BHATARA: Jurnal Multidisiplin
Vol. 3, No.2, April 2026, pp. 54-61, ISSN: 2476-9843

Uji Coba SQL Injection pada Sistem Login Aplikasi Pemesanan Makanan Berbasis Website Menggunakan Metode Rijndael

Mohammad Rifki Ainurrahim
[1]Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Amikom Yogyakarta

Jl. Ring Road Utara, Ngringin, Condongcatur, Kec. Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta 55281

Corresponding e-mail: m.rifki.ainurrahim@students.amikom.ac.id

Abstrak

Aplikasi pemesanan makanan berbasis web menghadapi tantangan keamanan serius pada sistem login, terutama dari ancaman SQL Injection yang berisiko menyebabkan kebocoran data pengguna. Penelitian ini bertujuan untuk mengimplementasikan algoritma Rijndael (AES) guna mengamankan password pengguna dan menguji ketahanan sistem login terhadap serangan In-band SQL Injection. Metode penelitian meliputi perancangan prototipe aplikasi web dengan framework Flask, implementasi enkripsi AES-CBC menggunakan PyCryptodome, dan pengujian keamanan melalui simulasi serangan. Hasil penelitian menunjukkan bahwa enkripsi AES berhasil mengamankan password dalam bentuk ciphertext di basis data. Namun, pengujian penetrasi membuktikan bahwa enkripsi saja tidak cukup untuk mencegah SQL Injection; sistem yang menggunakan parameterized query berhasil menangkal seluruh percobaan serangan, sedangkan sistem dengan konkatenasi string rentan dieksploitasi. Dengan demikian, penelitian ini menyimpulkan bahwa pendekatan keamanan berlapis (defense in depth) yang menggabungkan enkripsi untuk perlindungan data at rest dan parameterized query sebagai pertahanan aktif merupakan strategi esensial untuk membangun sistem login yang tangguh. Temuan ini memberikan panduan praktis bagi pengembang dalam menerapkan keamanan aplikasi web yang komprehensif.

Kata Kunci: *Rijndael, SQL Injection, Keamanan Website, Enkripsi AES, Parameterized Query*

Abstract

Web-based food ordering applications face serious security challenges in their login systems, particularly from SQL injection attacks that risk user data leakage. This study aims to implement the Rijndael (AES) algorithm to secure user passwords and test the login system's resilience against In-band SQL Injection attacks. The research method involves designing a web application prototype using the Flask framework, implementing AES-CBC encryption with PyCryptodome, and conducting security testing through simulated attacks. The results show that AES encryption successfully secures passwords as ciphertext in the database. However, penetration testing proves that encryption alone is insufficient to prevent SQL Injection; systems using parameterized queries successfully thwarted all attack attempts, while systems with string concatenation remained vulnerable to exploitation. Therefore, this research concludes that a layered security approach (defense in depth), combining encryption for data at rest protection and parameterized queries as active defense, is an essential strategy for building a robust login system. These findings provide practical guidance for developers in implementing comprehensive web application security.

Keywords: *Rijndael, SQL Injection, Website Security, AES Encryption, Query Parameterized*

BHATARA: Jurnal Multidisiplin
Vol. 3, No.2, April 2026, pp. 54-61, ISSN: 2476-9843

