Kombinasi AES dan HMAC SHA-256 untuk Pengamanan Parameter URL dari Serangan SQL Injection

Gagan Akhmad Fauzi[1]

Jurusan Informatika, Fakultas Teknik, Universitas Siliwangi, Tamansari, Jalan Mugarsari, Kota Tasikmalaya, Jawa Barat 46191, Indonesia[1]

217006110@student.unsil.ac.id[1]

SQL Injection merupakan salah satu ancaman terbesar dalam keamanan aplikasi web, dimana penyerang dapat memanipulasi database melalui exploitasi celah keamanan pada parameter URL. Serangan ini memungkinkan akses ilegal untuk mendapatkan data sensitif. Untuk mengatasi masalah tersebut, diperlukannya teknik pengamanan yang kuat pada pengiriman data, khususnya parameter URL. Penelitian ini mengimplemetasikan kombinasi algoritma kriptografi AES (Advanced Encryption Standard) dan SHA-256 (Secure Hash Algorithm 256) untuk mengamankan parameter URL pada aplikasi web. Algoritma AES-256-CBC digunakan untuk mengenkripsi data yang dikirim melalui URL dan SHA-256 untuk menghasilkan kunci enkripsi yang kuat serta memastikan integritas data melalui HMAC (Hash-based Message Authentication Code). Proses enkripsi ini, terdapat 64 byte data yang terenkripsi. 16 byte pertama sebagai Initialization Vektor (IV), yang memastikan setiap enkripsi unik meskipun data yang dienkripsi sama. 16 byte berikutnya berisi data yang telah di enkripsi menggunakan AES-256-CBC. Terakhir, 32 byte sisa adalah HMAC yang dihasilkan dari SHA-256, berfungsi untuk autentikasi data. Pengujian dilakukan manual dan otomatis menggunakan alat penetration testing SQLMAP untuk mensimulasikan serangan SQL Injection. Hasil pengujian tersebut gagal menginjeksi parameter URL yang telah di enkripsi. Kombinasi AES 256 untuk kerahasiaan data dan HMAC SHA-256 untuk integritas, terbukti efektif dalam mengamankan aplikasi web dari serangan SQL Injection dan menjaga keutuhan data.
***Kata Kunci* – AES, Parameter URL, SHA-256, SQL Injection**

I. PENDAHULUAN

Aplikasi web memiliki peran penting dalam berbagai aspek kehidupan, dari bisnis, layanan akademik hingga layanan publik. Perguruan tinggi saat ini banyak menggunakan aplikasi web untuk memudahkan mahasiswa dalam mengakses informasi, seperti nilai, jadwal kuliah dan lainnya [1]. Aplikasi web juga banyak digunakan sebagai sarana transaksi online, dimana aktivitas tersebut menghasilkan banyak data yang tersimpan dalam aplikasi [2]. Data yang dihasilkan mencakup informasi sensitif, seperti data pribadi pengguna yang sangat berharga dan harus dijaga. Keamanan data menjadi sangat penting, karena aplikasi web memiliki data sensitif yang harus dijaga dari akses yang tidak diinginkan oleh pihak lain. Perlindungan data pun menjadi prioritas utama dalam upaya menjaga integritas atau kerahasiaan serta mencegah potensi kebocoran atau penyalahgunaan informasi yang dapat merugikan [3]. Salah satu metode untuk menjaga keamanan data adalah kriptografi,karena teknik ini memastikan bahwa informasi tetap aman dan tidak bisa dibaca oleh orang yang tidak berwenang. Proses enkripsi, yang mengubah data (plaintext) menjadi bentuk kode (chipertext), berperan besar dalam melindungi kerahasiaan dan integritas data selama transmisi [3][4].

Aplikasi web memiliki dua metode dalam pengiriman dan penerimaan data pada server. Kedua metode tersebut merupakan parameter POST method dan parameter GET method. Terdapat perbedaan mendasar pada Metode POST dan GET, dimana Metode POST mengirimkan data secara tersembunyi di latar belakang, sehingga tidak terlihat di alamat web, sedangkan metode GET mengirimkan data melalui URL (Uniform Resource Locator), sehingga data yang terkirim terlihat di alamat

web [5]. URL (Uniform Resource Locator) merupakan sebuah alamat sumber. URL memiliki beberapa komponen, diantaranya yaitu schema yang dideklarasikan dengan http, https dan ftp, dilanjutkan dengan main-domain sebagai domain utama pada sebuah URL, selanjutnya komponen query paramater yang digunakan untuk mendeklarasikan data yang dikirim oleh client ke server [6]. Hal ini menunjukan adanya kelemahan aplikasi web terutama pada metode GET, karena data yang dikirimkan dapat terlihat pada Uniform Resource Locator (URL) sehingga rentan terhadap serangan SQL Injection. Serangan SQL Injection merupakan salah satu ancaman terbesar dalam keamanan aplikasi web, dimana penyerang dapat memanipulasi database melalui exploitasi celah keamanan seperti memasukan kode SQL berbahaya ke dalam parameter URL [7][8]. Database akan mengeksekusi kode SQL berbahaya yang disisipkan, memungkinkan penyerang dapat melakukan tindakan berbahaya seperti mengedit ataupun menghapus data. Serangan SQL Injection dapat memungkinkan akses ilegal untuk mendapatkan data sensitif [8].

Penelitian yang dilakukan oleh Hamid Wijaya pada (2020) menemukan bahwa implementasi kriptografi AES-128 dalam parameter URL berhasil mencegah serangan SQL Injection. Penggunaan algoritma AES-128 menghasilkan URL yang terenkripsi. Penelitian ini hanya menggunakan AES-128 tanpa kombinasi dengan algoritma lain, sehingga potensi celah keamanan masih ada, terutama jika menghadapi serangan brute force. Selain itu, hanya melakukan enkripsi tanpa adanya integritas atau autentikasi data [9].

Penelitian yang dilakukan oleh Muhamad Luthfi Assidiq dkk. Pada (2024) menunjukan bahwa kombinasi algoritma AES dan SHA-3 mampu menghasilkan kode yang unik dan berbeda untuk setiap data yang diproses, memastikan keamanan data sensitif tetap terjaga. Algoritma AES digunakan untuk menenkripsi data sedangkan algoritma SHA-3 digunakan untuk membuat kunci algoritma AES. Hasil tersebut mendukung efektivitas kombinasi kedua algoritma dalam menjaga keamanan data di dunia digital [10].

Penelitian yang dilakukan oleh Muhamad Rais Rabtsani pada (2024) menghasilkan bahwa penerapan sistem enkripsi pada aplikasi pembayaran YAPE SPP menggunakan algoritma Advanced Encryption Standard (AES) 256 dan SHA256 terbukti efektif. Algoritma AES sangat cocok untuk mengamankan data dalam sistem atau database, karena proses enkripsi membuat data sulit diakses dan diretas tanpa kunci dekripsi. Kunci enkripsi diperkuat dengan SHA256, sehingga lebih aman dari upaya pembacaan. Kombinasi enkripsi AES 256 bit dan SHA256 secara signifikan meningkatkan keamanan, sehingga upaya peretasan tidak berhasil karena data terenkripsi tidak dapat diakses tanpa kunci yang benar [11].

Penelitian yang dilakukan oleh Padla Shravan dkk. Pada (2023) menunjukan bahwa HMAC-SHA3 (Hash-based Message Authentication Code – Secure Hash Algorithm 3) adalah metode kriptografi yang digunakan untuk memastikan keaslian dan integritas pesan. HMAC menggabungkan fungsi hash SHA-3 (seperti SHA-256 atau SHA-512) dengan kunci rahasia untuk menghasilkan nilai otentikasi pesan (MAC). MAC (Message Authentication Code) digunakan untuk memverifikasi keaslian sebuah pesan [12].

