

## **Implementasi Framework CodeIgniter Untuk Keamanan Point Of Sales (Pos)**

*Fransesco<sup>a</sup>, Wahyu Joni Kurniawan<sup>b</sup>*

<sup>a</sup>Institut Bisnis dan Teknologi Pelita Indonesia, Jl. Jend. Ahmad Yani No. 78-88 Pekanbaru, [fransesco.037@student.pelitaindonesia.ac.id](mailto:fransesco.037@student.pelitaindonesia.ac.id)

<sup>b</sup>Institut Bisnis dan Teknologi Pelita Indonesia, Jl. Jend. Ahmad Yani No. 78-88 Pekanbaru, [wahyu.jonikurniawan@lecturer.pelitaindonesia.ac.id](mailto:wahyu.jonikurniawan@lecturer.pelitaindonesia.ac.id)

### **INFORMASI ARTIKEL**

#### *Sejarah Artikel:*

Diterima Redaksi: 25 April 2026

Revisi Akhir: 30 Maret 2026

Diterbitkan Online: 30 April 2026

### **KATA KUNCI**

CodeIgniter, Point of Sale, Keamanan Web, SQL Injection, CSRF.

### **KORESPONDENSI**

E-mail: [fransesco.037@student.pelitaindonesia.ac.id](mailto:fransesco.037@student.pelitaindonesia.ac.id)

### **A B S T R A C T**

Pertumbuhan Usaha Kecil dan Menengah (UKM) di sektor kuliner menuntut penggunaan sistem Point of Sale (POS) yang efisien, namun aspek keamanan aplikasi sering kali terabaikan karena penggunaan solusi yang dikembangkan secara kurang terstruktur. Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem POS berbasis framework CodeIgniter pada Kedai Kopi Megaria serta membandingkan tingkat keamanannya dengan sistem yang dibangun tanpa framework (native). Metode penelitian yang digunakan adalah Research and Development (R&D) dengan model pengembangan Waterfall yang meliputi tahapan analisis, perancangan, implementasi, dan pengujian. Pengujian keamanan dilakukan melalui metode White Box Testing serta menggunakan alat bantu standar industri seperti OWASP ZAP dan SQLMap untuk mendeteksi celah kerentanan. Hasil penelitian menunjukkan bahwa sistem berbasis CodeIgniter berhasil memitigasi risiko serangan SQL Injection, Cross-Site Scripting (XSS), dan Cross-Site Request Forgery (CSRF) secara efektif melalui fitur keamanan bawaan seperti query binding, XSS filtering, dan CSRF protection. Penggunaan framework ini sangat direkomendasikan bagi pengembang aplikasi bisnis skala UKM guna melindungi aset data transaksi dari potensi kerugian finansial akibat celah keamanan web.

## **1. PENDAHULUAN**

Perkembangan Usaha Kecil dan Menengah (UKM) di sektor kuliner saat ini mendorong adopsi teknologi informasi, khususnya sistem Point of Sale (POS), untuk meningkatkan efisiensi operasional. Namun, aspek keamanan aplikasi sering kali terabaikan karena penggunaan solusi yang dikembangkan secara kurang terstruktur atau tanpa framework yang baku [1]. Kondisi ini relevan dengan Kedai Kopi Megaria yang masih mengandalkan pencatatan sederhana sehingga rentan terhadap kesalahan input, kehilangan data, hingga potensi manipulasi laporan. Ancaman keamanan web seperti SQL Injection, Cross-Site Scripting (XSS), dan Cross-Site Request Forgery (CSRF) berdampak langsung pada integritas data serta kepercayaan pelanggan [2]. Celah keamanan pada sistem POS memiliki dampak ekonomi dan reputasi yang signifikan, di mana satu insiden kebocoran data dapat merusak aset paling berharga dalam bisnis kuliner [3]. Sebagai solusi, penggunaan framework CodeIgniter menawarkan fitur keamanan komprehensif dengan struktur yang ringan dan efisien untuk tim dengan sumber daya terbatas [4]. Penerapan fitur seperti query binding dan CSRF token terbukti secara signifikan menurunkan keberhasilan serangan injeksi dibandingkan dengan pengembangan sistem secara manual [5]. Selain itu, mekanisme keamanan bawaan ini sangat efektif dalam melindungi integritas data transaksi dari

