Simpatik: Jurnal Sistem Informasi dan Informatika
Volume 4 No. 2 Desember 2024

Peningkatan Keamanan dan Privasi Aplikasi *Website DNA Sequencing* Menggunakan Enkripsi AES 256 dan *Query Parameterization*

Diash Firdaus[1], Muhammad Zufar Dafy[2]

[1,2] Institut Teknologi Nasional Bandung, Bandung
[1,2] Jl. Khp Hasan Mustopa No.23, Neglasari, Kec. Cibeunying Kaler, Kota Bandung, Indoneisa
e-mail: [1]diash@itenas.ac.id, [2]dafyluck@gmail.com

Artikel Info : Diterima : 27-06-2024 | Direvisi : 07-10-2024 | Disetujui : 01-12-2024

Abstrak - Data DNA *Sequencing* merupakan informasi yang sangat sensitif dan esensial dalam berbagai aplikasi biomedis, termasuk penelitian genetika, diagnosis penyakit, dan pengembangan terapi medis. Keamanan dan privasi data ini menjadi sangat penting untuk mencegah penyalahgunaan dan kebocoran informasi yang dapat berdampak serius pada individu. Penelitian ini bertujuan untuk meningkatkan keamanan dan privasi pada aplikasi *website* DNA *Sequencing* melalui implementasi enkripsi AES-256 dan *query parameterization*. Enkripsi AES-256 digunakan untuk mengenkripsi data DNA sensitif sebelum disimpan dalam basis data, memastikan bahwa data tersebut terlindungi dari akses tidak sah dan kebocoran informasi. Selain itu, *query parameterization* diterapkan untuk melindungi aplikasi dari serangan injeksi SQL dengan memisahkan data dari pernyataan SQL, sehingga mencegah eksekusi perintah yang tidak diinginkan akibat *input* pengguna yang berbahaya. Hasil dari implementasi ini menunjukkan peningkatan signifikan dalam keamanan dan privasi aplikasi web DNA *Sequencing*. Data yang dienkripsi menjadi tidak dapat dibaca tanpa kunci deskripsi yang sesuai, dan *query parameterization* mencegah eksploitasi melalui injeksi SQL. Penelitian ini memiliki kontribusi penting dalam pengembangan aplikasi web yang aman, khususnya dalam konteks aplikasi biomedis yang menangani data sensitif seperti DNA. Rekomendasi untuk penelitian lebih lanjut meliputi eksplorasi metode enkripsi dan teknik keamanan tambahan serta pelaksanaan audit keamanan berkala dan pengujian penetrasi untuk memastikan sistem tetap terlindungi dari ancaman yang terus berkembang. Dengan demikian, penerapan enkripsi AES-256 dan *query parameterization* terbukti efektif dalam meningkatkan keamanan dan privasi aplikasi web DNA Sequencing.

Kata Kunci : DNA *Sequencing*, SQL *Injection*, AES 256

Abstracts *- DNA Sequencing data is highly sensitive and essential in various biomedical applications, including genetic research, disease diagnosis, and medical therapy development. Ensuring the security and privacy of this data is crucial to prevent misuse and information leakage, which can have serious implications for individuals. This study aims to enhance the security and privacy of a DNA Sequencing web application through the implementation of AES-256 encryption and query parameterization. AES-256 encryption is used to encrypt sensitive DNA data before storing it in the database, ensuring that the data is protected from unauthorized access and information leakage. Additionally, query parameterization is applied to protect the application from SQL injection attacks by separating data from SQL statements, preventing the execution of unintended commands due to malicious user input. The results of this implementation demonstrate a significant increase in the security and privacy of the DNA Sequencing web application. Encrypted data becomes unreadable without the appropriate decryption key, and query parameterization prevents exploitation through SQL injection. This research contributes significantly to the development of secure web applications, particularly in the context of biomedical applications handling sensitive data like DNA. Recommendations for further research include exploring additional encryption methods and security techniques, as well as conducting regular security audits and penetration testing to ensure the system remains protected against evolving threats. Thus, the application of AES-256 encryption and query parameterization has proven effective in enhancing the security and privacy of the DNA Sequencing web application.*

Keywords : *DNA Sequencing, SQL Injection, AES 256*

PENDAHULUAN

Dalam era digital yang semakin berkembang, keamanan dan privasi data menjadi aspek krusial dalam pengembangan aplikasi *website*, terutama yang berkaitan dengan informasi sensitif seperti DNA *sequencing* (Bonomi et al., 2018). Aplikasi *website* DNA *sequencing* memiliki peran penting dalam penelitian genetika dan medis, namun juga menghadapi tantangan besar dalam hal perlindungan data yang sangat sensitif dan pribadi (Kuo et al., 2022). Salah satu ancaman utama terhadap keamanan aplikasi *website* adalah serangan siber yang dapat mengakses atau memanipulasi data tanpa izin (Vinatzer et al., 2019). Hal ini menjadi semakin penting mengingat data DNA *sequencing* mengandung informasi genetik yang sangat pribadi dan dapat disalahgunakan jika jatuh ke tangan yang tidak bertanggung jawab (Fayans et al., 2020).