Berdasarkan penelitian sebelumnya, dapat disimpulkan bahwa penggunaan algoritma AES-128 telah berhasil meningkatkan parameter URL dari serangan SQL Injection. Namun, pendekatan tersebut masih memiliki kelemahan, seperti rentan terhadap serangan brute force karena memiliki kunci yang lebih pendek. Kelemahan lain adalah tidak semua penelitian yang berkaitan dengan peningkatan keamanan parameter URL memastikan integritas data secara penuh, hanya berfokus pada kerahasiaan. Untuk mengatasi hal tersebut, penelitian ini mengimplementasikan solusi yang efektif dengan mengusulkan kombinasi algoritma kriptografi AES (Advanced Encryption Standard) dan SHA-256 (Secure Hash Algorithm 256) untuk mengamankan parameter URL pada aplikasi web. Algoritma AES digunakan untuk mengenkripsi data yang dikirim melalui URL dan SHA-256 untuk menghasilkan kunci enkripsi yang kuat serta memastikan integritas data melalui HMAC (Hash-based Message Authentication Code). Kombinasi kedua algoritma ini, data yang dikirim melalui URL akan terlindungi dari potensi manipulasi dan akses ilegal.

pISSN: 2252-486X
eISSN: 2548-4710

Solusi ini melibatkan beberapa tahap. Pertama, data yang dikirimkan melalui parameter URL akan di enkripsi menggunakan algoritma AES-256-CBC. Proses enkripsi ini memastikan bahwa meskipun data yang dikirimkan sama, hasil enkripsi akan berbeda setiap kali berkat penggunaan Initialization Vektor (IV). Kunci untuk enkripsi ini dihasilkan melalui SHA-256, yang menjamin kekuatan kunci. Selanjutnya, HMAC yang dihasilkan dari SHA-256 akan dilampirkan pada data terenkripsi untuk menjaga integritas dan keaslian data. Dalam pengujian sistem, metode ini akan diuji menggunakan alat penetrasi untuk mensimulasikan serangan SQL Injection, guna memastikan bahwa parameter yang telah dienkripsi tidak dapat dieksploitasi oleh penyerang.

Tujuan utama dari penelitian ini dengan mengimplementasikan kombinasi algoritma AES dan SHA-256 adalah untuk meningkatkan keamanan data yang dikirim melalui parameter URL serta mencegah serangan SQL Injection pada aplikasi web. Solusi ini diharapkan dapat mencegah kerugian yang dapat ditimbulkan akibat kebocoran data, serta meningkatkan kepercayaan pengguna terhadap aplikasi. Kombinasi algoritma ini tidak hanya mencegah serangan SQL Injection, tetapi menjaga integritas dan kerahasiaan data.

II. TINJAUAN PUSTAKA

Penelitian yang dilakukan oleh Asih Indriati pada (2023) menghasilkan bahwa penerapan algoritma AES pada keamanan URL terbukti efektif. Penelitian tersebut menghasilkan peningkatan kerahasiaan data yang diakses melalui URL karena telah di enkripsi sepenuhnya, menyamarkan informasi serta mencegah serangan SQL Injection. Penelitian ini memiliki beberapa kelebihan seperti keamanan yang lebih baik terhadap URL, namun belum menggunakan algoritma tambahan untuk meningkatkan keamanan lebih lanjut, seperti kombinasi dengan algoritma hashing [1].

Penelitian yang dilakukan oleh Theodora Tantri Trisnawati dkk. Pada (2023) menunjukan bahwa penerapan algoritma RSA untuk mengenkripsi parameter GET yang muncul di URL berhasil meningkatkan keamanan data. Melalui proses enkripsi menggunakan RSA, value parameter GET yang semula berupa data “costumer” berhasil diubah menjadi chipertext “5a9cb05811aa6e4c” yang tidak dapat dibaca oleh pihak ketiga. Penelitian tersebut membuktikan bahwa enkripsi URL dengan algoritma RSA efektif dalam menjaga kerahasiaan data. Algortima RSA memiliki tingkat keamanan yang tinggi karena menggunakan kunci asimetris, namun kelemahannya adalah waktu proses yang lebih lama dibandingkan dengan algoritma AES yang menggunakan kunci simetris [13].

Penelitian yang dilakukan oleh Bagus Ajie Iswara dkk. Pada (2023) membandingkan penerapan algoritma AES dan RC4 pada URL untuk keamanan data. Penelitian tersebut menghasilkan bahwa hasil enkripsi algoritma AES mendapatkan string yang lebih panjang dibandinganken algoritma RC4. Penerapan algoritma tersebut dapat dipastikan bahwa algoritma AES lebih kuat keamanannya di bandingkan algoritma RC4 [6].

Beberapa penelitian sebelumnya, mengenai pengamanan URL dengan menggunakan algoritma kriptografi AES, RSA dan RC4 telah berhasil mengenkripsi URL dan melindungi dari serangan SQL Injection. Namun, pendekatan tersebut memiliki kelemahan, seperti Algoritma RSA membutuhkan waktu komputasi yang lebih lama untuk enkripsi dan dekripsi data, terutama untuk data yang besar. RC4 terbukti lemah dibandingkan dengan algoritma AES dalam hal keamanan kriptografi. Selain itu, Algoritma AES tanpa adanya kombinasi dengan algoritma hashing pada keamanan URL, rentan terhadap serangan brute force yang mencoba memecahkan kunci enkripsi yang lemah, serta serangan manipulasi, seperti perubahan data tanpa terdeteksi. Selain itu, rentan terhadap serangan penggantian chipertext, yang dapat memodifikasi data terenkripsi sehingga dapat memasukan data berbahaya yang dilakukan oleh penyerang. Kelemahan lain adalah tidak semua penelitian yang berkaitan dengan peningkatan keamanan parameter URL memastikan integritas data secara penuh, hanya berfokus pada kerahasiaan.

III. METODOLOGI PENELTIAN

Penelitian ini dilakukan dengan beberapa tahapan yang meliputi studi literatur, analisis kebutuhan sistem, perancangan model,

implementasi, serta pengujian dan analisis. Berikut adalah penjelasan setiap tahapan:

A. Studi Literatur

Tahap studi literatur bertujuan untuk mencari referensi teori yang relevan dengan studi kasus penelitian yang dilakukan. Proses ini melibatkan pengumpulan informasi yang luas tentang objek yang akan diteliti, seperti membahas tentang metode pengamanan URL, teknik kriptografi dan serangan SQL Injection. Literatur yang digunakan mencakup penelitian sebelumnya mengenai penerapan algoritma AES, SHA-256 dan kombinasi metode kriptografi lainnya dalam pengamanan aplikasi web.

B. Analisis Kebutuhan Sistem

Pada tahap ini terdapat kebutuhan software dan kebutuhan hardware yang dibutuhkan untuk mendukung implementasi sistem pengamanan berbasis algoritma kriptografi AES-256-CBC dan HMAC SHA-256.

C. Perancangan Model

Tahap perancangan model merupakan tahapan untuk merancang model sistem. Model ini mencakup struktur enkripsi dan dekripsi menggunakan algoritma AES-256-CBC untuk menjaga kerahasiaan data, dan penggunaan SHA-256 untuk menghasilkan kunci enkripsi yang aman, serta HMAC untuk memastikan integritas data. Diagram alur proses enkripsi dan dekripsi dirancang pada tahap.

D. Implementasi

Tahap implementasi merupakan tahap penerapan sesuai dengan acuan analisa sistem. Pada tahap implementasi, algoritma kriptografi AES-256-CBC dan SHA-256 diterapkan pada parameter URL yang dikirim dari client ke server, dengan menggunakan kunci enkripsi yang dihasilkan dari SHA-256. Selanjutnya, HMAC ditambahkan untuk menjaga keaslian dan integritas data. Implementasi ini dilakukan menggunakan bahasa pemrograman PHP dan library kriptografi yang relevan.