serangan umum aplikasi web. Oleh karena itu, penelitian ini bertujuan untuk mengimplementasikan sistem POS berbasis CodeIgniter pada Kedai Kopi Megaria guna meningkatkan keamanan data transaksi.

## **2. TINJAUAN PUSTAKA**

### **2.1. Sistem Point of Sales (POS)**

Point of Sale (POS) merupakan sistem informasi operasional yang berfungsi untuk mendukung proses transaksi penjualan antara penjual dan pelanggan secara real-time. Dalam perkembangannya, sistem POS telah bertransformasi dari pencatatan manual menjadi sistem berbasis web yang memungkinkan pengelolaan data penjualan, stok, dan laporan secara lebih akurat, konsisten, serta fleksibel untuk diintegrasikan dengan sistem lain. Implementasi POS yang efektif sangat bergantung pada integrasi perangkat keras dan perangkat lunak yang mampu mengurangi risiko kesalahan manusia dalam aktivitas bisnis harian [1].

### **2.2. Keamanan Aplikasi Web**

Keamanan aplikasi web bertujuan untuk melindungi sistem dari akses tidak sah dan manipulasi data sensitif, terutama pada aplikasi yang berinteraksi langsung dengan pengguna [6]. Terdapat tiga ancaman utama yang sering menyerang aplikasi web: SQL Injection yang memanipulasi perintah basis data

melalui input yang tidak divalidasi, Cross-Site Scripting (XSS) yang menyisipkan skrip berbahaya ke browser pengguna, dan Cross-Site Request Forgery (CSRF) yang memalsukan permintaan transaksi dari pengguna yang telah terautentikasi [7]. Oleh karena itu, penerapan mekanisme validasi input dan manajemen sesi yang ketat sangat diperlukan untuk menjaga integritas data.

### 2.3. Framework CodeIgniter

CodeIgniter adalah framework PHP yang menggunakan arsitektur Model-View-Controller (MVC) untuk memisahkan logika bisnis, antarmuka pengguna, dan pengelolaan data [6]. Berbeda dengan pengembangan native PHP yang memerlukan pengamanan manual, CodeIgniter menyediakan fitur keamanan bawaan seperti CSRF protection dengan token otomatis, XSS filtering, dan Query Builder yang menggunakan parameter binding untuk mencegah serangan SQL Injection [8]. Penggunaan framework ini secara signifikan meningkatkan keteraturan kode, mempercepat proses pengembangan, dan memudahkan pemeliharaan sistem bagi pengembang dengan sumber daya terbatas [6].

### 2.4. Alat Penguji Keamanan

Untuk mendeteksi celah keamanan secara sistematis, diperlukan perangkat lunak khusus seperti OWASP ZAP (Zed Attack Proxy) dan SQLMap. OWASP ZAP berfungsi sebagai proxy untuk menganalisis lalu lintas data dan mendeteksi kerentanan konfigurasi serta serangan XSS atau CSRF [7]. Sementara itu, SQLMap merupakan alat uji penetrasi otomatis yang dirancang khusus untuk mendeteksi dan mengeksploitasi celah SQL Injection pada parameter basis data aplikasi [9].

## 3. METODOLOGI

Bagian ini menguraikan tahapan sistematis dalam perancangan, implementasi, serta pengujian keamanan sistem Point of Sale (POS) menggunakan framework CodeIgniter. Pendekatan yang digunakan adalah penelitian dan pengembangan untuk memvalidasi ketahanan sistem terhadap berbagai ancaman keamanan web.