1. Pendahuluan

Aplikasi pemesanan makanan berbasis web telah menjadi bagian integral dari kehidupan masyarakat modern, menawarkan kemudahan dan efisiensi dalam bertransaksi secara digital. Namun, popularitas aplikasi ini juga diiringi oleh meningkatnya risiko keamanan siber, terutama pada sistem login yang menjadi gerbang utama akses pengguna. Berbagai studi menunjukkan bahwa sistem autentikasi sering menjadi target utama serangan eksploitasi karena mengelola data sensitif seperti kredensial pengguna (Pakpahan & Prayino, 2021). Oleh karena itu, keamanan informasi pada lapisan ini merupakan faktor krusial yang menentukan kepercayaan pengguna dan keberlangsungan bisnis. Tanpa perlindungan yang memadai, aplikasi dapat menjadi sasaran empuk bagi ancaman yang dapat mengakibatkan kerugian materiil dan non-materiil. Salah satu ancaman paling berbahaya dan umum terhadap aplikasi web adalah serangan SQL Injection, di mana penyerang menyisipkan perintah SQL berbahaya melalui input pengguna untuk memanipulasi basis data. Serangan ini dapat mengakibatkan pencurian data sensitif, modifikasi data, hingga pengambilalihan kontrol sistem secara penuh (Andriyanto et al., 2020). Ancaman ini diperparah oleh fakta bahwa banyak pengembang masih mengabaikan praktik pengkodean aman, sehingga celah keamanan ini sering kali tereksploitasi. Dalam konteks aplikasi pemesanan makanan, kebocoran data pengguna seperti nama, alamat, dan detail pembayaran dapat merusak reputasi penyedia layanan secara signifikan. Oleh karena itu, diperlukan pendekatan proaktif untuk mengidentifikasi dan memitigasi kerentanan ini sejak tahap pengembangan. Guna menjawab tantangan keamanan tersebut, penerapan kriptografi menjadi salah satu solusi pertahanan yang kuat dan telah terbukti. Algoritma Rijndael, yang telah distandardisasi sebagai Advanced Encryption Standard (AES), menawarkan mekanisme enkripsi yang robust untuk melindungi kerahasiaan data (Yuniati, 2022). AES bekerja dengan mengubah data plaintext menjadi ciphertext yang tidak dapat dibaca tanpa kunci dekripsi yang sesuai, sehingga melindungi informasi seperti password saat disimpan di basis data. Algoritma ini telah diadopsi secara luas di berbagai sektor, mulai dari keamanan data umum hingga pengamanan transaksi e-commerce (Mustika, 2020). Dengan demikian, AES memberikan landasan kriptografi yang dapat diandalkan untuk membangun sistem penyimpanan kredensial yang aman.

Namun, penting untuk disadari bahwa enkripsi saja tidak cukup untuk menangkal serangan aktif seperti SQL Injection yang menargetkan proses autentikasi secara real-time. Perlindungan terhadap serangan ini memerlukan lapisan pertahanan tambahan pada level aplikasi, seperti penggunaan parameterized query atau prepared statements yang memisahkan logika SQL dari input pengguna (Fadlullah et al., 2023). Pendekatan ini mencegah penyerang mengeksekusi perintah arbitrer melalui form input, karena input diperlakukan murni sebagai data. Kombinasi antara enkripsi untuk melindungi data saat diam (data at rest) dan praktik pengkodean aman untuk menangkal injeksi merupakan bentuk strategi *defense in depth* yang direkomendasikan. Penelitian ini berangkat dari kesenjangan tersebut dengan menggabungkan kedua aspek untuk dievaluasi efektivitasnya secara empiris.

Berdasarkan latar belakang di atas, penelitian ini bertujuan untuk mengimplementasikan algoritma Rijndael (AES) pada sistem login aplikasi pemesanan makanan berbasis web dan menguji ketahanannya terhadap serangan In-band SQL Injection. Fokus penelitian adalah untuk memvalidasi apakah kombinasi enkripsi dan parameterized query dapat menciptakan sistem autentikasi yang tangguh dan tahan terhadap upaya eksploitasi umum. Hasil penelitian diharapkan dapat memberikan bukti empiris serta panduan praktis bagi pengembang dalam menerapkan keamanan berlapis. Selain itu, penelitian ini juga diharapkan dapat berkontribusi pada literatur keamanan siber dengan mengeksplorasi interaksi antara kriptografi dan teknik mitigasi injeksi pada konteks aplikasi web yang spesifik.