E. Pengujian dan Analisis

Tahap pengujian dan analisis merupakan tahap yang fokus pada validasi implementasi yang telah diterapkan, yang bertujuan memastikan bahwa semua kebutuhan sesuai dengan spesifikasi yang telah ditetapkan dan dirumuskan dengan baik. Pengujian dilakukan dengan cara manual dan otomatis menggunakan alat penetration testing SQLMap untuk mensimulasikan serangan SQL Injection pada parameter URL yang telah terenkripsi. Keberhasilan pengujian di ukur berdasarkan apakah parameter URL yang terenkripsi mampu mencegah eksploitasi SQL Injection. Selain itu, dilakukan analisis terhadap kinerja enkripsi-dekripsi dan ketahanan metode yang di usulkan terhadap beberapa serangan, termasuk manipulasi data.

IV. IMPLEMENTASI DAN PEMBAHASAN

A. Analisis Kebutuhan Sistem

Tahap analisis kebutuhan sistem dalam penelitian ini bertujuan untuk mengidentifikasi komponen yang diperlukan untuk pengamanan data parameter URL. Analisis kebutuhan sistem terdiri dari dua poin utama:

1. Kebutuhan Hardware

TABEL 1
KEBUTUHAN *HARDWARE*

Jenis	Spesifikasi
Processor	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz
RAM	8 GB
Storage	SSD 259 GB

Tabel 1 merupakan spesifikasi hardware yang digunakan dalam pengujian sistem keamanan enkripsi pada parameter URL.

2. Kebutuhan Software

TABEL 2
KEBUTUHAN *SOFTWARE*

Jenis	Spesifikasi
Sistem Operasi	Windows 11
Sistem Operasi (VM)	Kali Linux 2.6
Virtual Machine	VirtualBox
Web Server	Apache 2.4.56
Database	MariaDB 10.4.28
Bahasa Pemrograman	PHP, HTML

pISSN: 2252-486X
eISSN: 2548-4710

Library Kriptografi	OpenSSL

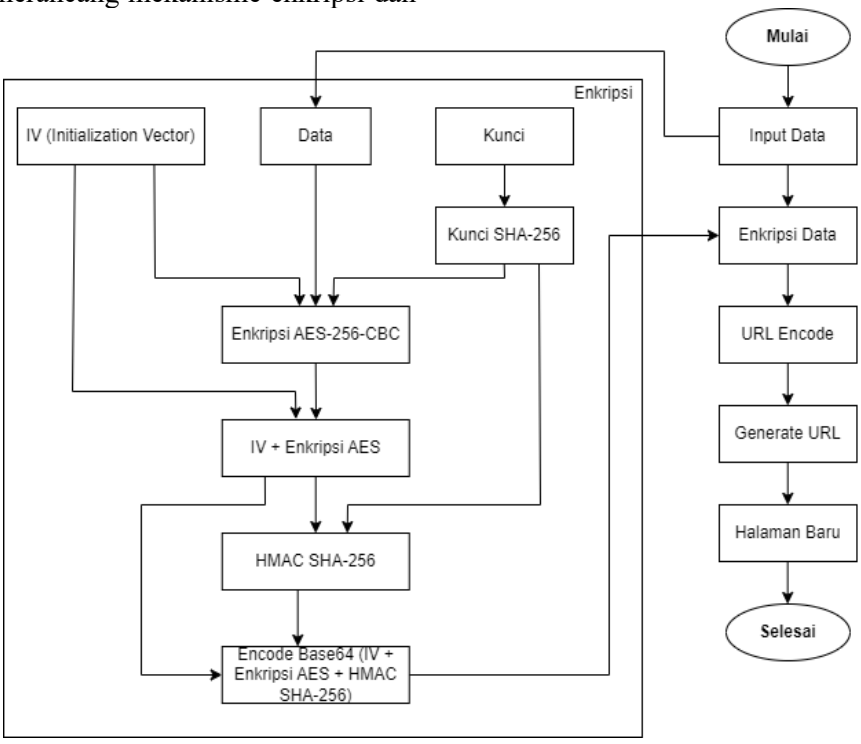
Tabel 2 merupakan spesifikasi software yang digunakan dalam pengujian sistem keamanan enkripsi pada parameter URL.

B. Perancangan Model

Perancangan model dalam penelitian ini merupakan langkah penting dalam menentukan bagaimana sistem pengamanan URL akan bekerja. Fokus utama dari perancangan model ini adalah merancang mekanisme enkripsi dan dekripsi data yang aman serta memastikan integritas data saat dikirimkan melalui URL. Berikut ini adalah model yang dirancang dalam penelitian ini:

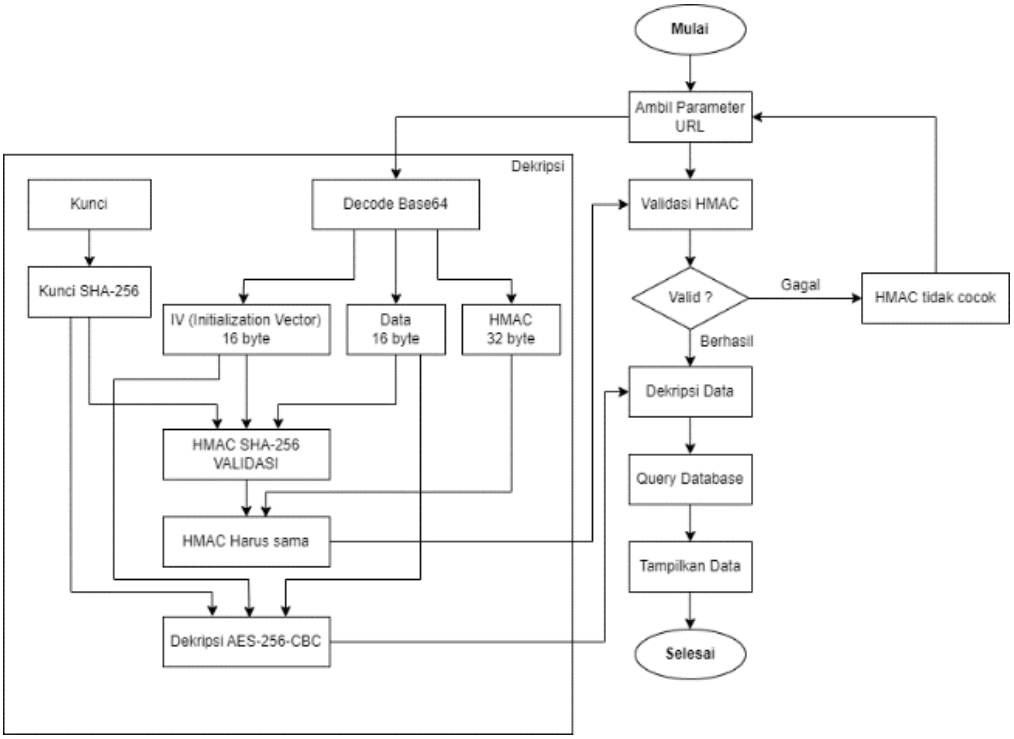
1. Diagram Alur Proses Enkripsi dan Dekripsi Data

Diagram alur proses dirancang untuk menggambarkan secara detail bagaimana sistem enkripsi dan dekripsi data bekerja. Proses ini melibatkan langkah-langkah berikut:



Gambar 1 Diagram Alur proses Enkripsi.