### 3.1. Kerangka Penelitian

Penelitian ini dijalankan melalui kerangka kerja yang terstruktur, diawali dengan studi literatur untuk memperoleh landasan teori mengenai keamanan sistem informasi dan fungsionalitas framework CodeIgniter. Tahap berikutnya adalah analisis kebutuhan sistem untuk mengidentifikasi fitur utama POS serta potensi celah keamanan yang perlu dimitigasi, yang kemudian dituangkan ke dalam perancangan arsitektur aplikasi dan alur data menggunakan diagram UML. Setelah rancangan selesai, dilakukan implementasi sistem dengan membangun modul-modul penting seperti autentikasi dan manajemen transaksi menggunakan praktik pengkodean aman. Seluruh rangkaian ini diakhiri dengan pengujian keamanan menggunakan metode White Box untuk memastikan setiap fungsi berjalan sesuai rancangan, serta evaluasi hasil untuk menganalisis efektivitas framework dalam meningkatkan keamanan data bisnis.

### 3.2. Metode Penelitian

Metode penelitian yang diterapkan dalam pengembangan ini adalah Research and Development (R&D) dengan model pengembangan Software Development Life Cycle (SDLC) berbasis Waterfall. Pemilihan model Waterfall didasari oleh kebutuhan akan tahapan pengembangan perangkat lunak yang sistematis dan berurutan, mulai dari spesifikasi kebutuhan hingga tahap pengujian fungsionalitas dan keamanan. Dengan model ini, setiap fase pengembangan dapat terdokumentasi dengan baik, sehingga perbandingan tingkat keamanan antara sistem berbasis framework dan sistem non-framework dapat dilakukan secara akurat.

### 3.3. Pengumpulan dan Jenis Data

Data yang digunakan dalam penelitian ini diklasifikasikan menjadi dua jenis, yaitu data primer dan data sekunder. Data primer diperoleh secara langsung melalui aktivitas observasi dan simulasi transaksi pada sistem POS yang dikembangkan, yang mencakup data log login pengguna serta hasil rekaman pengujian keamanan sistem. Sementara itu, data sekunder dikumpulkan melalui studi pustaka terhadap jurnal ilmiah, buku teks, dan dokumentasi teknis yang relevan dengan topik keamanan aplikasi web serta arsitektur framework CodeIgniter guna mendukung validitas hasil penelitian.

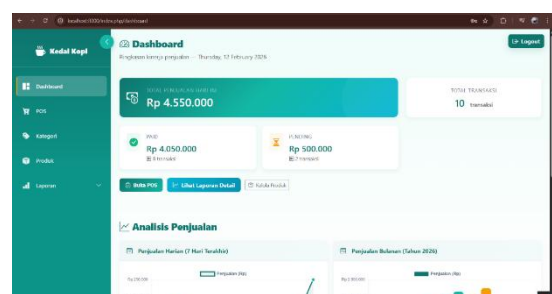
### 3.4. Teknik Pengujian Keamanan

Teknik pengujian difokuskan pada validasi ketahanan sistem terhadap serangan siber melalui metode White Box Testing dan Security Testing. White Box Testing dilakukan dengan menelusuri alur logika internal kode sumber untuk memastikan bahwa setiap jalur eksekusi bebas dari kesalahan logika yang dapat menimbulkan celah keamanan. Selain itu, dilakukan uji keamanan otomatis menggunakan alat bantu standar industri untuk mensimulasikan serangan nyata, yang meliputi uji SQL Injection dengan manipulasi query pada form input, uji Cross-Site Scripting (XSS) dengan penyisipan skrip berbahaya, serta uji Cross-Site Request Forgery (CSRF) melalui simulasi permintaan palsu. Hasil dari pengujian ini kemudian dianalisis secara komparatif untuk menilai perbedaan tingkat kerentanan antara sistem yang menggunakan fitur keamanan bawaan framework dengan sistem yang dibangun secara manual.