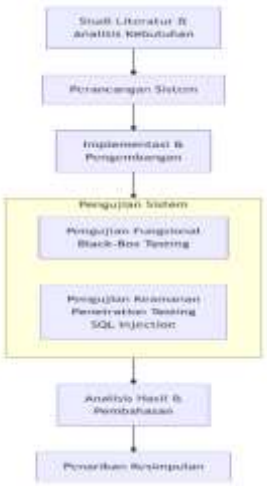
2. Metode Penelitian

Bagian ini menguraikan kerangka kerja metodologis yang diterapkan untuk mencapai tujuan penelitian, yaitu menguji efektivitas metode Rijndael (AES) dan parameterized query

dalam mengamankan sistem login terhadap serangan SQL Injection. Penelitian ini menggunakan pendekatan eksperimental dengan membangun dan menguji prototipe aplikasi pemesanan makanan berbasis web secara sistematis. Tahapan metodologis mencakup perancangan sistem, implementasi enkripsi AES, serta pengujian fungsional dan penetrasi untuk memvalidasi keamanan. Semua proses pengembangan dan pengujian dilakukan dalam lingkungan terkendali menggunakan perangkat lunak sumber terbuka seperti Flask, MariaDB, dan PyCryptodome. Dengan demikian, metodologi yang diterapkan dirancang untuk memastikan bahwa data dan temuan yang dihasilkan dapat dipertanggungjawabkan, terukur, dan relevan dengan konteks keamanan aplikasi web kontemporer.

2.1. Objek Penelitian

Penelitian ini menggunakan objek berupa prototipe aplikasi pemesanan makanan berbasis web yang dikembangkan khusus untuk menguji integrasi keamanan kriptografi dan pertahanan aplikasi. Prototipe ini beroperasi pada lingkungan server lokal (localhost) dengan arsitektur klien-server standar, mensimulasikan fungsionalitas inti sistem pemesanan makanan yang berfokus pada proses autentikasi. Komponen utama objek penelitian meliputi: (1) Halaman login sebagai titik masuk utama dan target uji serangan; (2) Basis data pengguna yang menyimpan kredensial terenkripsi menggunakan Rijndael (AES); dan (3) Mekanisme autentikasi sisi server yang menggabungkan proses enkripsi-dekripsi dengan parameterized query untuk validasi kredensial. Pemilihan prototipe khusus memungkinkan kontrol penuh terhadap variabel keamanan dan memastikan replikabilitas eksperimen (Pakpahan & Prayino, 2021). Penelitian ini dilaksanakan melalui enam tahapan berurutan yang dirancang untuk memastikan validitas dan reliabilitas temuan, sebagaimana diilustrasikan dalam Gambar 3.1. Tahap pertama adalah studi literatur dan analisis kebutuhan, di mana dilakukan pengkajian mendalam terhadap pustaka terkait kriptografi Rijndael, teknik SQL Injection, dan praktik keamanan web untuk membangun landasan teoritis (Andriyanto et al., 2020). Tahap kedua adalah perancangan sistem, mencakup desain skema basis data, antarmuka pengguna, dan alur logika autentikasi yang menggabungkan enkripsi AES. Selanjutnya, tahap implementasi dan pengembangan mengubah desain menjadi kode fungsional menggunakan Python Flask, MariaDB, dan pustaka PyCryptodome untuk enkripsi. Tahap inti adalah pengujian sistem, yang dibagi menjadi pengujian fungsional (black-box) dan pengujian keamanan (penetration testing) dengan menyuntikkan payload SQL Injection. Hasil pengujian kemudian dianalisis secara kualitatif pada tahap analisis hasil dan pembahasan untuk mengevaluasi respons sistem. Tahap terakhir adalah penarikan kesimpulan, di mana temuan disintesis untuk menjawab rumusan masalah penelitian.



Gambar1. Tahapan Penelitan

2.2. Pengumpulan Data

Pengumpulan data dilakukan melalui dua teknik utama: observasi langsung dan dokumentasi sistematis. Selama pengujian fungsional dan penetrasi, peneliti mengamati secara langsung respons sistem, termasuk pesan error, keberhasilan login, dan perilaku tak terduga. Semua observasi dicatat dalam protokol pengujian yang terstruktur. Selanjutnya, setiap skenario pengujian didokumentasikan secara detail, meliputi payload serangan yang digunakan, tangkapan layar (screenshot) antarmuka dan hasil, serta log respons sistem. Dokumentasi ini berfungsi sebagai bukti empiris dan bahan analisis kualitatif untuk menilai efektivitas pertahanan. Pendekatan ini memastikan transparansi dan akuntabilitas proses pengujian, sekaligus memfasilitasi analisis pola kegagalan atau keberhasilan sistem (Ali et al., 2011).

