat keamanan yang tinggi. Berikut adalah beberapa cara untuk menerapkan fitur keamanan web pada Laravel:

1. **Menggunakan Middleware untuk Proteksi Route**
Middleware dapat digunakan untuk mengontrol akses ke route tertentu dalam aplikasi. Kita dapat membuat middleware khusus untuk memverifikasi apakah pengguna telah melakukan otentikasi sebelum diizinkan mengakses route tertentu. Middleware juga dapat digunakan untuk menerapkan proteksi terhadap *Cross-Site Request Forgery* (CSRF) dan XSS.
Contoh:

```
// Atur kebijakan CSP dalam
middleware protected $middleware =
[ //

...
\Illuminate\Http\Middleware\AddCont
entSecurityPolicy::class, ];
```

2. **Otentikasi dan Otorisasi**
Laravel menyediakan sistem otentikasi yang kuat yang dapat diimplementasikan dengan mudah. Kita dapat menggunakan fitur bawaan seperti Laravel Passport untuk otentikasi API atau menggunakan fitur otentikasi bawaan Laravel untuk otentikasi pengguna. Selain itu, Laravel juga menyediakan sistem otorisasi yang memungkinkan Anda untuk menentukan peran dan izin pengguna dalam aplikasi. Kita dapat mengatur akses pengguna ke fitur-fitur tertentu berdasarkan peran mereka.
Berikut langkah-langkah untuk mengkonfigurasi sistem otentikasi dan otorisasi di Laravel:
a. **Instalasi Laravel Authentication Scaffolding**
Laravel menyediakan perintah artisan untuk menginstal scaffolding

JIKA (Jurnal of Informatics) Universitas Muhammadiyah Tangerang *P ISSN : 2549-0710*
Vol 8, No 3, July 2024, pp 281- 287 *E ISSN : 2722-2713*
DOI : 10.31000/jika.v8i3.11198

otentikasi dengan mudah. Anda dapat menggunakan perintah berikut untuk menginstalnya:

```
php artisan make:auth
```

Perintah ini akan menghasilkan tampilan, rute, dan kontroler yang diperlukan untuk otentikasi.

b. **Konfigurasi Guard dan Provider**

Laravel menggunakan konsep "guard" untuk mengatur bagaimana pengguna diautentikasi saat mengakses aplikasi. Anda dapat mengkonfigurasi guard dan provider otentikasi dalam file **config/auth.php**. Sesuaikan opsi seperti driver (misalnya, **session** atau **token**), model pengguna yang digunakan, dan tabel database yang digunakan untuk menyimpan informasi otentikasi.

c. **Menyesuaikan Model Pengguna**

Pastikan model pengguna mengimplementasikan antarmuka **Illuminate\Contracts\Auth\Authenticatable**. Ini memungkinkan Laravel untuk mengelola otentikasi pengguna.

d. **Menerapkan Otorisasi**

Laravel menyediakan middleware **auth** yang dapat digunakan untuk memastikan bahwa pengguna telah diautentikasi sebelum mengakses rute tertentu. Anda dapat menetapkannya ke dalam rute atau grup rute yang memerlukan otentikasi.
Dapat juga digunakan metode **authorize()** yang disediakan oleh kontroler dasbor yang dihasilkan oleh scaffolding otentikasi untuk memeriksa izin pengguna sebelum menampilkan data.

e. **Definisikan Role dan Permissions (Opsional)**

Jika diperlukan sistem otorisasi yang lebih kompleks dengan peran dan izin, Gunakan paket pihak ketiga seperti *Spatie Laravel Permission* atau *Laravel Bouncer*.
Dengan menggunakan paket ini, kita dapat dengan mudah mendefinisikan peran dan izin, serta menerapkan logika otorisasi yang kompleks dalam aplikasi.

3. **Validasi Input**

Penting untuk selalu memvalidasi input yang diterima dari pengguna sebelum menyimpannya ke dalam database. Laravel menyediakan fasilitas validasi yang kuat melalui fitur Validasi. Kita dapat menggunakan aturan validasi bawaan seperti *required*, *numeric*, *email*, dan lain-lain, serta membuat aturan validasi kustom sesuai kebutuhan aplikasi.

```
$request->validate([
    'username' => 'required|string',
]);
```

4. **Proteksi Terhadap SQL Injection**

Laravel menggunakan *parameter binding* untuk mengamankan *query database* dari serangan SQL injection secara otomatis. Pastikan gunakan parameter binding saat menjalankan *query database* untuk memastikan keamanan aplikasi.