## 4. HASIL DAN PEMBAHASAN

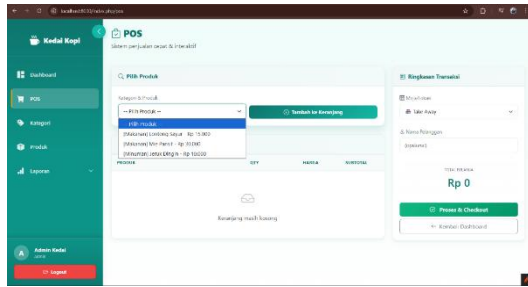
### 4.1. Implementasi Sistem Point of Sales (POS)

Sistem POS Kedai Kopi Megaria telah berhasil diimplementasikan menggunakan framework CodeIgniter dengan arsitektur Model-View-Controller (MVC) yang memisahkan logika bisnis, antarmuka pengguna, dan pengelolaan



Gambar 1 Halaman Dashboard

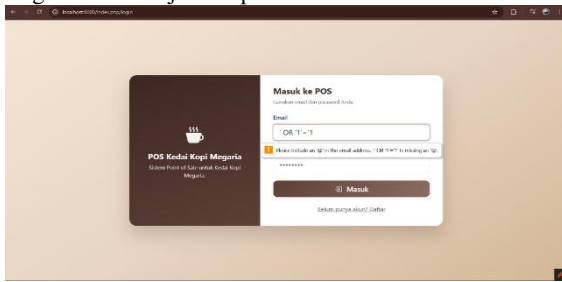
data. Antarmuka utama sistem, seperti yang terlihat pada Gambar 1. menyajikan ringkasan data operasional mulai dari pendapatan harian hingga grafik tren penjualan yang diambil secara otomatis dari basis data. Selain itu, modul transaksi penjualan pada Gambar 2. memungkinkan kasir melakukan input pesanan secara efisien dengan fitur filter kategori produk.



Gambar 2 Halaman Transaksi Penjualan

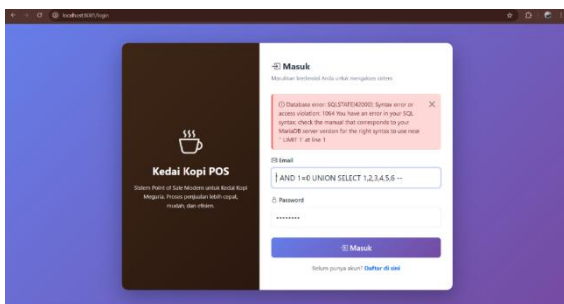
#### 4.2. Pengujian Keamanan White Box

Pengujian struktur internal dilakukan untuk membandingkan ketahanan sistem berbasis framework dan non-framework terhadap serangan siber. Pada sistem CodeIgniter, upaya serangan SQL Injection menggunakan payload 'OR '1'='1' pada form login dinyatakan gagal karena sistem telah menerapkan mekanisme query binding dan validasi input yang ketat, sebagaimana ditunjukkan pada Gambar 3.



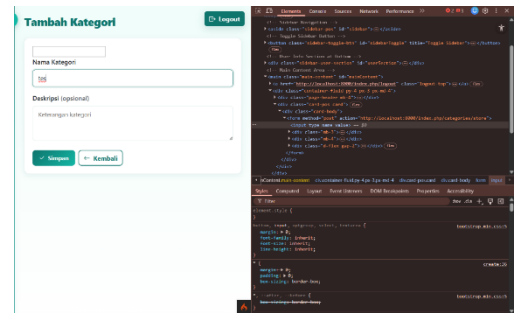
Gambar 3 Hasil Pengujian SQL Injection Pada POS CodeIgniter

Sebaliknya, pengujian pada sistem non-framework menunjukkan kerentanan kritis di mana input berbahaya diproses langsung oleh sistem dan menampilkan pesan kesalahan database yang dapat dieksploitasi lebih lanjut (Gambar 4).