Alur proses enkripsi yang ditampilkan dalam Gambar 1 melibatkan beberapa langkah utama. Dimulai dari input data, data tersebut dimasukkan kedalam proses enkripsi menggunakan AES-256-CBC. Dalam proses ini, kunci enkripsi diberikan sebagai string, yang kemudian di hash menggunakan algoritma SHA-256 untuk menghasilkan kunci dengan panjang 256 bit. Selanjutnya, IV (Initialization Vector) acak sepanjang 16 byte dibuat untuk memastikan setiap enkripsi unik, meskipun data yang di enkripsi sama. Data yang dimasukkan kemudian di enkripsi menggunakan algoritma AES-256-CBC sepanjang 16 byte, dengan kunci yang sudah di hash dan IV yang dihasilkan. Hasil enkripsi ini digabungkan dengan IV untuk menciptakan paket data. Untuk memastikan integritas data, HMAC (Hash-based Message Authentication Code) SHA-256 dihasilkan berdasarkan gabungan IV dan data terenkripsi, menggunakan kunci hash yang sama dengan proses enkripsi AES-256-CBC untuk memverifikasi keaslian data. Setelah HMAC dihasilkan sepanjang 32 byte, IV, data terenkripsi, HMAC digabungkan dengan menjadi satu kesatuan, yang kemudian di encode ke dalam format base64. Data dalam format base64 ini kemudian di encode lagi menggunakan URLEncode untuk dimasukan ke dalam parameter URL. Setelah itu, menghasilkan URL yang memiliki parameter yang terenkripsi dan siap ditampilkan pada halaman baru.



Gambar 2 Diagram Alur proses Dekripsi

Alur proses dekripsi yang ditampilkan dalam Gambar 2 melibatkan beberapa langkah utama. Proses dimulai dengan pengambilan parameter dari URL. Setelah parameter diterima, data kemudian masuk ke dalam proses dekripsi. Data yang tekah diterima ini dienkripsi dalam format Base64, sehingga langkah berikutnya adalah melakukan proses decode Base64 untuk memisahkan bagian-bagian data, yaitu IV (Initialization Vektor) sebesar 16 byte, data yang terenkripsi menggunakan AES-256-CBC sebesar 16 byte, dan HMAC sebesar 32 byte.

Pada saat yang sama, kunci yang digunakan dalam proses dekripsi adalah kunci yang sama dengan kunci yang digunakan dalam proses enkripsi. Kemudian kunci ini di hash menggunakan algoritma SHA-256 untuk memastikan keamanan. Setelah kunci dihash, sistem kemudian membuat HMAC baru menggunakan gabungan dari kunci yang sudah di hash, IV (yang diperoleh dari hasil decode Base64) dan data (yang diperoleh dari hasil decode Base64). HMAC ini dihasilkan menggunakan algoritma HMAC SHA-256. Selanjutnya adalah membandingkan HMAC yang baru dihasilkan dengan HMAC yang diterima dari hasil decode Base64. Jika kedua HMAC ini tidak sama, maka proses dihentikan dan sistem menampilkan pesan bahwa HMAC tidak cocok, dan proses kembali ke tahap awal untuk pengambilan parameter dari URL. Ini memungkinkan proses diulang dengan parameter yang baru atau yang diperbaiki. Namun, jika kedua HMAC cocok, proses dilanjutkan dengan mendekripsi data menggunakan kunci SHA-256 yang baru dihasilkan, IV dari hasil decode Base64 dan data dari hasil decode Base64. Setelah data berhasil didekripsi, sistem akan melakukan query ke database untuk mengambil informasi yang diperlukan. Data hasil query tersebut kemudian ditampilkan, dan proses selesai.

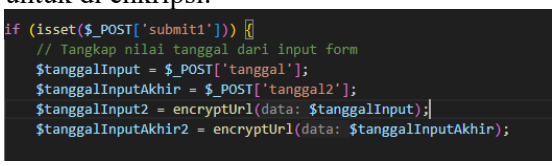
C. Implementasi

Tahap implementasi merupakan penerapan hasil perancangan yang telah dilakukan sebelumnya. Pada penelitian ini, fokus implementasi adalah pada pengamanan data parameter URL menggunakan algoritma AES-256-CBC dan SHA-256 untuk menjaga kerahasiaan, integritas, dan autentikasi data. HMAC (Hash-based Message Authentication Code) dengan SHA-256 diterapkan untuk menjamin integritas data yang dikirimkan melalui URL.

pISSN: 2252-486X
eISSN: 2548-4710

1. Implementasi Enkripsi Parameter URL

Enkripsi parameter URL dilakukan dengan menggunakan algoritma AES-256-CBC. Algoritma ini dipilih karena memiliki tingkat keamanan yang tinggi dengan panjang kunci 256 bit dan proses enkripsi yang membutuhkan IV (Initialization Vektor) untuk mencegah pola yang sama pada data terenkripsi. Proses enkripsi ini memastikan bahwa data yang dikirimkan dalam URL aman dari pihak yang tidak berwenang. Gambar 4.3 merupakan proses pengambilan data dari input pengguna untuk di enkripsi.



```
if (isset($_POST['submit1'])) {
    // Tangkap nilai tanggal dari input form
    $tanggalInput = $_POST['tanggal'];
    $tanggalInputAkhir = $_POST['tanggal2'];
    $tanggalInput2 = encryptUrl(data: $tanggalInput);
    $tanggalInputAkhir2 = encryptUrl(data: $tanggalInputAkhir);
```

Gambar 3 data yang akan di enkripsi

Gambar 3 menunjukkan alur proses pengambilan data yang diinput oleh pengguna dan kemudian dienkripsi sebelum dikirim melalui URL.

TABEL 3
PSEUDO CODE ENKRIPSI AES DAN SHA-256

```
function encryptUrl($data){
$kunci
='kuncisha2untukskripsiinformatika';

  // Hash kunci menggunakan SHA-256
$kunciSHA256 = hash('sha256', $kunci,
true);

 // IV acak 16 byte
$iv = openssl_random_pseudo_bytes(16);

 // Enkripsi data menggunakan AES-256-
CBC
$EnkripsiAES = openssl_encrypt($data,
'aes-256-cbc',$kunciSHA256,
OPENSSL_RAW_DATA, $iv);

 // Gabungkan IV dan data terenkripsi
$GabunganIVdanAES = $iv .
$EnkripsiAES;

 // Buat HMAC SHA-256 untuk
memastikan integritas
$hmac = hash_hmac('sha256',
$GabunganIVdanAES, $kunciSHA256,
true);

 // Gabungkan HMAC dengan data
terenkripsi dan encode dalam base64
 return
base64_encode($GabunganIVdanAES .
$hmac);
}
```

Tabel 3 menunjukan bahwa Fungsi encryptUrl($data) digunakan untuk mengenkripsi data sensitif sebelum dikirim melalui URL dengan menggunakan algoritma AES-256-CBC. Proses dimulai dengan mendefinisikan sebuah kunci enkripsi, karena menggunakan algoritma AES-256-CBC, maka kuncinya harus memiliki 256 bit atau 32 byte. Kunci dalam penelitian ini adalah ‘kuncisha2untukskripsiinformatika’. Kunci ini kemudian di hash menggunakan algortima SHA-256 untuk memiliki kunci yang lebih aman dan memiliki ukuran 32 byte. Fungsi ini membuat sebuah IV (Initialization Vektor) yang acak sepanjang 16 byte menggunakan openssl_random_pseudo_bytes(). IV digunakan dalam algortima AES untuk memastikan bahwa hasil enkripsi untuk data yang sama akan berbeda setiap kali proses enkripsi dilakukan. Setelah mempersiapkan kunci dan IV, fungsi ini menggunakan openssl_encrypt() untuk mengenkripsi data input. Proses ini menghasilkan chipertext, yang merupakan bentuk data terenkripsi dari input asli. IV dan Chipertext digabungkan menjadi satu string.