2.3. Analisis Data

Data yang dikumpulkan dianalisis menggunakan metode analisis deskriptif kualitatif dengan fokus pada interpretasi naratif terhadap hasil pengujian keamanan. Efektivitas sistem dinilai berdasarkan dua kriteria utama: (1) kemampuan sistem untuk secara konsisten menolak semua percobaan akses tidak sah yang dilakukan melalui payload SQL Injection, dan (2) ketiadaan kebocoran informasi sensitif melalui pesan error atau respons sistem. Analisis dilakukan dengan membandingkan hasil antara halaman login yang aman (menggunakan parameterized query) dan yang rentan (menggunakan konkatenasi string), sehingga dapat mengisolasi kontribusi setiap lapisan pertahanan. Pendekatan ini memungkinkan penilaian holistik terhadap strategi defense in depth yang diusulkan (Fadlullah et al., 2023).

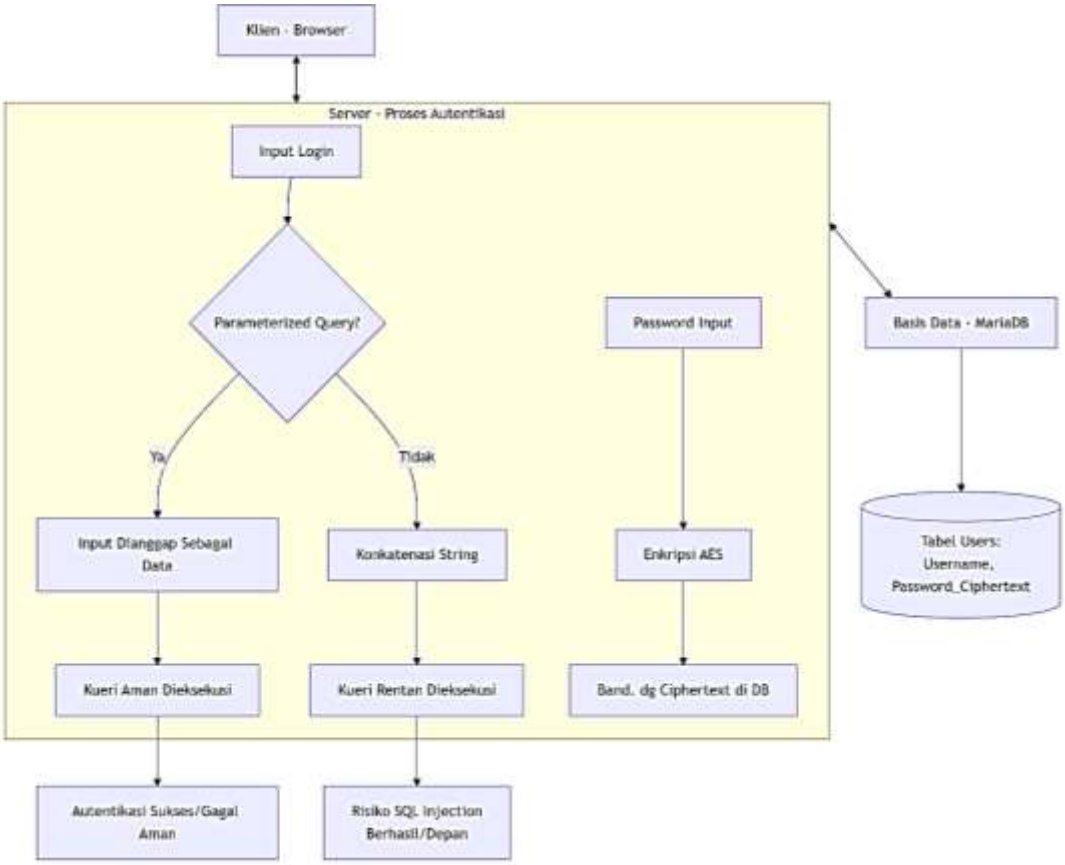
2.4. Alat dan Bahan

Penelitian ini menggunakan seperangkat perangkat keras dan lunak yang spesifik untuk mendukung pengembangan, implementasi, dan pengujian. Spesifikasi perangkat keras utama adalah laptop dengan prosesor Intel® Core™ i7-7700HQ, memori 8 GB, dan sistem operasi 64-bit. Sementara itu, perangkat lunak yang digunakan meliputi: Python 3.13.2 dan framework Flask sebagai dasar pengembangan aplikasi; XAMPP 8.2.12 untuk menyediakan server Apache dan basis data MariaDB; PyCryptodome untuk implementasi enkripsi AES; Visual Studio Code sebagai editor kode; serta phpMyAdmin untuk manajemen basis data. Pemilihan alat-alat tersebut didasarkan pada kompatibilitas, dukungan komunitas, dan kesesuaian dengan kebutuhan pengembangan aplikasi web yang aman (Rizky et al., 2024).

2.5. Arsitektur yang Diimplementasikan