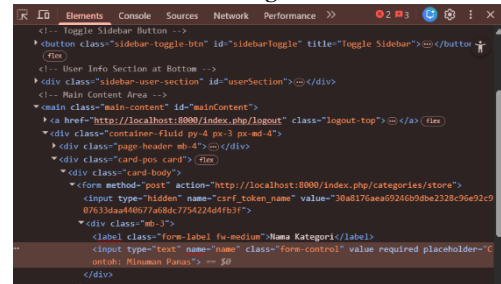


Gambar 4 Hasil Pengujian SQL Injection Pada POS Non-Framework

Terkait proteksi CSRF, sistem CodeIgniter terbukti aman karena setiap formulir menyertakan token unik; ketika token dihapus secara sengaja pada Gambar 5. Sistem secara otomatis menolak permintaan penyimpanan data (Gambar 6). Hasil pengujian logika ini dirangkum dalam Tabel 1 dan Tabel 2 yang menunjukkan perbedaan signifikan pada efektivitas filtrasi input kedua sistem.



Gambar 5 Hasil Pengujian CSRF Pada POS CodeIgniter



Gambar 6 Sistem Menolak Penyimpanan Data

Tabel 1 : Pengujian White Box POS Framework CodeIgniter

| Jenis Serangan | Skenario Pengujian                  | Hasil            | Keterangan     |
|----------------|-------------------------------------|------------------|----------------|
| SQL Injection  | Input 'OR '1'='1' pada form login   | Gagal Login      | Tidak rentan   |
| XSS            | Input <script>alert('XSS')</script> | Tidak Dieksekusi | Terfilter      |
| CSRF           | Kirim request tanpa token           | Ditolak          | Proteksi aktif |

Tabel 2 : Pengujian White Box POS Non-Framework

| Jenis Serangan | Skenario Pengujian                            | Hasil                                 | Keterangan     |
|----------------|-----------------------------------------------|---------------------------------------|----------------|
| SQL Injection  | Input 'OR '1'='1' pada form login             | Gagal Login, Sistem Menampilkan Error | Rentan         |
| XSS            | Input <img src=x onerror=alert('XSS Stored')> | Perintah Dieksekusi                   | Terfilter      |
| CSRF           | Kirim request tanpa token                     | Diterima                              | Proteksi aktif |

#### 4.3. Analisis Kerentanan Menggunakan OWASP ZAP dan SQLMap

Pengujian otomatis menggunakan standar industri memperkuat temuan pada tahap sebelumnya. Pemindaian dengan OWASP Zed Attack Proxy (ZAP) pada sistem berbasis framework hanya menghasilkan alert risiko rendah terkait konfigurasi header

keamanan seperti CSP dan X-Content-Type (Gambar 7).



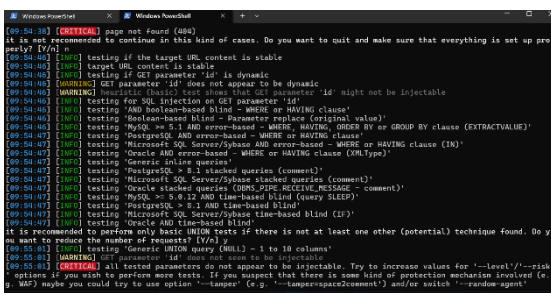
**Gambar 7 Hasil Penyerangan ZAP Pada POS CodeIgniter**

Namun, pada sistem non-framework, ZAP mendeteksi risiko tingkat sedang berupa absennya token anti-CSRF pada halaman registrasi (Gambar 8).