Untuk mencegah SQL Injection dalam aplikasi web Laravel, dapat diikuti praktik-praktik keamanan berikut:

a. Query Builder / Eloquent ORM: Gunakan Query Builder atau Eloquent ORM untuk membangun kueri database. Kedua metode ini secara otomatis melakukan pengikatan parameter, sehingga mencegah SQL Injection.
Contoh penggunaan Query Builder:

```
$users = DB::table('users')->where('username', $username)->get();
```

Contoh penggunaan Eloquent ORM:

```
$user = User::where('username', $username)->first();
```

b. Parameter Binding

JIKA (Jurnal of Informatics) Universitas Muhammadiyah Tangerang *P ISSN : 2549-0710*
Vol 8, No 3, July 2024, pp 281- 287 *E ISSN : 2722-2713*
DOI : 10.31000/jika.v8i3.11198

Jika diperlukan menulis kueri SQL langsung, gunakan parameter binding untuk mengikat nilai parameter ke dalam kueri. Laravel akan melakukan penyisipan parameter secara aman, menghindari risiko SQL Injection.
Contoh penggunaan parameter binding:

```
$users = DB::select("SELECT * FROM users WHERE username = ?", [$username]);
```

c. Validation

Validasi input pengguna sebelum menjalankan kueri database. Pastikan input tidak mengandung karakter khusus yang dapat dieksploitasi dalam kueri SQL.
Contoh validasi input:

```
$request->validate([
    'username' => 'required|string',
]);
```

d. Penggunaan Eloquent Accessors & Mutators:

Gunakan *Accessors* untuk menampilkan data dengan aman dari model *Eloquent*, dan Gunakan *Mutators* untuk menyimpan data ke *database* dengan aman.
Contoh penggunaan *Accessors* & *Mutators*:

```
// Accessor
public function getUsernameAttribute($value)
{
    return htmlspecialchars($value);
}

// Mutator
public function setUsernameAttribute($value)
{
    $this->attributes['username'] = htmlspecialchars($value);
}
```

e. Penggunaan Prepared Statements

Jika ingin menggunakan fitur yang tidak disediakan oleh Laravel (misalnya, menggunakan PDO secara langsung), gunakan prepared statements untuk menjalankan kueri parameterized.

Contoh penggunaan prepared statements:

```
$statement = $pdo->prepare("SELECT * FROM users WHERE username = :username");
$statement->bindParam(':username', $username);
$statement->execute();
```

5. **Proteksi Terhadap *Cross-Site Scripting* (XSS)**

Laravel menyediakan fasilitas proteksi terhadap serangan XSS dengan menggunakan fitur Blade, yang secara otomatis akan mengekstrak karakter berbahaya dari output. Selain itu, Anda juga dapat menggunakan fungsi Laravel seperti `{{ $var }}` atau `{!! $var !!}` untuk menghindari serangan XSS.

6. **Enkripsi Data Sensitif**

Laravel menyediakan fasilitas untuk enkripsi data sensitif menggunakan fitur enkripsi bawaan. Anda dapat menggunakan helper functions seperti *encrypt()* dan *decrypt()* untuk mengenkripsi dan mendekripsi data sensitif sebelum disimpan ke dalam *database*.

Berikut adalah cara menggunakan fitur enkripsi di Laravel:

a. **Konfigurasi Kunci Enkripsi**

Konfigurasikan kunci enkripsi yang kuat dalam file **.env** Anda. Kunci enkripsi ini digunakan untuk mengamankan data yang dienkripsi dan harus disimpan secara rahasia.

```
ENCRYPTION_KEY=your-encryption-key
```

b. **Menggunakan Helper Functions**

Laravel menyediakan sejumlah fungsi bantu (*helper functions*) yang dapat digunakan untuk mengenkripsi dan mendekripsi data. Dua fungsi utama yang digunakan adalah **encrypt()** dan **decrypt()**.
Untuk mengenkripsi data, gunakan fungsi **encrypt()**:

```
$encryptedData=encrypt('Sensitive data');
```

Untuk mendekripsi data yang telah dienkripsi, gunakan fungsi **decrypt()**:

```
$decryptedData=decrypt($encrypte
dData);
```

c. **Menggunakan Facade Crypt:**

Facade **Crypt** untuk mengenkripsi dan mendekripsi data dalam aplikasi. Facade ini memberikan akses ke fungsi-fungsi enkripsi Laravel secara langsung.
Contoh penggunaan Crypt facade untuk mengenkripsi data:

```
$encryptedData =
Crypt::encryptString('Sensitive
data');
```