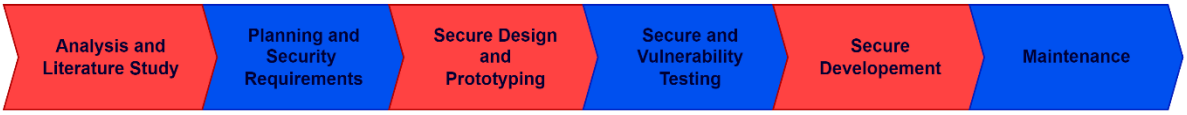
Enkripsi data merupakan salah satu metode yang efektif untuk meningkatkan keamanan informasi sensitif. *Advanced Encryption Standard* (AES) telah menjadi standar enkripsi yang diakui secara luas karena keamanannya yang tinggi (Isminarti et al., 2023). Khususnya, AES-256 menawarkan tingkat keamanan yang sangat kuat dengan menggunakan kunci 256-bit, menjadikannya pilihan yang ideal untuk mengamankan data sensitif seperti hasil DNA *sequencing* (Matta et al., 2021). Selain enkripsi data, keamanan aplikasi *website* juga perlu memperhatikan aspek keamanan pada sisi server, terutama dalam hal penanganan *query database* (Alodaynan & Alanazi, 2021). *Query parameterization* merupakan teknik yang efektif untuk mencegah serangan injeksi SQL, yang merupakan salah satu ancaman serius terhadap keamanan aplikasi web.

Implementasi enkripsi AES-256 pada aplikasi *website* DNA *sequencing* dapat memberikan jaminan privasi *end-to-end* dan integritas data antara server dan klien. Hal ini sangat penting mengingat sensitifitas data genetik yang diproses dan disimpan dalam aplikasi tersebut. Penggunaan AES-256 juga memungkinkan enkripsi data sebelum penyimpanan atau transmisi, sehingga meminimalkan risiko kebocoran informasi meskipun terjadi pelanggaran keamanan pada level jaringan atau penyimpanan (Isminarti et al., 2023). Meskipun ada sedikit penurunan performa akibat proses enkripsi, manfaat keamanan yang diperoleh jauh lebih besar dibandingkan dengan risiko kebocoran data sensitif (Ayuni et al., 2020)(Chandra et al., 2019). Di sisi lain, *Query Parameterization* membantu mencegah serangan injeksi SQL dengan memisahkan logika *query* dari data input pengguna. Teknik ini sangat penting dalam konteks aplikasi DNA *sequencing*, di mana *query database* sering digunakan untuk mengambil dan memproses data genetik (Manmadhan, 2012).

Dengan meningkatnya kesadaran akan pentingnya privasi data dan regulasi perlindungan data yang semakin ketat, penelitian ini juga memiliki relevansi tinggi dengan kebutuhan industri dan regulasi terkini. Implementasi solusi keamanan yang kuat seperti yang diusulkan dalam penelitian ini dapat membantu aplikasi *website* DNA *sequencing* memenuhi standar keamanan dan privasi yang diperlukan, serta meningkatkan kepercayaan pengguna terhadap layanan yang ditawarkan. (Scheibner et al., 2020). Kombinasi enkripsi AES-256 dan *query parameterization* menciptakan lapisan keamanan ganda yang melindungi data baik saat disimpan maupun saat diakses. Pendekatan ini tidak hanya meningkatkan keamanan data, tetapi juga membangun kepercayaan pengguna terhadap aplikasi, yang sangat penting dalam bidang sensitif seperti analisis genetik. Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi efektivitas penggunaan enkripsi AES-256 dan *query parameterization* dalam meningkatkan keamanan dan privasi aplikasi *website* DNA *sequencing*. Hasil penelitian diharapkan dapat memberikan kontribusi signifikan dalam pengembangan standar keamanan untuk aplikasi yang menangani data genetik sensitif.

METODE PENELITIAN

Metode yang digunakan dalam melakukan penelitian ini adalah metode *Secure Software Development Life Cycle* (SSDLC) yang dinilai memiliki kemampuan untuk mengurangi resiko serangan siber secara signifikan (Hasanul Fahmi, 2018). Dengan tahapan yang bisa dilihat pada Gambar 1. Pada tahap pengembangan Aplikasi *website* akan dibuat dengan menggunakan *Query Parameterization* untuk menangkal serangan SQL *injection* dan AES 256 untuk melakukan privasi pada data yang diinputkan ke dalam *database*.



Sumber : Hasil Penelitian (2024)

Gambar 1. Metode SSDLC

1. *Analysis and Literature Study*

Keamanan adalah salah satu faktor utama dalam pengembangan sebuah situs web. Tantangan dalam menjaga keamanan web sangat kompleks, melibatkan mekanisme yang dirancang untuk mencegah akses dan modifikasi data secara tidak sah oleh pengguna yang tidak terotorisasi. Salah satu metode pemantauan yang digunakan untuk memastikan keamanan sistem adalah *Intrusion Detection* System (IDS) (Febriansyah et al., 2023). IDS berfungsi untuk mendeteksi ancaman yang ada. Selanjutnya, tindak lanjut terhadap ancaman ini dilakukan dengan menggunakan *Intrusion Prevention System* (IPS), yang berperan sebagai pengaman untuk

memblokir alamat IP pengguna yang mencoba melakukan serangan injeksi, sehingga mereka tidak dapat mengakses situs web tersebut (Almaj Duddin & Fitrani, 2021). IDS sendiri merupakan metode atau alat yang diimplementasikan dalam suatu sistem untuk mendeteksi dan memantau keamanan sistem tersebut. IDS berfungsi untuk memonitoring berbagai peristiwa yang terjadi dalam sebuah sistem komputer atau jaringan, serta menganalisisnya guna mengidentifikasi adanya tanda-tanda mencurigakan. Tanda-tanda ini dapat mengindikasikan potensi ancaman terhadap kebijakan keamanan komputer yang telah diterapkan (Firdaus et al., 2020).