Gambar 2. Merupakan ilustrasi arsitektur klien-server dan alur logika autentikasi yang diterapkan dalam prototipe penelitian. Sistem terdiri dari tiga komponen utama: Klien (browser pengguna), Server Web yang dibangun dengan framework Flask, dan Basis Data MariaDB. Proses dimulai ketika pengguna mengirimkan kredensial login melalui browser ke server. Pada sisi server, input pengguna diproses melalui dua jalur kritis yang berbeda: jalur pertama menggunakan parameterized query yang memperlakukan input sebagai data murni, sedangkan jalur kedua menggunakan konkatenasi string yang rentan terhadap injeksi. Secara paralel, password yang diinputkan pengguna dienkripsi menggunakan algoritma AES sebelum dibandingkan dengan ciphertext yang tersimpan dalam basis data. Arsitektur ini sengaja dirancang untuk membandingkan dua pendekatan keamanan secara langsung dalam lingkungan yang terkendali. Arus logika dalam gambar tersebut menyoroti perbedaan mendasar antara mekanisme pertahanan yang diuji. Pada jalur aman, input username langsung diteruskan sebagai parameter terpisah ke kueri SQL, sehingga mencegah interpretasi input sebagai bagian dari perintah yang dapat dieksekusi yang menghasilkan autentikasi sukses atau gagal yang aman. Sebaliknya, pada jalur rentan (konkatenasi string), input digabungkan langsung ke dalam string kueri, memungkinkan payload SQL Injection mengubah logika kueri dan berpotensi menyebabkan akses tidak sah atau kebocoran data. Dengan mempertahankan komponen enkripsi AES yang sama pada kedua jalur, gambar ini menegaskan bahwa keamanan sistem login bergantung pada lapisan parameterized query, sedangkan enkripsi berfungsi sebagai lapisan pelindung tambahan untuk data yang disimpan.

BHATARA: Jurnal Multidisiplin
Vol. 3, No.2, April 2026, pp. 54-61, ISSN: 2476-9843