Contoh penggunaan Crypt facade untuk mendekripsi data:

```
$decryptedData =
Crypt::decryptString($encryptedD
ata);
```

d. **Enkripsi Data pada Model Eloquent:**

Jika ingin secara otomatis mengenkripsi dan mendekripsi atribut model Eloquent, dapat digunakan fitur enkripsi yang disediakan oleh Laravel.
Tambahkan atribut yang ingin dienkripsi dalam properti **$encryptable** pada model.

```
class User extends Authenticatable
{
    use Encryptable;

    protected $encryptable = [
        'email',
        'credit_card_number',
    ];
}
```

7. **Monitoring dan Logging**

Penting untuk memantau aktivitas dan memeriksa log secara teratur untuk mendeteksi aktivitas mencurigakan atau percobaan serangan. Dapat menggunakan fitur logging Laravel untuk menyimpan log ke berbagai penyimpanan seperti file log, database, atau layanan pihak ketiga.

KESIMPULAN

Aplikasi web aplikasi web yang dibangun menggunakan framework Laravel telah diimplementasikan dengan langkah-langkah keamanan yang tepat. Proteksi terhadap serangan SQL Injection, Cross-Site Scripting (XSS), dan Cross-Site Request Forgery (CSRF) telah diterapkan dengan efektif, menjadikan aplikasi web lebih aman dan terlindungi dari berbagai ancaman keamanan.

DAFTAR PUSTAKA

Aljawarneh, S. A. (2020). Cybersecurity Risks in Remote Learning: Problems and Solutions. *Information Systems Management*, *37*(4), 299-308.

Halfond, W. G. J., & Orso, A. (2010). Amnesia: analysis and monitoring for neutralizing SQL-injection attacks. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *19*(1), 2-2.

Herdiansah, A., Sugiyani, Y., Fitriawati, N., & Cholid, H. N. (2023). Sistem Informasi Akademik Penilaian Hasil Kegiatan Belajar Mengajar Sekolah Menengah Pertama. *JIKA (Jurnal Informatika)*, *7*(3), 364–370. https://doi.org/10.31000/jika.v7i3.8838

Herdiansah, A., Sugiyani, Y., Septarini, R. S., & Mahpud, M. (2022). *Penerapan Metode Pemodelan UML (Unified Modelling Language) dan RAD (Rapid Application Development) pada Pembangunan Sistem Informasi Akademik Sekolah* (A. Wahdi, Ed.; 1st ed.). CV. Dewa Publishing

Litchfield, D., & Anley, C. (2007). *The Database Hacker's Handbook: Defending Database Servers*. Wiley Publishing.

Mell, P., & Scarfone, K. (2007). Common Vulnerabilities and Exposures (CVE). *Journal of Cyber Security and Mobility*, *1*(1), 107-112.

Nurofik, A., Rahajeng, E., Munti, N. Y. S., Sutisna, Firmansyah, H., Sani, A., Hendarsyah, D., Adrianto, S., Darma, W. A., Herdiansah, A., Ariestiandy, D., Nurnaningsih, D., Setiawan, I., Wiyono, A. S., & Zaharah. (2021). *Pengantar Teknologi Informasi* (I. Kusumawati & M. Sari, Eds.; Ed.1). Insania

OWASP. (2017). OWASP Top Ten. OWASP Foundation.

OWASP. (2021). Open Web Application Security Project (OWASP). Diakses dari: https://owasp.org/

Russell, S., Ganguli, S., & Raghunathan, A. (2018). *Web Application Security is a Stack: How to CYA (Cover Your Apps) Completely.* O'Reilly Media

Stamp, M., Black, U., & Zakin, I. (2016). *Web Security: A Beginner's Guide.* McGraw-Hill Education.

Susanti, S., & Irawan, C. (2023). Sistem Informasi Fleet Management Menggunakan Framework Laravel pada PT. Sajira Mahardika. *JIKA (Jurnal Informatika)*, *7*(4), 415–422. https://doi.org/10.31000/jika.v7i4.8574

Taufiq, R., Heriyanto, H., Destriana, R., Faridi, F., & Nurnaningsih, D. (2023). Perancangan Sistem Informasi Penjualan Roti Kurni Bakeri Berbasis Web Menggunakan Metode Waterfall. *JIKA (Jurnal Informatika)*, *7*(3), 292–298. https://doi.org/10.31000/jika.v7i3.8298

Verizon. (2020). Data Breach Investigations Report (DBIR). Verizon Communications.

Yanuarti, E., Sarwindah, S., Perkasa, E. B., & Briliantza, A. (2022). Penerapan Model RAD Dalam Sistem Administrasi Layanan TV Kabel Berbasis Web. *JIKA (Jurnal Informatika)*, *6*(3), 220–226. https://doi.org/10.31000/jika.v6i3.6229