a. SQL Injection

SQL Injection adalah salah satu bentuk serangan terhadap aplikasi web yang memungkinkan penyerang untuk mengintervensi kueri SQL yang dieksekusi oleh aplikasi. Serangan ini memanfaatkan celah keamanan dalam input pengguna untuk mengeksekusi pernyataan SQL yang tidak diinginkan, yang dapat menyebabkan pengungkapan informasi sensitif, modifikasi data, atau bahkan pengambilalihan kontrol penuh atas sistem basis data (Manmadhan, 2012). SQL *Injection* terjadi ketika penyerang memasukkan atau "menyuntikkan" kode SQL berbahaya ke dalam kueri melalui input pengguna yang tidak divalidasi atau tidak dilindungi dengan baik. Serangan ini bekerja dengan cara mengubah logika kueri SQL yang dijalankan oleh aplikasi, sehingga memungkinkan penyerang untuk memperoleh akses tanpa izin atau melakukan tindakan yang tidak diotorisasi (Odeh & Taleb, 2024). Serangan SQL *Injection* dapat mengakibatkan berbagai kerugian serius. Sekali penyerang berhasil mengeksploitasi celah ini, mereka dapat mencuri informasi sensitif seperti data pengguna, informasi kartu kredit, dan data perusahaan yang bersifat rahasia. Selain itu, penyerang juga dapat menghapus atau mengubah data yang ada, mengakibatkan kerugian besar bagi organisasi dan menurunkan kepercayaan pengguna terhadap keamanan aplikasi tersebut (Riyanti et al., 2024).

b. Man In the Middle Attack (MITM)

Man-in-the-Middle (MITM) adalah salah satu jenis serangan siber yang paling umum dan berbahaya. Serangan ini melibatkan penyusup yang memposisikan dirinya di antara komunikasi dua pihak, dengan tujuan mencuri informasi, memanipulasi data, atau mengganggu komunikasi. MITM dapat terjadi dalam berbagai konteks, termasuk jaringan nirkabel, komunikasi email, dan pertukaran data di internet. MITM terjadi ketika seorang penyerang berhasil memasuki jalur komunikasi antara dua pihak yang sah tanpa terdeteksi. Penyerang dapat menguping (*eavesdropping*), memodifikasi pesan yang dikirimkan, atau bahkan menyamar sebagai salah satu pihak yang terlibat dalam komunikasi. Serangan ini sering kali memanfaatkan kerentanan dalam protokol komunikasi atau kelemahan dalam sistem keamanan yang ada (Auliafitri et al., 2024).

c. Advanced Encryption Standard 256 (AES 256)

Advanced Encryption Standard (AES) 256-bit adalah salah satu algoritma enkripsi simetris yang diakui dan digunakan secara luas untuk melindungi data sensitif. AES dikembangkan oleh Vincent Rijmen dan Joan Daemen, dan dipilih sebagai standar enkripsi oleh *National Institute of Standards and Technology* (NIST) pada tahun 2001. AES memiliki ukuran kunci yang bervariasi, yaitu 128, 192, dan 256 bit, dengan AES 256-bit menjadi yang paling kuat di antara ketiganya. AES 256-bit menggunakan kunci sepanjang 256 bit, yang berarti terdapat 2^{256} kemungkinan kunci yang dapat dipilih, membuatnya sangat sulit untuk dipecahkan melalui *brute-force*. Algoritma ini bekerja dengan memproses blok data berukuran 128 bit dalam beberapa putaran transformasi yang melibatkan substitusi, pengacakan, dan penggabungan elemen data (Aldianto & Wibowo, 2023)(Irawan et al., 2023)

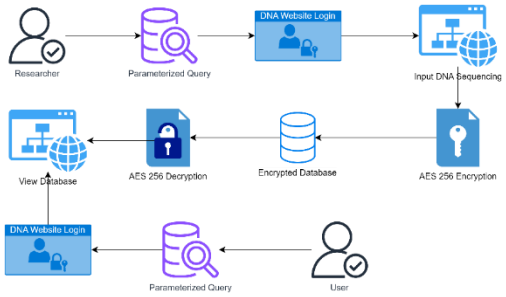
d. Query parameterization

Query parameterization adalah teknik yang digunakan dalam pengembangan aplikasi untuk melindungi terhadap serangan SQL *Injection*, salah satu ancaman keamanan teratas bagi aplikasi web. Teknik ini melibatkan penggunaan parameter yang dipisahkan dari kueri SQL, sehingga input pengguna tidak dapat mengubah struktur sintaksis dari kueri SQL tersebut. Dalam *query parameterization*, kueri SQL dibuat dengan *placeholder* untuk parameter, dan nilai parameter ini diberikan secara terpisah. Hal ini memastikan bahwa *input* pengguna diperlakukan secara murni sebagai data dan bukan sebagai bagian dari perintah SQL yang dapat dieksekusi (Yulianingsih, 2016).

e. Planning and Security Design

Pada Tahap *Planning*, penelitian ini menggabungkan metode *Parameterize Query* dan AES 256 dalam membangun Web, untuk desain dari *website* yang akan dibangun bisa dilihat pada Gambar 2. Pada Gambar 2 terdapat 2 akun dengan nama *Researcher* dan *user*. *Researcher* akan berlaku sebagai akun yang melakukan

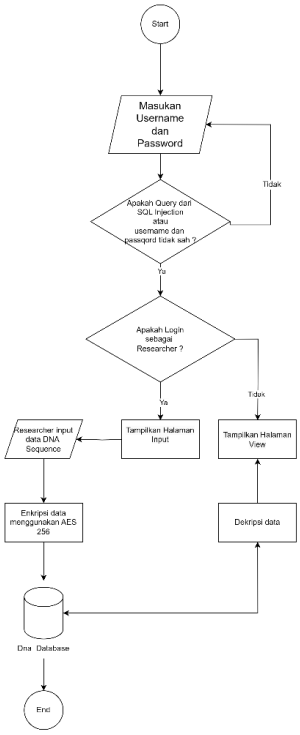
upload data DNA *Sequencing* sedangkan Akun *user* adalah akun yang hanya bisa melihat web yang bersisi data dari DNA *Sequencing*.