Fungsi ini menghasilkan HMAC (Hash-based Message Authentication Code) menggunakan SHA-256 untuk memastikan integritas dari data yang telah dienkripsi. HMAC dihasilkan dengan menggunakan gabungan antara IV dan ciphertext, serta kunci yang telah di has sebelumnya. Pembuatan HMAC penting karena memungkinkan penerima untuk memverifikasi bahwa data yang diterima tidak telah dimanipulasi atau dirusak. IV, ciphertext dan HMAC digabungkan menjadi satu string dan diencode dalam format Base64 menggunakan base64_encode(). Encoding ini penting untuk menjadikan data lebih aman. Fungsi ini tidak hanya mengenkripsi data, tetapi juga memastikan integritas dan keamanan data selama proses pengiriman melalui URL.

Tabel 4 merupakan proses membuat URL dan penggunaan fungsi urlencode() untuk meng-encode kedua data yang terenkripsi sebelum dimasukkan ke dalam URL. Proses encoding ini penting untuk memastikan bahwa karakter-karakter khusus dalam data, seperti karakter tambah (+), spasi dan simbol lainnya, tidak menggangu struktur URL yang dihasilkan. Penggunaan urlencode() mengkonversi karakter-karakter tersebut menjadi format yang sesuai untuk URL, seperti karakter tambah (+) menjadi %2B, sehingga data dapat dikirim dengan aman tanpa risiko kesalahan.

TABEL 4
CODE GENERATE URL

```
onclick="location.href='cetaktest.php?tanggal=<?= urlencode($tanggalInput2); ?>&tanggal2=<?= urlencode($tanggalInputAkhir2); ?>'">
```

Penggunaan urlencode() pada data yang terenkripsi memastikan bahwa setiap karakter yang berpotensi merusak struktur URL akan diubah menjadi format yang aman. Proses encoding menghasilkan URL yang memiliki parameter yang terenkripsi dan menampilkan halaman baru.

TABEL 5
URL YANG TERENKRIPSI

```
http://localhost/disduk1/cetaktest.php?tanggal=jtbe%2FWhxntbwySfpxqnvcgzn5SpxtLqZl2jnJ8FA7qbUmH3JFHZ5V9JV4P4Sqdj3dwzul5v4R5iev3HcJ%2F33BA%3D%3D&tanggal2=jgq4hYobeSbfT14UzGBkXI1pfeuhbjE7PSawXcCT0ILlXAIqIp9CMpo92Ir%2B4nnUwPFpbrzHeK5T8U%2Bt73vflw%3D%3D
```

Tabel 5 merupakan URL di halaman baru yang memiliki parameter query tanggal dan tanggal2. Parameter query tersebut akan diterima dan dapat di proses lebih lanjut, seperti mendekripsi nilai-nilai tersebut menggunakan fungsi dekripsi yang telah di siapkan sebelumnya.

2. Implementasi Dekripsi Parameter URL

Proses dekripsi bertujuan untuk mengambil data yang terenkripsi dalam parameter URL dan mengembalikannya ke bentuk aslinya. Gambar 4.4 merupakan proses pengambilan data yang terenkripsi dalam parameter URL menggunakan bahasa pemrograman php.

```
if (isset($_GET['tanggal'], $_GET['tanggal2'])) {
    $tanggalDekripsi = decryptUrl(EnkripsiData: $_GET['tanggal']);
    $tanggal2Dekripsi = decryptUrl(EnkripsiData: $_GET['tanggal2']);
```

Gambar 4 Pengambilan parameter URL

Gambar 4 menunjukan proses pengambilan nilai-nilai parameter URL yang terenkripsi. Apabila kedua parameter tersebut ada, maka proses pengambilan nilai-nilai parameter URL dapat dilakukan.

Tabel 6 merupakan proses dekripsi algoritma AES-256-CBC dan verifikasi HMAC SHA-256.

TABEL 6
PSEUDO CODE FUNGSI DEKRIPSI AES-256-CBC

```
function decryptUrl($EnkripsiData)
{
 $kunci
='kuncisha2untukskripsiinformatika';
 // Hash kunci menggunakan SHA-256
 $kunciSHA256 = hash('sha256', $kunci, true);

//decode gabungan data terenkripsi dari parameter URL  menggunakan Base64
$decodedData = base64_decode($EnkripsiData);

//ambil IV (Initialization Vektor) 16 byte pertama dari data yang sudah di decode
$iv = substr($decodedData, 0, 16);

// ambil data terenkripsi 16 byte kedua dari data yang sudah di decode
$Data = substr($decodedData, 16, -32);

// ambil HMAC 32 byte terakhir dari data yang sudah di decode
$hmac = substr($decodedData, -32);

//membuat HMAC dari IV dan data terenkripsi hasil dari decode data
$PerbandinganHMAC = hash_hmac('sha256', $iv . $Data, $kunciSHA256, true);

// Membandingkan HMAC yang telah di buat dan HMAC hasil dari decode
   if (!hash_equals($hmac, $PerbandinganHMAC)) {
```


pISSN: 2252-486X
eISSN: 2548-4710

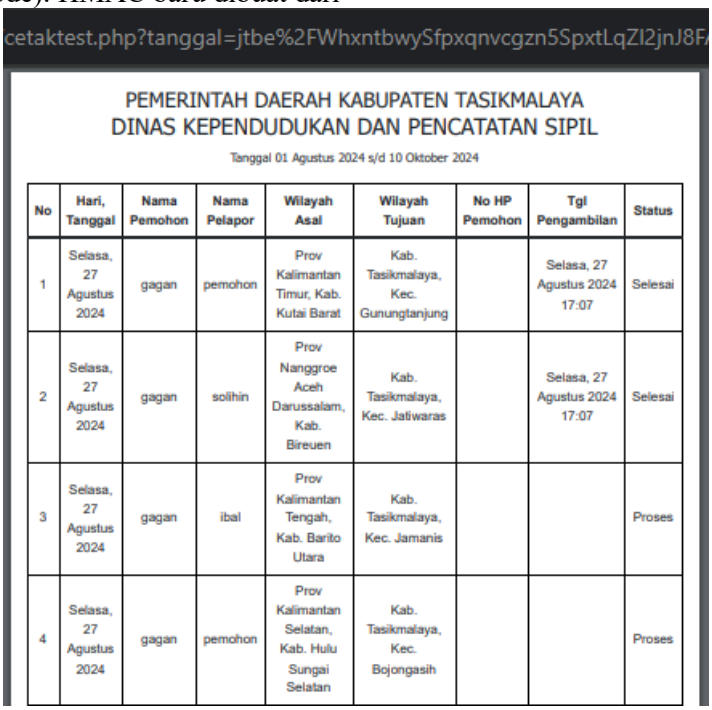
```
        die( "HMAC tidak cocok");
    }

// dekripsi data menggunakan AES-256-CBC
$DekripsiData = openssl_decrypt($Data, 'aes-256-cbc', $kunciSHA256, OPENSSL_RAW_DATA, $iv);

//mengembalikan data asli hasil dekripsi
 return $DekripsiData;
}
```

Tabel 6 menunjukan bahwa Fungsi decryptUrl() bertujuan untuk mendekripsi data yang telah dienkripsi sebelumnya menggunakan algoritma AES-256-CBC. Fungsi ini dimulai dengan mendefinisikan kunci rahasia yang sama dengan kunci dalam proses enkripsi, kunci tersebut di hash menggunakan algoritma SHA-256 untuk meningkatkan keamanannya.
Data terenkripsi yang dikodekan dalam Base64 diambil dari parameter URL dan di decode menggunakan base64_decode(), yang menghasilkan data mentah. Fungsi kemudian memisahkan 16 byte pertama sebagai IV (Initialization Vector), 16 byte berikutnya sebagai data terenkripsi, dan 32 byte terakhir sebagai HMAC (Hash-based Message Authentication Code). HMAC baru dibuat dari gabungan IV dan data terenkripsi menggunakan kunci yang telah di hash sebelumnya. HMAC baru ini dibandingkan dengan HMAC asli dari hasil decode Base64 untuk memverifikasi integritas data. Ketidakcocokan HMAC akan menghentikan proses dan menampilkan pesan kesalahan "HMAC tidak cocok" untuk mencegah penggunaan data yang telah dimodifikasi. HMAC yang cocok akan melanjutkan proses dekripsi data dari hasil encode Base64 menggunakan AES-256-CBC dengan IV dari hasil encode Base64 dan kunci yang telah di hash sebelumnya menjadi data asli. data asli ini akan digunakan untuk menjalankan query ke database.