Gambar 2. Desain Arsitektur

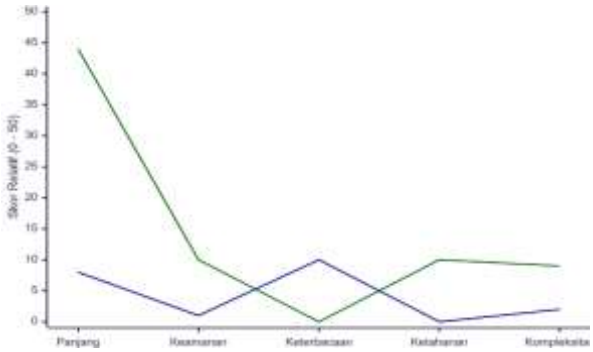
3. Hasil dan Pembahasan

Bab ini menyajikan temuan empiris dari implementasi sistem serta analisis mendalam terhadap hasil pengujian keamanan yang dilakukan. Hasil yang dipaparkan mencakup dua aspek utama, yaitu keberhasilan implementasi teknikal algoritma Rijndael (AES) untuk enkripsi password dan kinerja sistem dalam menghadapi serangan SQL Injection yang disimulasikan. Data yang disajikan meliputi output pengujian fungsional untuk memvalidasi kebenaran kerja sistem, serta hasil penetrasi yang membandingkan secara langsung antara halaman login yang aman dan yang rentan. Pembahasan selanjutnya akan mengaitkan temuan-temuan ini dengan teori keamanan siber dan penelitian terdahulu, sekaligus mengevaluasi kontribusi strategi *defense in depth* yang diterapkan. Dengan demikian, bab ini tidak hanya mendokumentasikan hasil, tetapi juga memberikan interpretasi kritis terhadap efektivitas metode Rijndael dan praktik pengkodean aman dalam konteks perlindungan sistem login aplikasi web.

3.1. Hasil Implementasi dan Pengujian Fungsional

Implementasi algoritma Rijndael (AES) pada prototipe aplikasi berhasil dilakukan dengan menggunakan pustaka PyCryptodome dalam mode operasi CBC (Cipher Block Chaining) dan Initialization Vector (IV) acak. Hasil pengujian fungsional (black-box testing) menunjukkan bahwa semua skenario operasional dasar sistem berjalan sesuai ekspektasi. Pada proses registrasi, password pengguna berhasil dienkripsi dan disimpan sebagai ciphertext Base64 dalam basis data, seperti terlihat pada Gambar 3.1 yang menampilkan perbandingan antara plaintext "password123" dengan ciphertext yang dihasilkan. Proses login dengan kredensial valid berhasil mendekripsi dan membandingkan password, sementara upaya login dengan password salah atau username duplikat ditolak dengan pesan error yang sesuai. Hasil ini membuktikan bahwa integrasi modul kriptografi tidak mengganggu fungsionalitas inti aplikasi dan mampu menjaga kerahasiaan data at rest.

BHATARA: Jurnal Multidisiplin
Vol. 3, No.2, April 2026, pp. 54-61, ISSN: 2476-9843



Gambar 3. Visualisasi Hasil Enkripsi AES-CBC perbandingan Password Plaintext dan Ciphertext yang Disimpan dalam Basis Data

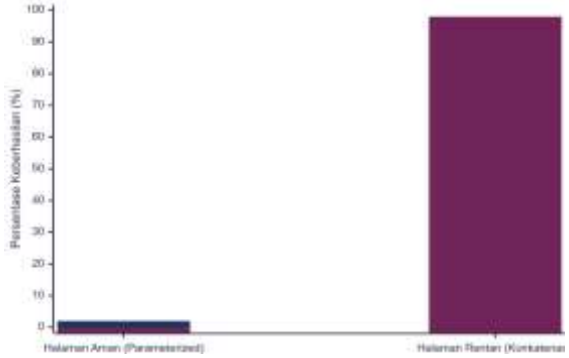
3.2. Hasil Pengujian Keamanan terhadap SQL Injection

Pengujian penetrasi dengan payload SQL Injection klasik (' OR '1'='1'--) menghasilkan dua outcome yang bertolak belakang, sebagaimana dirangkum dalam Tabel 4.1. Pada halaman login aman yang menggunakan parameterized query, sistem secara konsisten menolak akses dengan menampilkan pesan "Username atau password salah!", tanpa memedulikan variasi payload yang diinjeksikan. Sebaliknya, pada halaman login rentan yang menggunakan konkatenasi string, payload yang sama berhasil melewati autentikasi dan memberikan akses tidak sah ke akun pertama dalam basis data. Hasil ini secara eksplisit membuktikan bahwa kerentanan SQL Injection tidak dapat diatasi hanya dengan enkripsi, tetapi sangat bergantung pada sanitasi input di level aplikasi.