Sumber : Hasil Penelitian (2024)

Gambar 3. *Flow Chart Website* DNA *Sequence*

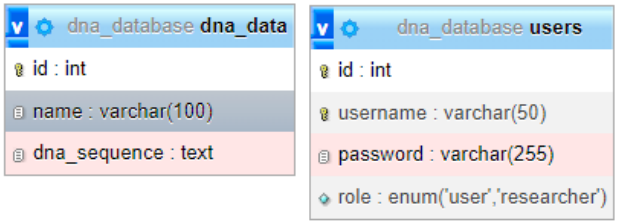
Gambar 3 merupakan *Flow Chart* dari system yang dibuat. Untuk mengatasi serangan *Sql Injection* dan serangan dari MITM (*man in the middle Attack*). Proses dimulai dengan pengguna mengakses sistem dan berniat untuk memasukkan data *username* dan *password*, selanjutnya Sistem melakukan pengecekan terhadap input *query* untuk mendeteksi adanya upaya SQL *Injection*. SQL *Injection* adalah teknik serangan di mana penyerang memasukkan kode berbahaya ke dalam *query* SQL untuk mengakses atau memanipulasi *database* secara tidak sah. Jika terdeteksi serangan SQL *Injection* atau *username* dan *password* tidak sah (Ya), sistem akan mengakhiri proses dan menolak akses. Jika tidak terdeteksi SQL *Injection* (Tidak), proses berlanjut ke langkah berikutnya. Sistem memverifikasi apakah pengguna yang *login* adalah seorang peneliti (*researcher*). Jika pengguna adalah peneliti (Ya), mereka akan diarahkan ke halaman input data. Jika pengguna bukan peneliti (Tidak), mereka akan diarahkan ke halaman *view* yang sesuai dengan hak akses mereka. Akun Peneliti yang telah terverifikasi kemudian dapat memasukkan data urutan DNA (DNA *sequence*) ke dalam sistem melalui halaman input yang disediakan. Data urutan DNA yang dimasukkan oleh akun peneliti akan dienkripsi menggunakan algoritma *Advanced Encryption* Standard (AES) dengan kunci 256-bit. AES-256 adalah salah satu algoritma enkripsi terkuat dan digunakan untuk memastikan bahwa data yang disimpan dalam *database* terlindungi dari akses tidak sah. Data urutan DNA yang telah dienkripsi kemudian disimpan ke dalam *database* untuk keperluan lebih lanjut. Penyimpanan data yang terenkripsi ini memastikan bahwa informasi sensitif tetap aman meskipun terjadi pelanggaran keamanan pada *database*.



Sumber : Hasil Penelitian (2024)

Gambar 3. *Flow Chart Website* DNA *Sequence*

Database yang digunakan pada penelitian ini berjumlah 1 dengan memiliki 2 tabel yaitu *table* dna_Data untuk *input* data DNA *Sequencing* dan *table users* untuk data yang akan digunakan untuk *login*. Gambar 3 merupakan gambar dari ERD yang dibuat.

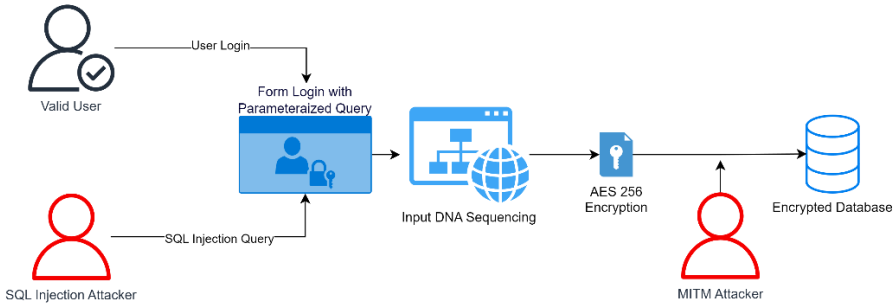


Sumber : Hasil Penelitian (2024)

Gambar 4. ERD *database*

2. *Secure and Vulnerability Testing*

Pada tahap pengujian akan dilakukan dengan menguji halaman login dengan 15 *query* dari Sql *injection* dan melakukan penyerangan dengan asumsi MITM mendapatlan *database* hasil dari *input* DNA *Sequencing*.



Sumber : Hasil Penelitian (2024)

Gambar 4. *Secure* and *Vulnerability testing*

3. *Secure Deployment*

Pada penelitian ini untuk *Framework* Server menggunakan *Flask* (Wijayanto & Susetyo, 2022) dan untuk *Database* menggunakan *MySql*. *Flask* adalah kerangka kerja web mikro untuk *Python* yang telah menjadi sangat populer di kalangan pengembang web dan *MySQL* adalah salah satu sistem manajemen basis data relasional (RDBMS) yang paling populer dan banyak digunakan di dunia.