TABEL 7
PROSES MENJALAN QUERY SQL

```
 $get = mysqli_query($c, "select * from datapindah where tanggal between '$Tanggal1Dekripsi' and '$Tanggal2Dekripsi' ");
```

Tabel 7 menunjukan bagaimana fungsi mengambil data dari tabel berdasarkan data yang telah didekripsi. Hasil dari query ini disimpan dalam variabel $get, yang kemudian dapat digunakan untuk menampilkan informasi yang relevan seperti pada Gambar 5.



Gambar 5 Menampilkan Informasi

Gambar 5 menampilkan informasi dari data yang telah di dekripsi sebelumnya. Data yang telah berhasil didekripsi digunakan sebagai parameter dalam query database untuk menampilkan informasi yang relevan dan sesuai dengan kebutuhan.

D. Pengujian dan Analisis

Tahap pengujian dan analisis fokus pada pengujian SQL Injection sebelum dan setelah penerapan kombinasi AES-256-CBC dan HMAC SHA-256 dengan cara manual dan otomatis menggunakan SQLMap.

1. Pengujian SQL Injection secara Manual

Pengujian ini dilakukan dengan cara melakukan input manual pada parameter URL ada di aplikasi. Metode ini melibatkan pengujian payload SQL Injection untuk mengevaluasi apakah aplikasi web rentan terhadap serangan ini sebelum penerapan enkripsi. Tabel 8 menunjukan penerapan payload ' OR '1'='1'; # pada parameter URL sebelum di enkripsi yang bertujuan untuk melihat kerentanan SQL Injection pada aplikasi web.

TABEL 8
PENGUJIAN PAYLOAD SQLI SEBELUM DI ENKRIPSI

http://localhost/disduk/cetaktest.php?tanggal=2024-08-01&tanggal2=2024-10-10’ OR ‘1’=‘1’; #

Pengujian payload SQL Injection pada parameter URL menampilkan pesan error yang memperlihatkan nama tabel dari database aplikasi web yang terdapat pada Tabel 9.

TABEL 9
PESAN ERROR

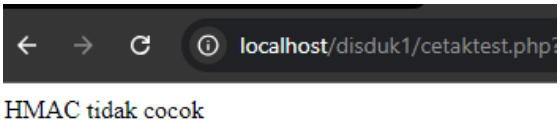
Fatal error: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB *server* version for the right syntax to use near '' at line 1 in C:\xampp\htdocs\disduk\cetaktest.php:190 Stack trace: #0 C:\xampp\htdocs\disduk\cetaktest.php(190): mysqli_*query*(Object(mysqli), 'select * from d...') #1 {main} thrown in

Tabel 9 menunjukkan bahwa aplikasi web masih rentan terhadap SQL Injection, meskipun menghasilkan kesalahan. Kesalahan ini, meskipun menghentikan eksekusi script, tetapi menunjukkan potensi kerentanan yang ada. Penerapan metode enkripsi AES-256-CBC dan HMAC SHA-256 menjadi sangat penting untuk melindungi data parameter URL dan integritas data dari potensi serangan yang lebih berbahaya.

TABEL 10
PENGUJIAN PAYLOAD SQLI SETELAH DI ENKRIPSI

http://localhost/disduk1/cetaktest.php?tanggal=woPrSz4ORzH1krXOKV46WKvuy6jbc%2FsncJBFcXSSJ5058i%2Fs8A2Ryeo4BeY8So3roTXhYUFE5bRGJqan%2Fhh95g%3D%3D%20&tanggal2=5ybFExK2IXTap334mmHQHQzgmbdH8X5vwpgK8J0LjGnC2LW8np6pGB7m15ng0aNY3lrRhevD1%2BTLRoSjXbm5Vw%3D%3D' OR '1'='1'; #

Tabel 10 merupakan pengujian payload SQL Injection pada parameter URL setelah penerapan kombinasi AES-256-CBC dan HMAC SHA-256. Pengujian ini menunjukkan bahwa setelah proses enkripsi diterapkan, payload SQL Injection tidak berfungsi karena dalam proses enkripsi menggunakan AES-256-CBC terdapat HMAC SHA-256 untuk memastikan integritas data dan dapat menampilkan pesan “HMAC tidak cocok”, apabila data telah dimodifikasi seperti pada Gambar 6.



Gambar 6 Pesan HMAC tidak cocok

Gambar 6 menunjukan bahwa aplikasi web menampilkan pesan “HMAC tidak cocok” disebabkan karena dalam proses enkripsi, HMAC digunakan untuk memastikan integritas data, sehingga ketika parameter URL dimodifikasi oleh payload, HMAC akan mendeteksi ketidaksesuaian dan menghentikan proses dekripsi. Tahap pengambilan parameter URL, terjadi proses dekripsi yang memverifikasi HMAC sebelum melanjutkan ke tahap selanjutnya.

2. Pengujian SQL Injection Menggunakan SQLMAP

pISSN: 2252-486X
eISSN: 2548-4710

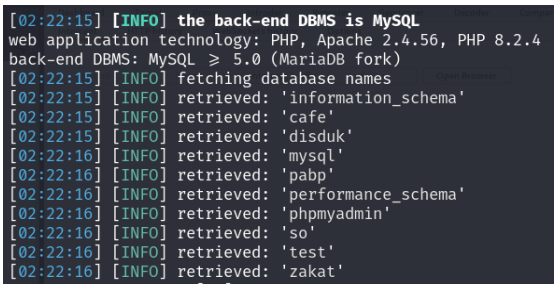
Pengujian SQL Injection secara otomatis menggunakan SQLMap dilakukan untuk mendeteksi dan mengeksploitasi kerentanan SQL Injection pada parameter URL sebelum dan setelah penerapan enkripsi algoritma AES-256-CBC dan HMAC SHA-256. Menurut Lika dkk. dalam penelitian yang dikutip oleh Ade Riyanti dkk., salah satu alat yang sangat berguna dalam mengeksploitasi kelemahan keamanan pada situs web adalah SQLmap. Alat ini sering digunakan untuk mendeteksi dan mengeksploitasi kerentanan SQL Injection [8].

Sebelum penerapan enkripsi, parameter URL di uji untuk melihat kerentanan seranga SQL Injection. Tabel 11 merupakan perintah SQLMap yang digunakan dalam pengujian ini.

TABEL 11
PERINTAH SQLMAP

sqlmap -u "http://192.168.100.100/disduk/cetaktest.php?tanggal=2024-08-01&tanggal2=2024-10-10" --dbs

Tabel 11 menunjukan bahwa dalam perintah SQLMap -u digunakan untuk menentukan URL yang akan di uji dan --dbs digunakan untuk menampilkan semua nama database.