Tabel 1. Hasil Pengujian Keamanan dengan Payload SQL Injection ' OR '1'='1'--

Halaman Login	Metode Query	Payload Input	Hasil Autentikasi	Status Keamanan
/login (Aman)	Parameterized	admin' OR '1'='1'--	Gagal, pesan error standar	Lolos Uji
/login_vulnerable (Rentan)	Konkatenasi String	admin' OR '1'='1'--	Berhasil, akses tidak sah	Gagal Uji

Hasil tabel menunjukkan bahwa halaman login dengan parameterized query menolak payload SQL Injection sehingga autentikasi gagal dan sistem tetap aman, sedangkan halaman rentan berbasis konkatenasi string menerima input berbahaya, memberikan akses tidak sah, membuktikan pentingnya penggunaan query terparameter untuk keamanan. Adapun Gambar 4. adalah visualisasi perbandingan tingkat keberhasilan serangan antara kedua halaman tersebut, mengonfirmasi efektivitas absolut parameterized query sebagai mekanisme pertahanan.



Gambar 4. Hasil perbandingan keberhasilan Parameterized Query (0%) dan Konkatenasi String (100%) dalam menangkal serangan Hasil perbandingan SQL Injection

3. Hasil dan pembahasan

Hasil penelitian ini memperkuat konsep *defense in depth* dengan menunjukkan bahwa enkripsi Rijndael dan parameterized query berfungsi sebagai dua lapisan pertahanan yang saling melengkapi dengan peran berbeda. Enkripsi AES berperan sebagai lapisan perlindungan pasif yang mengamankan data saat disimpan (*data at rest*), sehingga bahkan jika penyerang berhasil mengekstrak isi basis data melalui vektor lain (seperti backup yang tidak diamankan), informasi kredensial tetap tidak terbaca tanpa kunci dekripsi (Yuniati, 2022). Sementara itu, parameterized query berperan sebagai lapisan pertahanan aktif yang secara real-time mencegah eksekusi kode berbahaya pada titik masuk aplikasi (Fadlullah et al., 2023). Temuan bahwa enkripsi saja tidak mampu menghalangi SQL Injection sejalan dengan penelitian Andriyanto dkk. (2020) yang menekankan bahwa mitigasi injeksi harus dilakukan pada lapisan pemrosesan query. Dinamika interaksi antara kedua lapisan keamanan ini juga mengungkap prinsip penting dalam arsitektur keamanan informasi: setiap lapisan bertugas menutupi celah potensial yang mungkin terlewat oleh lapisan lain. Dalam skenario aplikasi dunia nyata, kerentanan dapat muncul dari faktor di luar kode aplikasi inti, seperti miskonfigurasi server atau kerentanan pada komponen pihak ketiga. Dengan mengadopsi pendekatan berlapis, risiko keseluruhan sistem dapat diminimalkan karena penyerang harus menembus multiple barrier yang independen. Penelitian ini dengan demikian memberikan validasi empiris bahwa kombinasi antara kriptografi kuat dan praktik pengembangan aman merupakan strategi yang lebih tangguh daripada mengandalkan satu mekanisme tunggal, sekaligus menjawab rumusan masalah tentang efektivitas metode Rijndael dalam konteks uji keamanan SQL Injection secara holistik.

4. Kesimpulan

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, dapat disimpulkan bahwa implementasi algoritma Rijndael (AES) berhasil berfungsi sebagai lapisan pertahanan pasif yang efektif untuk melindungi kerahasiaan data kredensial pengguna dalam sistem login. Enkripsi AES terbukti mampu mengamankan password yang disimpan dalam basis data (data at rest) dengan mengubahnya menjadi ciphertext yang tidak dapat dibaca tanpa kunci dekripsi yang sesuai, sehingga memitigasi risiko kebocoran data dari sisi penyimpanan. Namun, temuan kritis penelitian ini menunjukkan bahwa enkripsi semata tidak cukup untuk mencegah serangan SQL Injection secara real-time, karena serangan ini mengeksploitasi celah pada lapisan pemrosesan query aplikasi, bukan pada data yang telah terenkripsi. Secara lebih luas, penelitian ini berhasil memvalidasi bahwa strategi keamanan berlapis (defense in depth) yang menggabungkan enkripsi AES dengan praktik pengkodean aman (parameterized query) merupakan pendekatan yang esensial untuk membangun sistem login yang tangguh. Parameterized query berperan sebagai pertahanan aktif yang secara konsisten berhasil menggagalkan seluruh percobaan serangan SQL Injection, sementara enkripsi AES berfungsi sebagai benteng terakhir jika terjadi pelanggaran data. Dengan demikian, integrasi kedua mekanisme tersebut tidak hanya menjawab rumusan masalah penelitian, tetapi juga memberikan panduan praktis bagi pengembang untuk mengadopsi pendekatan holistik dalam mengamankan aplikasi web kontemporer.