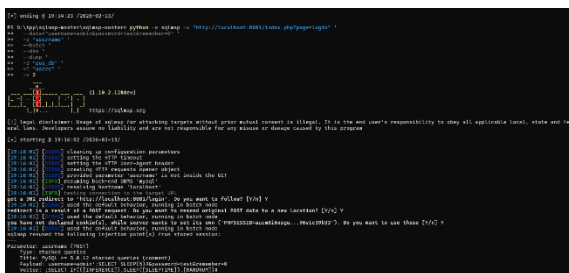
**Gambar 8 Hasil Penyerangan ZAP Pada POS Non-Framework**

Selanjutnya, pengujian dengan SQLMap pada Gambar 4.39 menunjukkan bahwa aplikasi CodeIgniter tidak rentan terhadap teknik boolean-based maupun time-based blind injection.



**Gambar 9 Hasil Penyerangan SQLMap Pada POS CodeIgniter**

Di sisi lain, SQLMap berhasil mendeteksi celah pada parameter username sistem non-framework menggunakan teknik stacked queries yang memungkinkan penyerang mengeksekusi perintah SQL tambahan secara ilegal (Gambar 4.41).



**Gambar 10 Hasil Penyerangan SQLMap Pada POS Non-Framework**

Rekapitulasi hasil pengujian tersedia pada Tabel 3 dan Tabel 4.

**Tabel 3 : Hasil Pengujian POS CodeIgniter**

| No | Jenis Uji | Parameter                  | Hasil                      |
|----|-----------|----------------------------|----------------------------|
| 1  | ZAP       | CSRF Token                 | Tidak ditemukan kerentanan |
| 2  | ZAP       | Security Header            | Aman                       |
| 3  | ZAP       | Cookie HttpOnly & SameSite | Aman                       |
| 4  | ZAP       | Clickjacking               | Tidak ditemukan kerentanan |
| 5  | ZAP       | Information Disclosure     | Tidak ditemukan kerentanan |
| 6  | SQLMap    | SQL Injection              | Tidak rentan               |
| 7  | SQLMap    | Dump Database              | Tidak berhasil             |
| 8  | SQLMap    | Ekstraksi Data             | Tidak berhasil             |

**Tabel 4 : Hasil Pengujian Pada POS Non-Framework**

| Jenis Uji | Parameter                  | Hasil                      |
|-----------|----------------------------|----------------------------|
| ZAP       | CSRF Token                 | Tidak ditemukan kerentanan |
| ZAP       | Security Header            | Aman                       |
| ZAP       | Cookie HttpOnly & SameSite | Aman                       |
| ZAP       | Clickjacking               | Tidak ditemukan kerentanan |
| ZAP       | Information Disclosure     | Tidak ditemukan kerentanan |
| SQLMap    | SQL Injection              | Tidak rentan               |
| SQLMap    | Dump Database              | Tidak berhasil             |
| SQLMap    | Ekstraksi Data             | Tidak berhasil             |

#### 4.4. Analisis Komparatif Tingkat Keamanan

**Tabel 5 : Perbandingan POS Framework CodeIgniter dan POS Non-Framework**

| Aspek Pengujian          | POS Framework                                                                                                            | POS Non-Framework                                                                                                  |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| Hasil OWASP ZAP          | Terdapat beberapa alert dengan tingkat risiko rendah (Low Risk), mayoritas berupa peringatan konfigurasi header keamanan | Terdapat alert dengan tingkat risiko sedang hingga rendah, termasuk potensi celah keamanan pada konfigurasi sistem |
| Jenis Alert ZAP          | Missing Security Header, Cookie Without Secure Flag, dan informasi konfigurasi server                                    | Missing Security Header, Cookie Without Secure Flag, serta potensi kesalahan validasi input                        |
| Hasil SQLMap             | Tidak ditemukan indikasi kerentanan SQL Injection                                                                        | Ditemukan potensi parameter yang rentan terhadap SQL Injection                                                     |
| Teknik SQLMap            | Boolean-based blind, time-based blind (tidak berhasil menembus sistem)                                                   | Boolean-based blind dan time-based blind berhasil mendeteksi potensi celah                                         |
| Tingkat Kerentanan       | Rendah                                                                                                                   | Sedang                                                                                                             |
| Keamanan Query Database  | Menggunakan ORM dan prepared statement sehingga lebih aman                                                               | Menggunakan query manual sehingga lebih rentan                                                                     |
| Proteksi Input           | Validasi input otomatis oleh framework                                                                                   | Validasi input dilakukan manual dan belum optimal                                                                  |
| Sistem Proteksi Tambahan | Memiliki proteksi bawaan framework                                                                                       | Tidak memiliki proteksi bawaan                                                                                     |
| Kesimpulan Keamanan      | Lebih aman terhadap serangan web dan SQL Injection                                                                       | Lebih rentan terhadap serangan SQL Injection                                                                       |