HASIL DAN PEMBAHASAN

1. Impelemntasi *Query Parameterization*

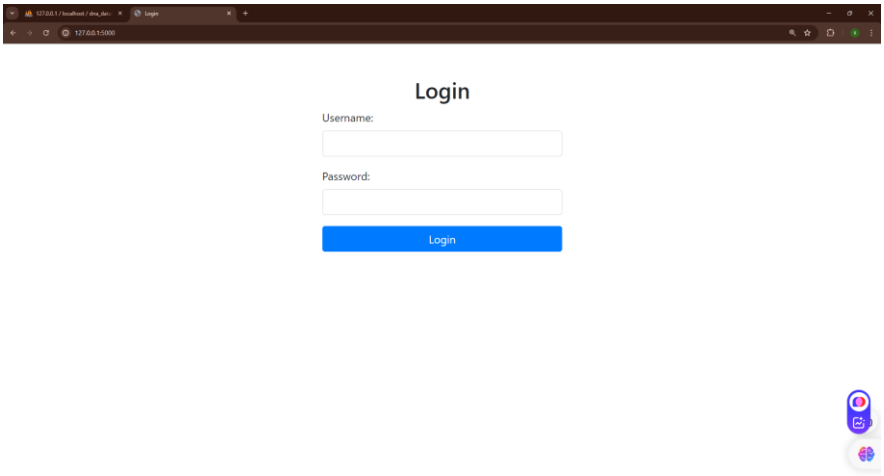
Pada hasil dan pembahasan bisa dilihat pada Gambar 5. yang menunjukan *source code* penggunaan *Query parameterization* yang diimplementasikan pada *Framework Flask*. Pada bagian variabel *query = "SELECT * FROM users WHERE username=%s AND password=%s"* diberikan %s pada *username* dan pada *password* dengan tujuan untuk pemisahan antara kode SQL dan data yang diberikan oleh pengguna. Hal ini memastikan bahwa data pengguna diperlakukan secara eksplisit sebagai data, bukan sebagai bagian dari kode SQL yang dapat dieksekusi. Dengan demikian, teknik ini mencegah eksploitasinya sebagai perintah SQL yang tidak diinginkan oleh pihak yang tidak bertanggung jawab. Selanjutnya, *Placeholder* %s dalam *string query* digantikan dengan nilai aktual dari variabel yang diberikan (misalnya, *username* dan *password*) oleh *driver database*. Proses ini melibatkan pelolosan (*escaping*) karakter-karakter khusus yang mungkin ada dalam input pengguna, sehingga tidak akan diinterpretasikan sebagai bagian dari sintaks SQL.

```python
@app.route('/login', methods=['POST'])
def do_login():
    username = request.form['username']
    password = request.form['password']

    db = get_db_connection()
    cursor = db.cursor(dictionary=True)
    query = "SELECT * FROM users WHERE username=%s AND password=%s"
    cursor.execute(query, (username, password))
    user = cursor.fetchone()

    if user:
        session['user_id'] = user['id']
        session['role'] = user['role']

        if user['role'] == 'researcher':
            return redirect(url_for('input_data'))
        else:
            return redirect(url_for('view_data'))
    else:
        flash('Invalid username or password')
        return redirect(url_for('login'))
```

Sumber : Hasil Penelitian (2024)

Gambar 5. Implemetasi Query parameterization

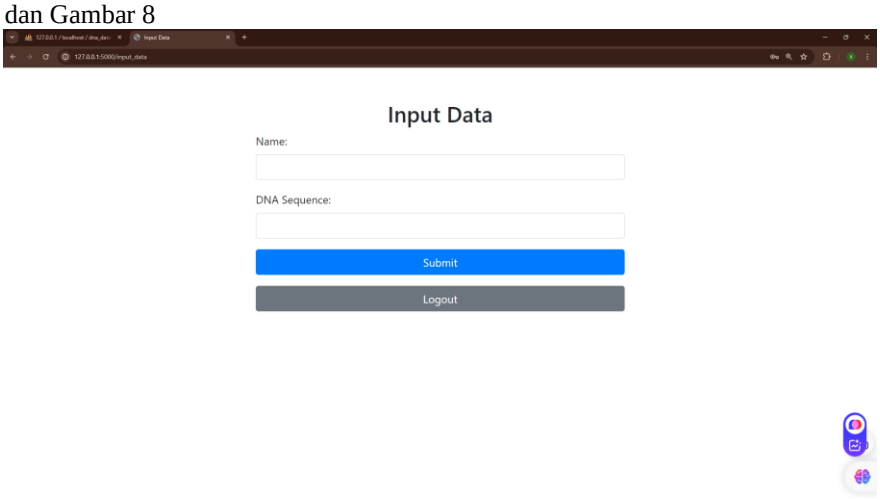
Selanjutnya pada Gambar 6 merupakan Implementasi dari *website* dengan *form login* yang sudah memiliki *Query parameterization*. Pada bagian ini *webserver* dijaalankan pada ip 127.0.0.1 pada port 5000.