Gambar 7 hasil Scanning SQLMap

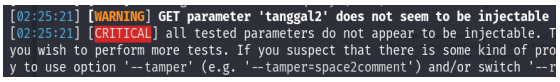
Hasil pengujian ini pada Gambar 7 menunjukkan bahwa SQLMap mampu melakukan serangan SQL Injection dan menampilkan daftar nama database dari server yang digunakan oleh aplikasi web. Ini menunjukkan bahwa parameter URL aplikasi web rentan terhadap eksploitasi SQL Injection, yang dapat dimanfaatkan oleh penyerang untuk mengakses data sensitif di dalam database.

Pengujian SQLMap dilakukan kembali pada parameter URL yang telah terenkripsi. Tabel 12 merupakan perintah SQLMap yang digunakan dalam pengujian kerentanan SQL Injection setelah penerapan kombinasi AES-256-CBC dan HMAC SHA-256 pada parameter URL.

TABEL 12
PERINTAH SQLMAP

sqlmap -u "http://192.168.100.100/disduk1/cetaktest.php?tanggal=woPrSz4ORzH1krXOKV46WKvuy6jbc%2FsncJBFcXSSJ5058i%2Fs8A2Ryeo4BeY8So3roTXhYUFE5bRGJqan%2Fhh95g%3D%3D%20&tanggal2=5ybFExK2IXTap334mmHQHQzgmbdH8X5vwpgK8J0LjGnC2LW8np6pGB7m15ng0aNY3lrRhevD1%2BTLRoSjXbm5Vw%3D%3D" --dbs

Pengujian ini menunjukkan bahwa SQLMap tidak dapat mengeksploitasi semua parameter URL yang telah dienkripsi seperti pada Gambar 8.



Gambar 8 Hasil scanning SQLMap

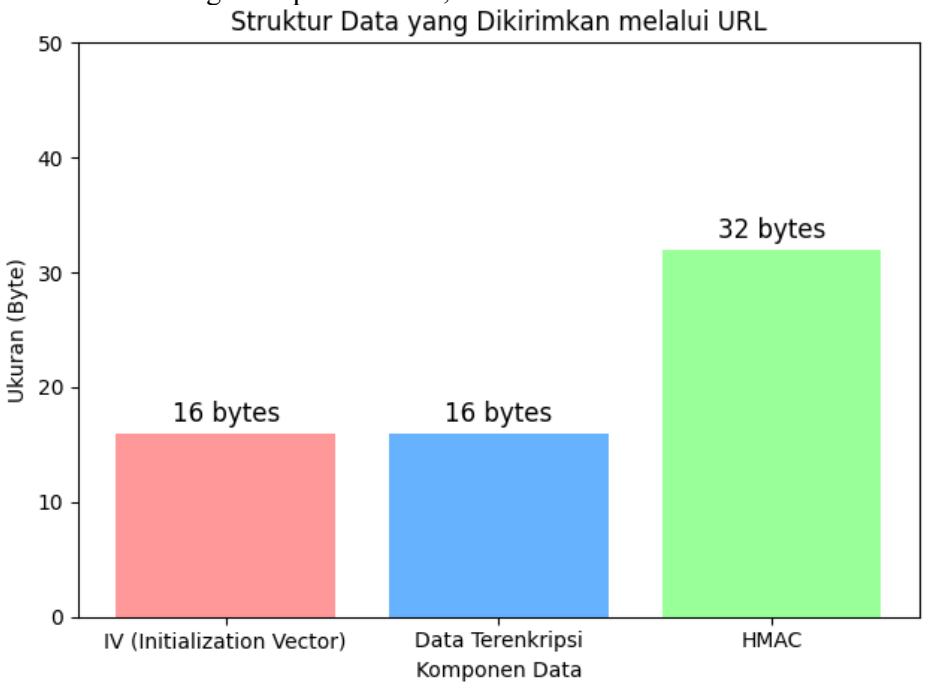
Hasil pengujian ditunjukkan pada Gambar 8, di mana SQLMap menampilkan pesan "all tested parameters do not appear to be injectable". Ini menunjukkan bahwa parameter URL tidak lagi rentan terhadap serangan SQL Injection setelah proses enkripsi diterapkan. Pesan tersebut menunjukan bahwa SQLMap tidak dapat mengeksekusi serangan karena data dalam parameter URL telah terlindungi oleh enkripsi dan integritasnya dijaga melalui HMAC. Setiap upaya untuk memodifikasi data yang dikirim akan terdeteksi, dan proses dekripsi akan dihentikan jika HMAC tidak sesuai, sehingga serangan tidak dapat dilakukan.

Hasil pengujian ini memperkuat pentingnya penggunaan enkripsi AES-256-CBC dan HMAC SHA-256 dalam mencegah kerentanan SQL Injection, karena metode ini secara efektif melindungi parameter URL dari eksploitasi yang dilakukan oleh SQLMap.

3. Analisis Hasil Pengujian

Analisis hasil pengujian menunjukkan bahwa penerapan enkripsi memberikan perlindungan terhadap serangan SQL Injection, terutama melalui kombinasi AES-256-CBC

dan HMAC SHA-256. Setiap data yang dikirimkan melalui URL,tidak hanya di enkripsi tetapi juga dilindungi oleh HMAC untuk memastikan bahwa data tidak dapat dimodifikasi. Struktur data yang dikirimkan melalui URL terdiri dari tiga komponen utama, 16 byte pertama digunakan sebagai IV (Initialization Vektor), 16 byte berikutnya merupakan data terenkripsi dan 32 byte terakhir adalah HMAC yang berfungsi untuk memverifikasi integritas data.



Gambar 9 Struktur data

Gambar 9 merupakan Struktur data yang dikirim melalui URL memiliki fungsi masing-masing. IV (Initialization Vector) berfungsi untuk mempermudah proses dekripsi, karena dalam proses dekripsi dengan algoritma AES-256-CBC, IV yang digunakan harus sama dengan IV yang digunakan dalam proses enkripsi. Data terenkripsi berfungsi untuk menyimpan informasi asli dalam bentuk yang dienkripsi menggunakan algoritma AES-256-CBC, yang nantinya akan di dekripsi. HMAC berfungsi untuk memverifikasi integritas data.

Struktur data yang telah dimodifikasi dapat menyebabkan perubahan pada ukuran data yang dikirimkan. Parameter URL yang dimodifikasi, baik melalui serangan atau manipulasi lainnya, hal ini akan mempengaruhi keseluruhan data yang dikirimkan, termasuk IV, data terenkripsi dan HMAC. Perubahan pada struktur data mengakibatkan perubahan pada nilai HMAC. Nilai HMAC yang berubah menyebabkan proses dekripsi gagal, karena HMAC secara ketat memverifikasi bahwa data belum diubah, setiap usaha untuk memodifikasi data selama pengiriman akan menghasilkan ketidakcocokan HMAC dan sistem akan segera menghentikan proses dekripsi untuk mencegah serangan. Penggunaan HMAC memberikan lapisan keamanan tambahan yang kuat, memastikan bahwa penyerang mencoba menginjeksi kode jahat melalui parameter URL, modifikasi tersebut akan terdeteksi dan serangan tidak akan berhasil. Hal ini menjadikan kombinasi enkripsi algoritma AES-256-CBC dan HMAC SHA-256 sebagai mekanisme yang sangat efektif dalam melindungi aplikasi web dari serangan SQL Injection.