Berdasarkan analisis perbandingan yang dituangkan dalam Tabel 4.5, sistem POS berbasis framework terbukti memiliki tingkat kerentanan yang lebih rendah dibandingkan sistem non-

framework. Keunggulan utama terletak pada penggunaan Object-Relational Mapping (ORM) dan prepared statements yang secara otomatis memitigasi serangan injeksi, serta adanya sistem proteksi bawaan untuk CSRF dan XSS yang tidak tersedia pada pengembangan secara manual (native). Hal ini mengonfirmasi bahwa implementasi framework CodeIgniter secara signifikan meningkatkan stabilitas dan keandalan sistem dalam melindungi data transaksi bisnis.

## 5. KESIMPULAN DAN SARAN

### 5.1. Kesimpulan

Berdasarkan seluruh rangkaian penelitian, perancangan, hingga pengujian teknis yang telah dilaksanakan, dapat disimpulkan bahwa implementasi framework CodeIgniter pada sistem Point of Sale (POS) Kedai Kopi Megaria telah berhasil menjawab rumusan masalah dan mencapai tujuan penelitian secara komprehensif. Melalui penerapan metode Waterfall, sistem ini dirancang dengan arsitektur Model-View-Controller (MVC) yang tidak hanya mampu mengotomatisasi pencatatan transaksi, pengelolaan stok, dan laporan penjualan secara akurat, tetapi juga memberikan perlindungan keamanan yang jauh lebih andal dibandingkan pengembangan secara manual. Fitur keamanan bawaan CodeIgniter, seperti Query Binding, XSS Filtering, dan CSRF Protection, terbukti efektif dalam memitigasi serangan umum pada aplikasi web yang dapat merusak integritas data bisnis. Hasil pengujian menggunakan alat bantu OWASP ZAP dan SQLMap menunjukkan perbedaan tingkat keamanan yang sangat signifikan antara kedua model pengembangan yang diuji. Pada sistem berbasis CodeIgniter, tidak ditemukan kerentanan kritis yang dapat dieksploitasi, di mana temuan hanya bersifat informatif terkait rekomendasi konfigurasi header keamanan. Sebaliknya, sistem non-framework ditemukan memiliki kerentanan kritis terhadap serangan SQL Injection melalui teknik stacked queries dan sangat rawan terhadap manipulasi data via CSRF karena ketiadaan mekanisme token keamanan. Secara keseluruhan, penelitian ini membuktikan bahwa penggunaan framework CodeIgniter secara signifikan meningkatkan stabilitas, keandalan, dan keamanan sistem POS bagi pelaku UKM guna melindungi aset ekonomi dan reputasi bisnis dari potensi kerugian finansial akibat celah keamanan aplikasi web.

### 5.2. Saran

Terdapat beberapa saran yang dapat dipertimbangkan untuk pengembangan sistem di masa mendatang guna meningkatkan kualitas layanan dan keamanan. Pengembang disarankan untuk melakukan penguatan pada konfigurasi header keamanan, khususnya dengan menerapkan Content Security Policy (CSP) yang lebih ketat serta mengaktifkan HTTP Strict Transport Security (HSTS) guna menutup celah risiko rendah yang ditemukan pada saat pengujian. Selain itu, pemeliharaan dan pembaruan rutin terhadap versi framework CodeIgniter serta versi PHP menjadi hal yang sangat penting agar sistem tetap terlindungi dari kerentanan keamanan terbaru.