Sumber : Hasil Penelitian (2024)

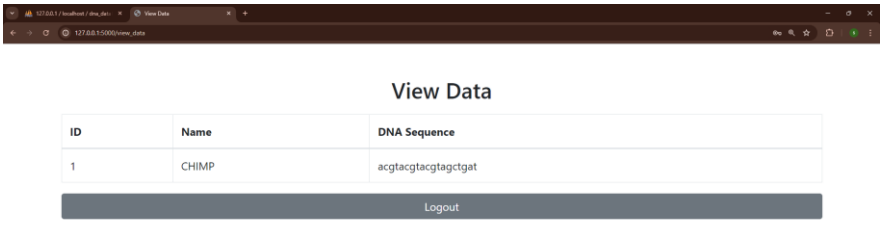
Gambar 6. Web UI Halaman Login

Jika pengguna melakukan *login* dengan *username* dan *password* yang sah maka aplikasi akan berpindah kepada halaman *input* untuk pengguna dengan tipe akun *researcher* dan akan masuk ke halaman *view* jiga melakukan *login* menggunakan akun dengan tipe *user*. Kedua l halaman *website* dari *input* dan *view* bisa dilihat pada Gambar 7 dan Gambar 8



Sumber : Hasil Penelitian (2024)

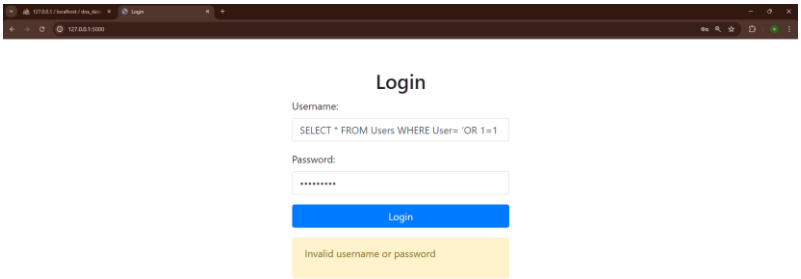
Gambar 7. Web UI Halaman *Researcher*



Sumber : Hasil Penelitian (2024)

Gambar 8. Web UI Halaman *User*

Pada tahap pengujian halaman *login* akan diberikan *query* sql *injection* (Laranjeiro et al., 2012). Gambar 9 merupakan contoh dari sql *injection* pada halaman Login



Sumber : Hasil Penelitian (2024)

Gambar 9. Penyerang Tidak Berhasil Masuk

Pengujian lainnya yang dilakukan bisa dilihat pada Tabel 1, Tabel 1 merupakan hasil pengujian form *login* menggunakan sql *injection*.

Tabel 1 Pengujian menggunakan SQL Inejction

No	Sql Injection	Hasil
1	OR 1=0 --	Tidak berhasil masuk
2	SELECT * FROM Users WHERE User= ‘OR 1=1 --’	Tidak berhasil masuk
3	' OR 1 -- -	Tidak berhasil masuk
4	admin' or '1'='1	Tidak berhasil masuk
5	user' or '1'='1	Tidak berhasil masuk
6	researcher' or '1'='1	Tidak berhasil masuk
7	diash' or '1'='1	Tidak berhasil masuk
8	admin") or "1"="1"/*	Tidak berhasil masuk
9	admin") or ("1"="1"/*	Tidak berhasil masuk
10	admin") or "1"="1	Tidak berhasil masuk
11	admin") or "1"="1"--	Tidak berhasil masuk
12	" or true--	Tidak berhasil masuk
13	' or true--	Tidak berhasil masuk
14	") or true--	Tidak berhasil masuk

15	') or true--	Tidak berhasil masuk

2. Impelemntasi *AES 256*

Pada Tahap implementasi AES 256, data yang diinputkan kedalam *database* akan mendapatkan proses enkripsi dengan AES 256 terlbih dahulu. Gambar 9 merupakan *source code* untuk pemanfaatan AES 256 pada *framework Flask*. Tahap pertama adalah lakukan *generate secret key* secara random.

```
from cryptography.hazmat.primitives import keywrap
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend
import os

key = os.urandom(32)

with open("secret.key", "wb") as key_file:
    key_file.write(key)
```

Sumber : Hasil Penelitian (2024)

Gambar 10. *Generate Random Secret Key*

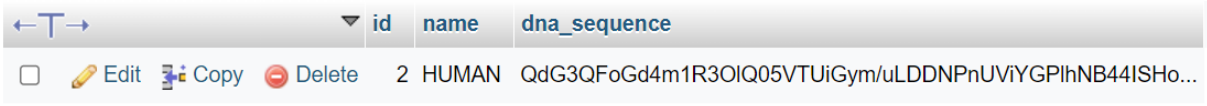
Selanjutnya lakukan enkripsi data menggunakan AES256 dengan *padding* PKCS7 yang bisa dilihat pada Gambar 10.

```
def encrypt(data, key):
    padder = padding.PKCS7(algorithms.AES.block_size).padder()
    padded_data = padder.update(data) + padder.finalize()
    iv = os.urandom(12)
    cipher = Cipher(algorithms.AES(key), modes.GCM(iv), backend=default_backend())
    encryptor = cipher.encryptor()
    ciphertext = encryptor.update(padded_data) + encryptor.finalize()
    encrypted_data = iv + encryptor.tag + ciphertext
    return base64.b64encode(encrypted_data).decode('utf-8')
```

Sumber : Hasil Penelitian (2024)