Berdasarkan hasil perbandingan penelitian terkait pengamanan parameter URL, menunjukan bahwa metode yang digunakan dalam penelitian ini lebih unggul dalam hal perlindungan terhadap modifikasi data.

pISSN: 2252-486X
eISSN: 2548-4710

TABEL 13
PERBANDINGAN PENELITIAN

Peneliti	Algoritma/ Teknik	Integritas Data
Hamid Wijaya[9]	AES-128	Tidak Ada
Theodora Tantri Trisnawati dkk.[13]	RSA	Tidak Ada
Asih Indriati[1]	AES	Tidak Ada
Mochammad Firman Arif dkk.[14]	Base64 dan Rotation13 (Encoding)	Tidak Ada
Taufiq Hidayatullah [15]	Base64 (Encoding)	Tidak Ada
Penelitian ini	AES-256-CBC dan SHA-256	HMAC SHA-256

Beberapa penelitian terdahulu, seperti yang dilakukan oleh Hamid Wijaya [9] dan Asih Indriati [1], menggunakan AES tanpa mekanisme untuk memastikan integritas data. Theodora Tantri Trisnawati dkk. [13], menggunakan algoritma RSA, tetapi juga tanpa perlindungan integritas. Penelitian lain, seperti yang dialkukan oleh Mochammad Firman Arif dkk. [14] dan Taufiq Hidayatullah [15], menggunakan teknik encoding seperti Base64 dan Rotation13, yang tidak menawarkan tingkat keamanan yang memadai untuk melindungi data dari manipulasi. Penelitian ini menonjol karena menggunakan kombinasi enkripsi AES-256-CBC yang kuat bersama dengan HMAC SHA-256, yang tidak hanya mengenkripsi data tetapi juga memastikan bahwa setiap perubahan atau modifikasi pada data akan langsung terdeteksi

V. KESIMPULAN DAN SARAN

Penelitian ini menunjukkan bahwa kombinasi enkripsi algoritma AES-256-CBC dan HMAC SHA-256 sangat efektif dalam melindungi data parameter URL dari serangan, termasuk SQL Injection, baik melalui pengujian manual atau otomatis menggunakan SQLMap. Struktur data yang dihasilkan terdiri dari IV sepanjang 16 byte, data terenkripsi dengan AES-256-CBC sepanjang 16 byte, dan HMAC SHA-256 sepanjang 32 byte yang menjamin integritas data dengan mendeteksi setiap modifikasi u\yang dilakukan oleh pihak yang tidak berwenang. AES-256-CBC memastikan kerahasiaan data, sementara HMAC SHA-256 memastikan integritasnya.

Saran dari penelitian ini, pengamanan terus ditingkatkan dengan menggunakan algoritma yang lebih kuat dan lebih baru seperti SHA-3, yang memiliki tingkat keamanan lebih tinggi dan lebih tahan terhadap serangan di masa mendatang. Selain itu, penting untuk melakukan uji coba keamanan tidak hanya terhadap SQL Injection, tetapi juga berbagai jenis serangan cyber lainnya seperti CrossSite Scripting (XSS), Cross-Site Request Forgery (CSRF), serangan brute-force dan serangan lainnya.

REFERENSI

[1] A. Indriati, F. T. Informasi, P. Studi, and T. Informatika, “Penerapan algoritma aes pada keamanan url studi kasus website mahasiswa atma luhur,” vol. 1, 2023.

[2] J. Pt and G. Informasi, “Implementasi Intrusion Prevention System Suricata,” no. 672015239, 2019.

[3] Dola Ramalinda, Jayadi, and Agung Rachmat Raharja, “Strategi Perlindungan Data Menggunakan Sistem Kriptografi Dalam Keamanan Informasi,” *J. Int. Multidiscip. Res.*, vol. 2, no. 6, pp. 665–671, 2024, doi: 10.62504/jimr679.

[4] D. Nanda, R. Herlambang, N. Pravitasari, P. Korespondensi, and I. Pendahuluan, “Penerapan Kriptografi AES untuk Keamanan Data Aplikasi Pemesanan Bibit Ternak pada BPSI UAT,” *Remik*, vol. 8, pp. 29–44, 2024.

[5] I. P. A. Eka Pratama, “Pengujian Performansi Lima Back-End JavaScript Framework Menggunakan Metode GET dan POST,” *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 6, 2020, doi: 10.29207/resti.v4i6.2675.

[6] B. A. Iswara, J. Tarigan, S. Man, and

A. Candra, "Comparison AES and RC4 Algorithm for Secure Data Passed Through an URL," *J. SAINTIKOM (Jurnal Sains Manaj. Inform. dan Komputer)*, vol. 22, no. 2, p. 357, 2023, doi: 10.53513/jis.v22i2.8400.

[7] R. Hermawan, "Teknik Uji Penetrasi Web Server Menggunakan SQL Injection dengan SQLmap di Kalilinux," *STRING (Satuan Tulisan Ris. dan Inov. Teknol.*, vol. 6, no. 2, p. 210, 2021, doi: 10.30998/string.v6i2.11477.

[8] A. Riyanti, B. M. Rahmanto, D. R. Hardianto, R. D. A. Yuristiawan, and A. Setiawan, "Uji Penetrasi Injeksi SQL terhadap Celah Keamanan Database Website menggunakan SQLmap," *J. Internet Softw. Eng.*, vol. 1, no. 4, p. 9, 2024, doi: 10.47134/pjise.v1i4.2623.

[9] H. Wijaya, "Jurnal Akademika Penerbit Implementasi Kriptografi Aes-128 Untuk Mengamankan Url (Uniform Resource Locator) Dari Sql Injection," *J. Akad.*, vol. 17, no. 1, pp. 8–13, 2020, [Online]. Available: https://www.ejournal.lppmunidayan.ac.id/index.php/akd

[10] M. L. Assidiq, F. Mahardika, and D. Santika, "Implementation of Aes and Sha-3 Cryptography Algorithms in Securing User Sensitive Data on the Artesian Water Bill Payment Transaction Website," *JOCSIT .. J. Collab. Sci. Informatics Technol.*, vol. 1, no. 1, pp. 41–54, 2024, doi: 10.69933/jocsit.v1i1.63.

[11] M. R. Rabtsani, A. Triayudi, and G. Soepriyono, "Kombinasi AES (Advanced Encryption Standard) dan Algoritma SHA256 untuk Keamanan Data dalam Pembayaran Tagihan Aplikasi," vol. 2, pp. 1–19, 2024, doi: 10.58905/SAGA.vol2i1.250.

[12] P. Sravan and S. Verdandi, "Building a Secure and High-Performance HMAC Processor Based on SHA-3 for the Cloud-Based Financial Sector," *Proc. 2nd IEEE Int. Conf. Adv. Comput. Commun. Appl. Informatics, ACCAI 2023*, 2023, doi: 10.1109/ACCAI58221.2023.10200480.

[13] T. T. Trisnawati, S. Yurinanda, W. Syafmen, and C. Multahadah, "Penerapan Algoritma Rivest-Shamir-Adleman (RSA) pada Enkripsi Uniform Resource Locator (URL) Website untuk Keamanan Data," *Euler J. Ilm. Mat. Sains dan Teknol.*, vol. 11, no. 2, pp. 205–215, 2023, doi: 10.37905/euler.v11i2.21169.

[14] M. F. Arif and Muhammad Misdram, "Implementasi Enkripsi Url Pada Website Menggunakan Metode Base64 Dan Rotation13," *Spirit*, vol. 12, no. 1, pp. 20–25, 2020.

[15] T. Hidayatullah, "Implementasi Algoritma Base-64 Dalam Mengamankan Url (Uniform Resource Locator) Website Layanan Pengaduan Masyarakat Desa Bojongraharja," *J. Media Infotama*, vol. 18, no. 2, pp. 338–339, 2022, [Online]. Available: https://jurnal.unived.ac.id/index.php/jmi/article/view/2937%0Ahttps://jurnal.unived.ac.id/index.php/jmi/article/download/2937/2606

pISSN: 2252-486X
eISSN: 2548-4710