Dari sisi fungsionalitas, sistem ini dapat dikembangkan lebih lanjut dengan menambahkan integrasi metode pembayaran digital (e-wallet) serta modul manajemen inventaris yang lebih mendalam untuk meningkatkan efisiensi operasional UKM. Terakhir, bagi peneliti selanjutnya, disarankan untuk memperluas cakupan keamanan hingga ke lapisan jaringan (network security)

dan pengerasan server (server hardening), mengingat fokus penelitian ini masih terbatas pada keamanan di sisi lapisan aplikasi (application layer).

## DAFTAR PUSTAKA

- [1] S. Sukirno and H. Suhendar, "Pengembangan Sistem Point of Sale Menggunakan Framework Codeignitier Berbasis Web," *J. Algoritma*, vol. 19, no. 2, pp. 660–668, Nov. 2022, doi: 10.33364/algoritma/v.19-2.1181.
- [2] M. Ridwan, T. H. Sinaga, and M. Elsera, "Penerapan Framework Codeigniter Dalam Perancangan Aplikasi Manajemen Iuran Perumahan Griya Mandiri," *Djtechno J. Teknol. Inf.*, vol. 3, no. 1, pp. 49–58, 2022, doi: 10.46576/djtechno.v3i1.2196.
- [3] P. Kustianto, S. Auliana, B. Rakhim Setya Permana, A. Rohman, and A. Munawir, "Penerapan Framework Codeigniter 3 Pada Aplikasi Penjualan Di Toko Kue Kering Miya Jaya Serang," *JATI (Jurnal Mhs. Tek. Inform.)*, vol. 9, no. 1, pp. 831–836, 2024, doi: 10.36040/jati.v9i1.12410.
- [4] M. Muqorobin and N. A. R. Rais, "Comparison of PHP Programming Language with Codeigniter Framework in Project CRUD," *Int. J. Comput. Inf. Syst.*, vol. 3, no. 3, pp. 94–98, Aug. 2022, doi: 10.29040/ijcis.v3i3.77.
- [5] S. Sibagariang and M. E. P. Sinaga, "Rancang Bangun Aplikasi Point Of Sales Menggunakan Framework Codeigniter," *J. MAHAJANA Inf.*, vol. 8, no. 1, pp. 26–35, Jun. 2023, doi: 10.51544/jurnalmi.v8i1.4081.
- [6] N. G. Rachmadi, A. Febrian, and C. Anwar, "Pengujian Perangkat Lunak Berstandar ISO/IEC 29119 pada Website Literasi Digital 'SukaBaca,'" *OKTAL J. Ilmu Komput. dan Sci.*, vol. 4, no. 12, pp. 846–860, 2025, [Online]. Available: <https://github.com/nazazulfiqi/suka-baca-be>
- [7] Abdul Fattah Hasibuan, Tommy, and Divi Handoko, "Analisis Kerentanan Website Dengan Aplikasi Owasap Zap," *J. Ilmu Komput. dan Sist. Inf.*, vol. 2, no. 2, pp. 257–270, Jun. 2023, doi: 10.70340/jirsi.v2i2.51.
- [8] T. Penulis *et al.*, "SISTEM MANAJEMEN BASIS DATA MENGGUNAKAN MySQL", Accessed: Feb. 14, 2026. [Online]. Available: [www.freepik.com](http://www.freepik.com)
- [9] H. Salsabila and R. F. Fithra, "PEMANFAATAN SQLMAP UNTUK DETEKSI SQL INJECTION PADA SITUS WEB," *J. Inform. dan Tek. Elektro Terap.*, vol. 13, no. 3, Jul. 2025, doi: 10.23960/jitet.v13i3.6507.