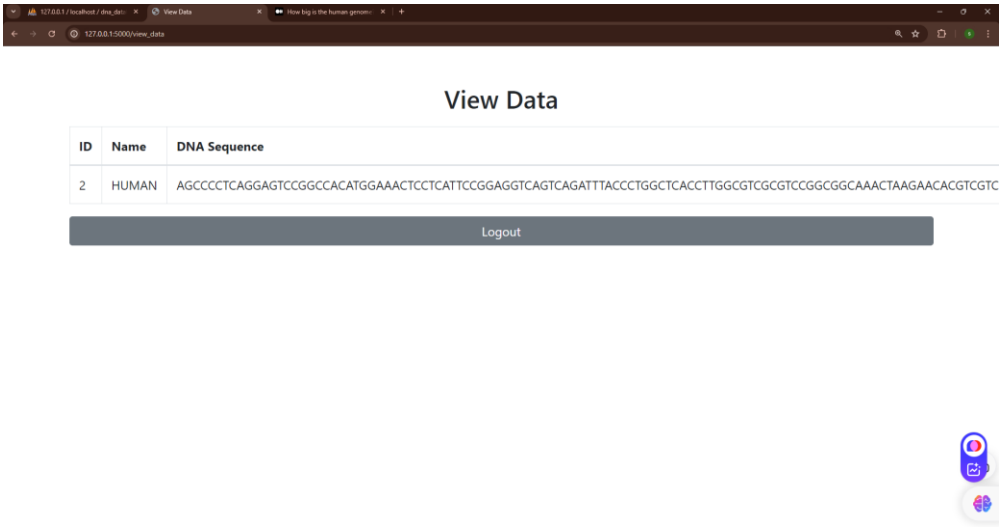
Gambar 11. Enkripsi DNA Sequence data dengan AES 256

Jika diasumsikan penyerang melakukan serangan menggunakan MITM dan berhasil mendapatkan data yang dikirim, maka bisa dilihat pada Gambar 12. Bahwa data yang diinputkan oleh penyerang tidak bisa dilihat langsung kecuali dilakukan proses deskripsi oleh akun sah yaitu akun *user* yang bisa dilihat pada Gambar 13.



Sumber : Hasil Penelitian (2024)

Gambar 12. Hasil Enkripsi DNA *Sequence* data dengan AES 256



Sumber : Hasil Penelitian (2024)

Gambar 13. *User* Sah Dapat Melihat Data DNA *Sequencing*

KESIMPULAN

Enkripsi AES-256 digunakan untuk mengenkripsi data DNA yang sensitif sebelum penyimpanan dalam *database*, memastikan bahwa data tersebut aman dari akses tidak sah dan kebocoran. *Query parameterization* diterapkan untuk melindungi aplikasi dari serangan injeksi SQL dengan memisahkan data dari pernyataan SQL, mencegah eksekusi perintah yang tidak diinginkan akibat *input* pengguna berbahaya. Hasil implementasi menunjukkan bahwa kombinasi enkripsi AES-256 dan *query parameterization* secara signifikan meningkatkan keamanan dan privasi aplikasi web DNA *Sequencing*. Data DNA terenkripsi menjadi tidak dapat dibaca tanpa kunci deskripsi yang sesuai, sementara *query parameterization* mencegah eksploitasi melalui injeksi SQL. Penelitian ini memberikan kontribusi penting pada pengembangan aplikasi web yang aman, terutama dalam konteks aplikasi biomedis yang menangani data sensitif seperti DNA. Untuk penelitian lebih lanjut, disarankan untuk mengeksplorasi metode enkripsi dan teknik keamanan tambahan serta melakukan audit keamanan berkala dan pengujian penetrasi. Dengan demikian, penerapan enkripsi AES-256 dan *query parameterization* terbukti efektif dalam meningkatkan keamanan dan privasi aplikasi web DNA *Sequencing*, memberikan perlindungan kuat terhadap ancaman keamanan data. Memberikan pernyataan bahwa apa yang diharapkan, seperti yang dinyatakan dalam "Pendahuluan" akhirnya dapat mengakibatkan "Hasil dan Diskusi", sehingga ada kompatibilitas. Selain itu dapat juga ditambahkan prospek pengembangan hasil penelitian dan prospek penerapan studi lanjutan. Hindari Data Statistik dan Sampaikan pula rekomendasi untuk penelitian berikutnya berdasarkan sumber.

REFERENSI

Aldianto, D., & Wibowo, A. (2023). *Implementasi Kriptografi Dengan Aes 256 Dan Md 5 Implementation of Cryptography With Aes 256 and Md 5 To Secure Data At Pt . Ebdesk Technology*. *2*(September), 288–295.

Almaj Duddin, H. A., & Fitrani, A. S. (2021). Securing Input and Output Processes on The Web to Minimize SQL-Injection and XSS Attacks Using IDS and IPS Methods. *JOINCS (Journal of Informatics, Network, and Computer Science)*, *4*(1), 6–12. https://doi.org/10.21070/joincs.v4i1.1577

Alodaynan, A. M., & Alanazi, A. A. (2021). A survey of cybersecurity vulnerabilities in healthcare systems. *International Journal of Advanced and Applied Sciences*, *8*(12), 48–55. https://doi.org/10.21833/IJAAS.2021.12.007

Auliafitri, D., Rizkisuro, E., Rangga, M., Malik, M., & Setiawan, A. (2024). Optimalisasi Pengujian Penetrasi: Penerapan Serangan MITM (Man in the Middle Attack) menggunakan Websploit. *Journal of Internet and Software Engineering*, *1*(3), 1–12. https://journal.pubmedia.id/index.php/pjise

Ayuni, S., Budiati, I., Reagan, H. A., Riyadi, Larasaty, P., Pratiwi, A. I., Kurniasih, A., & Meilaningsih, T. (2020). Analisis Hasil Survei Dampak COVID-19 Terhadap Pelaku Usaha Jilid II. *Badan Pusat Statistik Republik Indonesia*, vi+ 22 halaman.