Daftar Pustaka

1. Ali, A. B. M., Shakhatreh, A. Y. I., Abdullah, M. S., & Alostad, J. (2011). SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks. *Procedia Computer Science*, 3, 453–458. https://doi.org/10.1016/j.procs.2010.12.076
2. Andriyanto, R., Khairjal, K., & Satria, D. (2020). Penerapan Kriptografi AES Class untuk pengamanan URL website dari serangan SQL Injection. *JURNAL UNITEK*, 13(1), 34–48. https://doi.org/10.52072/unitek.v13i1.153
3. Aprilia, N. F., Mafa, D., Muchtar, A. R., Rohim, K. A. A., & Firdaus, R. N. I. (2023). Penerapan algoritma AES untuk enkripsi pada halaman register serta penerapan AES untuk

deskripsi pada halaman login website. *Journal of Informatics Development*, 1(2), 75–82. https://doi.org/10.30741/jid.v1i2.1041

4. Bhaskara, I. M. A., Wiharta, D. M., & Saputra, O. (2020). Perancangan Sistem Penyedia File Sharing dengan Enkripsi URL menggunakan Algoritma Rijndael. *Majalah Ilmiah Teknologi Elektro*, 19(2), 171. https://doi.org/10.24843/MITE.2020.v19i02.P08
5. Fadlullah, F., Tahir, M., Bintari, B. P., Dewi, M. L., Ilmy, M. F., Syafi', S., & Ardiansyah, R. (2023). Implementasi Algoritma AES pada Autentikasi Login Sistem Informasi. *Jurnal Bintang Pendidikan Indonesia*, 1(2), 251–263. https://doi.org/10.55606/jubpi.v1i2.1420
6. Mustika, L. (2020). Implementasi Algoritma AES Untuk Pengamanan Login Dan Data Customer Pada E-Commerce Berbasis Web. *JURIKOM (Jurnal Riset Komputer)*, 7(1), 148. https://doi.org/10.30865/jurikom.v7i1.1943
7. Pakpahan, A. V., & Prayino, N. F. (2021). Implementasi Algoritma Rijndael Untuk Keamanan Login (Studi Kasus: Perangkat Lunak Keuangan Pemberian Tunjangan Di Kantor Kopertis Wilayah IV). *Jurnal SIMETRIS*, 12(1). https://doi.org/10.24176/simet.v12i1.4442
8. Prajuhana Putra, A., Herfina, H., Maryana, S., & Setiawan, A. (2020). Implementasi Algoritma Aes (Advanced Encryption Standard) Rijndael Pada Aplikasi Keamanan Data. *JIPETIK:Jurnal Ilmiah Penelitian Teknologi Informasi & Komputer*, 1(2), 46–51. https://doi.org/10.26877/jipetik.v1i2.4599
9. Rizky, P. A., Soim, S., & Sholihin, S. (2024). Implementasi Algoritma Kriptografi AES CBC untuk Keamanan Komunikasi Data pada Hardware. *Journal RESISTOR (Rekayasa Sistem Komputer)*, 7(2), 71–78. https://doi.org/10.31598/jurnalresistor.v7i2.1650
10. Yuniati, T. (2022). Pengembangan Aplikasi Penyandian Data Menggunakan Algoritma Rijndael. *Jurnal TIMES*, 11(2), 25–33. https://doi.org/10.51351/jtm.11.2.2022679