Bonomi, L., Huang, Y., & Ohno-Machado, L. (2018). Privacy Challenges and Research Opportunities for Genomic Data Sharing. *Physiology & Behavior*, *176*(5), 139–148. https://doi.org/10.1038/s41588-020-

0651-0.

Chandra, R. V. H., Kusyanti, A., & Data, M. (2019). Analisis Performa Proses Enkripsi dan Dekripsi Menggunakan Algoritme AES-128 Pada Berbagai Format File. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *3*(1), 481–486. http://j-ptiik.ub.ac.id

Fayans, I., Motro, Y., Rokach, L., Oren, Y., & Moran-Gilad, J. (2020). Cyber security threats in the microbial genomics era: Implications for public health. *Eurosurveillance*, *25*(6), 1–8. https://doi.org/10.2807/1560-7917.ES.2020.25.6.1900574

Febriansyah, F., Asti Dwiyanti, Z., & Diash Firdaus. (2023). Deteksi Serangan Low Rate Ddos Pada Jaringan Tradisional Menggunakan Machine Learning Dengan Algoritma Decision Tree. *Cyber Security Dan Forensik Digital*, *6*(1), 6–11. https://doi.org/10.14421/csecurity.2023.6.1.3951

Firdaus, D., Munadi, R., & Purwanto, Y. (2020). DDoS Attack Detection in Software Defined Network using Ensemble K-means++ and Random Forest. *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2020*, 164–169. https://doi.org/10.1109/ISRITI51436.2020.9315521

Hasanul Fahmi. (2018). Analisis Qos (Quality of Service) Pengukuran Delay, Jitter, Packet Lost Dan Throughput Untuk Mendapatkan Kualitas Kerja Radio Streaming Yang Baik. *Jurnal Teknologi Informasi Dan Komunikasi*, *7*(2), 98–105.

Irawan, B. O. P. I., Tahir, M., Windrastuti, N. A., Cholili, D. Y., Mulaikah, D., & Wachid, A. B. M. S. (2023). Implementasi Kriptografi Pada Keamanan Data Menggunakan Algoritma Advance Encryption Standard (Aes). *Jurnal Simantec*, *11*(2), 167–174. https://doi.org/10.21107/simantec.v11i2.20034

Isminarti, Ilham, A. A., Arief, A., & Syafaruddin. (2023). Improved Data Security Using Advanced Encryption Standard Algorithm on Long-Range Communication System At Smart Grid. *ICIC Express Letters, Part B: Applications*, *14*(5), 499–508. https://doi.org/10.24507/icicelb.14.05.499

Kuo, T. T., Jiang, X., Tang, H., Wang, X. F., Harmanci, A., Kim, M., Post, K., Bu, D., Bath, T., Kim, J., Liu, W., Chen, H., & Ohno-Machado, L. (2022). The evolving privacy and security concerns for genomic data analysis and sharing as observed from the iDASH competition. *Journal of the American Medical Informatics Association*, *29*(12), 2182–2190. https://doi.org/10.1093/jamia/ocac165

Laranjeiro, N., Vieira, M., & Madeira, H. (2012). A robustness testing approach for SOAP Web services. *Journal of Internet Services and Applications*, *3*(2), 215–232. https://doi.org/10.1007/s13174-012-0062-2

Manmadhan, S. (2012). A Method of Detecting Sql Injection Attack to Secure Web Applications. *International Journal of Distributed and Parallel Systems*, *3*(6), 1–8. https://doi.org/10.5121/ijdps.2012.3601

Matta, P., Arora, M., & Sharma, D. (2021). A comparative survey on data encryption Techniques: Big data perspective. *Materials Today: Proceedings*, *46*(xxxx), 11035–11039. https://doi.org/10.1016/j.matpr.2021.02.153

Odeh, A., & Taleb, A. A. (2024). Ensemble learning techniques against structured query language injection attacks. *Indonesian Journal of Electrical Engineering and Computer Science*, *35*(2), 1004–1012. https://doi.org/10.11591/ijeecs.v35.i2.pp1004-1012

Riyanti, A., Rahmanto, B. M., Hardianto, D. R., Daffa, R., & Yuristiawan, A. (2024). *Uji Penetrasi Injeksi SQL terhadap Celah Keamanan Database Website menggunakan SQLmap*. *4*, 1–9.

Scheibner, J., Ienca, M., Kechagia, S., Troncoso-Pastoriza, J. R., Raisaro, J. L., Hubaux, J. P., Fellay, J., & Vayena, E. (2020). Data protection and ethics requirements for multisite research with health data: A comparative examination of legislative governance frameworks and the role of data protection technologies. *Journal of Law and the Biosciences*, *7*(1), 1–30. https://doi.org/10.1093/jlb/lsaa010

Vinatzer, B. A., Heath, L. S., Almohri, H. M. J., Stulberg, M. J., Lowe, C., & Li, S. (2019). Cyberbiosecurity Challenges of Pathogen Genome Databases. *Frontiers in Bioengineering and Biotechnology*, *7*(May), 1–11. https://doi.org/10.3389/fbioe.2019.00106

Wijayanto, C., & Susetyo, Y. A. (2022). Implementasi Flask Framework Pada Pembangunan Aplikasi Sistem Informasi Helpdesk (SIH). *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *7*(3), 858–868. https://doi.org/10.29100/jipi.v7i3.3161

Yulianingsih, Y. (2016). Menangkal Serangan SQL Injection Dengan Parameterized Query. *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, *2*(1), 46–49. https://doi.org/10.26418/jp.v2i1.15507